ESTADO DEL ARTE REVISIÓN SISTEMÁTICA DE LA SEGURIDAD ORIENTADA A REST

Nidia Estefanía Corredor Ceballos

Universidad Católica de Colombia
Facultad de Ingeniería
Ingeniería de Sistemas
Bogotá D.C., Colombia
2017

ESTADO DEL ARTE REVISIÓN SISTEMÁTICA DE LA SEGURIDAD ORIENTADA A REST

Nidia Estefanía Corredor Ceballos

Trabajo de Grado para optar al título de
Ingeniero de Sistemas

Director
M. Sc. Mario Martínez
Ingeniero de Sistemas

Universidad Católica de Colombia
Facultad de Ingeniería
Ingeniería de Sistemas
Bogotá D.C

2017

Nota de aceptación

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

_____
Presidente del Jurado

_____
Jurado

_____
Jurado

Bogotá, 12, noviembre, 2017

DEDICATORIA

A:

Dios por darme la oportunidad de vivir y culminar una etapa más de vida, estar conmigo y bendecirme en cada momento, dar fortaleza a mi corazón e iluminar mi mente, pero, sobre todo, por haber puesto en mi camino a grandes personas que han sido un cimiento y compañía durante todo el tiempo de mi vida.

Mis padres quienes son mi fuente de inspiración, amor y dedicación para alcanzar mis metas profesionales y personales, quiero agradecer inmensamente su apoyo y dedicación, en la construcción de este gran proyecto de vida. Mis hermanos, por ser mi mejor compañía en los momentos oportunos de mi vida y a todos mis amigos que estuvieron en esos momentos cuando necesitaba una palabra de apoyo.

AGRADECIMIENTOS

Quiero darle mi agradecimiento al director de esta tesis, el Ingeniero Mario Martínez, quien me ha ayudado a construir este trabajo, con continuo conocimiento, apoyo y motivación. Esta tesis es el resultado de un trabajo en equipo, pero sin lugar a duda no hubiera sido posible sin su ayuda.

A mis compañeros que día a día se convirtieron en mis grandes amigos, por darme su amistad, su tiempo, su apoyo y sus consejos. Al programa de Ingeniería de Sistemas, por abrirme sus puertas y permitirme ser un miembro de una gran familia y a todos mis amigos que de una u otra manera pusieron su granito de arena y sus mejores deseos en la culminación de este trabajo.

CONTENIDO

	pág.
INTRODUCCIÓN	13
1. SERVICIOS WEB REST	17
1.1 REST	17
1.2 SERVICIOS WEB	20
2. SEGURIDAD WEB	22
2.1 SEGURIDAD EN SERVICIOS WEB REST	22
2.2 TIPOS DE VULNERABILIDADES QUE AFECTAN LA SEGURIDAD WEB	60
3. VULNERABILIDADES DE LOS SERVICIOS WEB	100
3.1 PRINCIPALES TAXONOMÍAS DE VULNERABILIDADES	100
3.2 TAXONOMÍA BASADA EN PROCESOS – ORIENTACIÓN AL EVENTO	101
4. METODOLOGÍA	108
4.1 PLANIFICACIÓN DE LA REVISIÓN SISTEMÁTICA	108
4.2 EJECUCIÓN DE LA REVISIÓN SISTEMÁTICA	110
4.3 REPORTE DE REVISIÓN	113
5. RESULTADOS	116
6. CONCLUSIONES	136
7. RECOMENDACIONES	138
REFERENCIAS	139

LISTA DE TABLAS

pág.

Tabla 1. Principales amenazas que afectan a las cookies de autenticación 44
Tabla 2. Detalles de las técnicas defensivas XSS recientes 48
Tabla 3. Ejemplo del conjunto de datos del resultado del ataque unión consulta 67
Tabla 4. Amenazas de inyección SQL 71
Tabla 5. Herramientas Inyección SQL 70
Tabla 6. Clases de ataques y su descripción 81
Tabla 7. Estadísticas de ataques XSS sobre OSN 83
Tabla 8. La categoría de las reacciones del sistema malo 87
Tabla 9. Protocolo revisión sistemática 110
Tabla 10. Artículos seleccionados 103

LISTA DE FIGURAS

pág.

Figura 1. Grafo generador de tokens **¡Error! Marcador no definido.**
Figura 2. Arquitectura para el almacenamiento de valores en la base de datos **¡Error! Marcador no definido.**
Figura 3. Algoritmo por autenticación 37
Figura 4. Algoritmo para recuperar los valores de la base de datos **¡Error! Marcador no definido.**
Figura 5. Gestión de memoria Pseudo código **¡Error! Marcador no definido.**
Figura 6. Algoritmo para extraer archivos JavaScript externos (JS) **¡Error! Marcador no definido.**
Figura 7. Pseudocódigo para generador de metadatos estructurales **¡Error! Marcador no definido.**
Figura 8. Sintaxis de programa **¡Error! Marcador no definido.**
Figura 9. Programa ejemplo **¡Error! Marcador no definido.**
Figura 10. Inyección SQL paso a paso **¡Error! Marcador no definido.**
Figura 11. Ataque referencia insegura a objetos **¡Error! Marcador no definido.**
Figura 12. Script para cambiar la contraseña a través de CSRF **¡Error! Marcador no definido.**
Figura 13. Taxonomía basada en procesos – orientación al evento **¡Error! Marcador no definido.**
Figura 14. Taxonomía 99
Figura 15. Artículos por fecha **¡Error! Marcador no definido.**
Figura 16. Artículos por idioma 113
Figura 17. Artículos base de datos **¡Error! Marcador no definido.**
Figura 18. Clasificación de Taxonomía Inyección de SQL 112
Figura 19. Clasificación de Taxonomía Pérdida de autenticación y administración de sesión 114
Figura 20. Clasificación de Taxonomía Cross-site Scripting (XSS) 117
Figura 21. Clasificación de Taxonomía Referencias Inseguras a Objetos Directos 118
Figura 22. Clasificación de Taxonomía Configuración errónea de seguridad 120
Figura 23. Clasificación de Taxonomía Exposición de datos sensibles 121
Figura 24. Clasificación de Taxonomía Falta de control de acceso de nivel de funciones 123
Figura 25. Clasificación de Taxonomía Cross-Site Request Forgery (CSRF) 124
Figura 26. Clasificación de Taxonomía Uso de componentes con vulnerabilidades conocidas 126

Figura 27. Clasificación de Taxonomía Redireccionamiento y reenvío no validado 127

GLOSARIO

CACHE-CONTROL: Es una de las cabeceras de la memoria cache es única porque no solo permite establecer una opción de configuración de cache sino varias, Cache-control: private, max-age=0, must-revalidate.

CRAWLER: Es un programa que inspecciona las páginas Web de forma metódica. Crawler o araña Web hace una copia de las páginas Web que han sido visitadas a través de un motor de búsqueda.

FUZZING: Es una técnica que se emplea para realizar pruebas de software.

HTML: Lenguaje de marcas de hipertexto, hace referencia al lenguaje de marcado para la elaboración de páginas Web.

HTTP: Es un protocolo de transferencia de hipertexto que permite comunicación y transferencia de información.

JCRACHER: Es una herramienta de prueba para código Java.

TAXONOMÍA: Es la ciencia de la ordenación o clasificación.

SERVIDOR WEB: Es un programa que se ejecuta en el lado del servidor.

URI: Es un identificador de recursos de una red que consta de esquema, autoridad, ruta, consulta y fragmento.

URL: Es la ruta que se ubica en el lado del navegador, la cual sirve para solicitar un recurso en un servidor.

WEB: El concepto se utiliza en el ámbito tecnológico para nombrar a una red informática o a la internet.

WEBSOCKETS: Es un canal de comunicación bidireccional y full-duplex. Está diseñada para ser implementada en navegadores y servidores Web, pero puede utilizarse por cualquier aplicación cliente/servidor.

XML: Es un lenguaje de marcado generalizado simplificado y adaptado a Internet

LISTADO DE ACRÓNIMOS

API: Interfaz de Programación de Aplicaciones
BSF: Negocios Servicios financieros y de seguros
BSO: Negocios otros
BSR: Negocios ventas al por menor/comercio
CSRF: Solicitud de falsificación de sitio cruzado
DBMS: Sistema de gestión de base de datos
DOM: Modelo de objetos del documento
DSL: Línea de suscripción digital
EDU: Instituciones educativas
GOV: Gobierno y Ejército
HMAC: Código de autenticación de mensaje
IDS: Sistema de detección de intrusos
NGO: Organizaciones sin ánimo de lucro
OSN: Redes sociales en línea
OTC: Cookies una vez
OWASP: Proyecto de seguridad de aplicaciones Web
REST: Transferencia de Estado Representacional
SDLC: Ciclo de vida del desarrollo de programas
SQLIA: Inyección de ataque SQL
URL: Localizadores uniformes de recursos
WAF: Firewall de aplicaciones Web
XSS: Secuencia de comandos de sitio cruzado

RESUMEN

REST es un estilo de arquitectura para el diseño de servicios Web que aprovecha al máximo los recursos de la Web por lo que es eficiente y fácil de implementar en comparación con arquitecturas como SOAP. Sin embargo, REST tiene riesgos en la implementación de servicios Web que son importantes tener en cuenta para no poner en producción un sistema vulnerable. Los principales afectados son las empresas y personas que acceden a la información distribuida en la red para exponer sus datos.

En este trabajo se analizan los principales tipos de ataque, las vulnerabilidades más conocidas y los tipos de seguridad en los servicios Web REST. Se hizo mediante la revisión sistemática de las publicaciones sobre los tipos de vulnerabilidad de Servicios Web REST los cuales se analizaron tomando como referencia una taxonomía de vulnerabilidades con el fin de desarrollar la estructura de los tipos seleccionados.

Se conocieron las vulnerabilidades de los servicios Web REST y se identificaron sus características tales como: el atacante, la herramienta, la vulnerabilidad, el blanco, el resultado no autorizado, el objetivo, el tipo de organización, el tipo de ataque y la debilidad de la aplicación. Se identificaron los principios de seguridad (integridad, disponibilidad, y confidencialidad) vulnerados por los tipos existentes en los servicios Web.

Palabras claves: REST, Seguridad, Servicios Web, Vulnerabilidad.

INTRODUCCIÓN

REST es un estilo de arquitectura software para sistemas distribuidos. Este concepto se originó en el año 2000 por Roy Fielding, en una tesis doctoral sobre la Web. REST ha pasado a ser utilizada por el área de desarrollo como una oportunidad de creación de negocios. REST ha cambiado la manera de hacer Ingeniería de software y en el campo de las APIs, es la base del desarrollo de servicios de aplicaciones. REST es considerada al día de hoy, una arquitectura de principios que permite crear servicios profesionales y aplicaciones que son usadas por dispositivos electrónicos y por los clientes HTTP. En todo proyecto o aplicación se dispone de un API REST. En los servicios de internet en la creación de APIs REST, es un estándar eficiente y lógico.

Una de las ventajas que ofrece REST para el desarrollo de software, es la implementación de una arquitectura cliente y servidor, que hace que los sistemas sean eficaces y sus atributos de calidad como portabilidad permita que se ejecute en diferentes plataformas, adaptándose a las necesidades de rendimiento a medida que la cantidad de usuarios aumentan y admite que los componentes de los desarrollos se actualicen eficientemente. Las aplicaciones basadas en REST son flexibles, se pueden migrar a otros servidores y dan la posibilidad de hacer cambios en la base de datos contando que las peticiones sean correctas y se adapten a todo tipo de plataforma de trabajo.

Los desarrolladores eligen la arquitectura REST como la API que se ajusta a las necesidades del mercado laboral y su éxito respecto a otras alternativas arquitecturales de software revelan la importancia de hacer una revisión sistemática de las publicaciones que analizan los problemas de seguridad, con el fin de conocer el estado de los tipos de vulnerabilidad críticos, frecuentes y como se pueden mitigar a través de distintos mecanismos de seguridad en los servicios Web.

Los servicios Web es un conjunto de aplicaciones o tecnologías con alto grado de interoperabilidad en la Web. Estas aplicaciones intercambian información lo que hace posible ofrecer servicios que a su vez están expuestos a numerosos ataques

que hacen vulnerables los sistemas, afectando la seguridad. Los servicios Web están compuestos por dos partes independientes, una de ellas es la aplicación que usa el servicio Web y la otra es el servicio.

La seguridad Web son los mecanismos que las empresas que trabajan en internet implementan para minimizar el impacto de los ciberataques, cualquier sistema es vulnerable. Los sitios Web, son propensos a ser el blanco de los ataques ocasionados por los piratas informáticos, la falta de conocimiento en temas de ciberseguridad de los trabajadores trae riesgos considerables para la organización y la falta de mantenimiento, actualización de las aplicaciones en el servidor Web abre una ventana de acceso a la ciberdelincuencia que puede convertir a las empresas en víctimas lo que llevaría a la pérdida de millones de pesos o el fin de la organización. En este trabajo se utiliza el término de tipo de vulnerabilidad, pero en realidad se usan muchos nombres diferentes para referirse al mismo concepto, específicamente, se tienen nombres, tales como, tipo de ataque, tipo de vulnerabilidad, riesgos, amenazas para referirse a una vulnerabilidad de los servicios Web.

La seguridad Web REST es el conjunto de recomendaciones de seguridad para servicios REST, dado que REST es uno de los métodos más utilizados para publicar servicios Web. Entre las recomendaciones de seguridad es relevante revisar las vulnerabilidades conocidas como: Autenticación y Gestión de la sesión, protección de métodos HTTP, protección de acciones privilegiadas y los recursos sensibles, protección contra la falsificación entre muchos más utilizados. Desde hace varios años el desarrollo de aplicaciones ha tenido un gran crecimiento no solo a nivel interno, sino también a nivel externo de la empresa, lo que genera problemas de seguridad. OWASP se ha dedicado a publicar sus revisiones sobre los riesgos a los que están expuestos los servicios Web REST, lo que se debe tener en cuenta para no poner en producción un sistema vulnerable. Esas publicaciones han dado lugar a seleccionar los riesgos más populares y así mismo son referenciados en artículos científicos, trabajos de postgrado, libros de seguridad y organizaciones.

El problema que REST presenta es la falta de seguridad y cualquier persona malintencionada puede interceptar los mensajes HTTP y leerlos, lo que promueve a la necesidad de incrementar algún tipo de seguridad. Las ventajas que ofrece REST en cuanto a portabilidad, escalabilidad y actualización eficiente traen que por su propia naturaleza los servicios Web RESTful se vuelven más vulnerables en la red no solo frente a la interceptación de mensajes sino a otro tipo de ataques como la suplantación de identidad. Por ello es necesario incorporar mecanismos de seguridad a los servicios Web utilizando los métodos apropiados que, a su vez, tiene varias formas de implementarse. Existen muchas recomendaciones sobre seguridad en servicios SOAP pero nada de ello aplica directamente sobre REST. Debido a este problema, cada desarrollador investiga una forma, cometiéndose muchos fallos.

De acuerdo con lo expuesto surge la pregunta objeto de estudio, **¿**Cuáles son las debilidades que se presentan y ponen en riesgo la seguridad en los Servicios Web REST**?**

Para resolver este problema se debe cumplir el siguiente objetivo general y objetivos específicos.

OBJETIVO GENERAL

Desarrollar una revisión sistemática basada en tipos de vulnerabilidad de Servicios Web REST.

OBJETIVOS ESPECÍFICOS

- Seleccionar los tipos de vulnerabilidades actuales que afectan la seguridad de Servicios Web, considerando los componentes cliente y servidor.
- Definir la taxonomía y desarrollar la estructura de los tipos seleccionados.
- Evaluar la taxonomía propuesta, identificando los principios de seguridad vulnerados por los tipos y fines, existentes en los Servicios Web.

Para identificar las debilidades que se presentan y ponen en riesgo la seguridad en los servicios Web REST se hace mediante la revisión sistemática de las publicaciones científicas que se encuentran en revistas especializadas en temas de seguridad de la información en servicios Web REST. La metodología de la revisión sistemática es una herramienta útil que permite, entre otros, obtener información de calidad y actualizada. La necesidad de realizar el estado del arte es presentar a las empresas, a los usuarios un trabajo que ayude a tener una visión clara sobre las vulnerabilidades que se presentan en los sistemas de información. OWASP dice que es crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones. Se analizan las publicaciones de los años 2013 hasta el 2017 que investigan el problema de la seguridad de los servicios Web REST. Las limitaciones contextuales hacen referencia a las debilidades que ponen en riesgo la seguridad en los servicios Web REST seleccionados que abarcan la investigación, lo que hace posible tener un conocimiento de las debilidades comunes en los servicios Web. El presente estado del arte abarca los tipos de vulnerabilidad enumerados en el proyecto de seguridad de aplicación abierta OWASP Top 10 2013 porque ser los ataques más graves que se presentan en los servicios Web. El proyecto OWASP, está dedicado a determinar y combatir las causas que hacen una aplicación insegura. Esta es una organización con una visión de educar que trabaja con el objetivo de fortalecer el área de seguridad de

las aplicaciones Web. Por otro lado, estudian los mecanismos de seguridad informática la cual se encarga de la seguridad de sitios Web, aplicaciones Web y servicios Web. La seguridad de aplicaciones Web se basa en los principios de seguridad de aplicaciones como son integridad, disponibilidad y confidencialidad.

Se utiliza la taxonomía como instrumento de análisis. La taxonomía de los servicios Web surge como un concepto que se aplica al campo informático desde principios de los años 90, para contextualizar a los sistemas en temas de acceso a la información. En la actualidad se trabaja con sistemas que se dedican a manejar grandes volúmenes de información contenidos en internet.

Uno de los principales beneficios de las taxonomías es que permite clasificar las vulnerabilidades desde distintos enfoques para estudiar su estructura o descubrir nuevas vulnerabilidades. Esto es de ayuda para los diseñadores quienes deben entender primero las posibles amenazas antes de diseñar sistemas seguros. Para que implementen estándares de seguridad o herramientas lógicas como depuraciones, firewall o verificaciones en las aplicaciones Web que ayuden a atenuar el impacto de un problema de segurida.

Este trabajo está organizado de la siguiente manera: en el capítulo 1 se define la arquitectura REST y los servicios WEB, en el capítulo 2 se describe la seguridad Web, en el capítulo 3 se describe la estructura de las vulnerabilidades de los servicios Web, en el capítulo 4 se detalla la metodología de la revisión sistemática, en el capítulo 5 se presentan los resultados de la aplicación de la taxonomía basada en procesos, en el capítulo 6 se exponen las conclusiones y en el capítulo 7 las recomendaciones.

1. SERVICIOS WEB REST

En este capítulo se presenta la definición, servicios y servicios Web y el estilo de arquitectura REST, conceptos que son utilizados cuando se describa la seguridad en servicios Web, la taxonomía de las vulnerabilidades y vulnerabilidades en servicios Web REST.

1.1 REST

El estilo de arquitectura Transferencia de Estado Representacional (Representational State Transfer) o REST que se expone a continuación está basado en la disertación de Roy Thomas Fielding [62]. REST, es un estilo de arquitectura diseñado especialmente para sistemas distribuidos de hipermedios cuyo término acuñó Roy Thomas Fielding en su disertación. Su funcionalidad se da por el cumplimiento de unos principios que consiste en un conjunto de restricciones aplicadas a elementos dentro de la arquitectura. REST se caracteriza por ser escalable, por la generalidad de sus interfaces y el despliegue independiente de los componentes. Los servicios Web basados en este estilo de arquitectura, publican un conjunto de recursos relacionados donde los clientes pueden acceder a través de un identificador de recursos uniforme (Uniform Resource Identifier) URI e interactuar con una interfaz uniforme. Vincular los recursos implica que los clientes suelen realizar múltiples interacciones para lograr su objetivo.

Es posible utilizar el concepto de servicio de conversación, donde los sistemas de mensajería, indican un conjunto de protocolos básicos por ejemplo el protocolo de transferencia de hipertexto (Hypertext Transfer Protocol) HTTP y las interacciones de solicitud de respuesta que son impulsadas por el mismo cliente que interactúa con uno o más servicios Web Restful.

REST hace posible la comunicación entre un cliente y una Interfaz de programación de Aplicaciones (Application Programming Interface) API Web

Restful. La API genera la comunicación entre los componentes de software. En la actualidad, existen frameworks de desarrollo de software que cuentan con un patrón llamado CRUD (CREATE, RECOVER, UPDATE, DELETE), que son verbos para crear aplicaciones y servicios Web que facilitan el manejo de la información en la base de datos operando con el lenguaje SQL con los comandos INSERT, SELECT, UPDATE y DELETE [1].

PRINCIPIOS

- **Identificación de recursos.** Para acceder a un recurso se hace por un medio de un identificador único URI, que se ubica en la última parte de un Localizador Uniforme de Recursos (Uniform Resource Locator) URL que hace el papel de archivo que contiene la información solicitada y el resto de la dirección son directorios lógicos que llevan el recurso.

- **Manipulación de recursos.** El uso de recursos HTTP son accedidos mediante las operaciones GET, POST, PUT, DELETE. Un API REST funciona con mensajes o recursos que son accesibles por los URIs. Al solicitar un recurso mediante el protocolo HTTP en el navegador se ejecutan los métodos GET, POST, PUT, DELETE. Si se desea realizar una petición al servidor, se hace por medio del método GET, el método POST es el encargado de crear un nuevo recurso y modificarlo en el servidor, para actualizar un archivo se utiliza el método PUT y para borrar un recurso es con el método DELETE.

- **Representación de contenido.** El intercambio de datos durante la comunicación entre el cliente y el servidor se puede realizar con formatos de lenguaje independientes como JavaScript Object Notation JSON o el lenguaje de Marcas Extensible (Extensible Markup Language) XML**.**

- **Comunicación sin estado.** Cada mensaje HTTP contiene toda la información necesaria para comprender la petición lo que permite que ni el cliente ni el servidor necesiten recordar ningún estado de las comunicaciones entre mensajes.

- **HATEOAS**. Acrónimo de Hypermedia As The Engine Of Application State (hipermedia como motor del estado de la aplicación). Es un cliente que interactúa con una aplicación de red proporcionada por los servidores de aplicaciones. El servidor devuelve la representación de un recurso, donde parte de la información devuelta serán identificadores únicos en forma de hipervínculos a otros recursos asociados

Una vez puesto en funcionamiento REST estará en capacidad de integrar varios sistemas de información sin importar la plataforma, lenguaje o modelo de objeto.

En algunos casos los encargados de implementar esta arquitectura saben cuándo y dónde guardar las respuestas a ciertas peticiones, lo que incrementa el rendimiento y disminuye la latencia. REST se destaca de otros estilos de arquitectura por la mantenibilidad que ofrece.

Una arquitectura REST debe cumplir con las siguientes restricciones específicas:

RESTRICCIONES

- **Arquitectura Cliente – Servidor.** REST se basa en dos agentes básicos Cliente y Servidor e independientes que intercambian información lo que permite una alta flexibilidad. También se conoce como una arquitectura por capas, donde el objetivo es separar la capa lógica de negocios de la lógica de diseño, de esta manera el cliente conserva su independencia de dichas capas.

- **Stateless.** Sin estado. Es un protocolo que no maneja estado de comunicaciones lo cual hace que cada petición sea una transacción que no tenga relación con solicitudes anteriores, de modo que la comunicación se compone de solicitud y respuesta. El servidor no debe almacenar información del cliente, por lo que ya se empieza a utilizar tecnologías que implementan el estado dentro de la arquitectura, como WebSockets.

- **Cacheable.** Es un atributo conveniente para almacenar en caché. El servidor define el modo de cachear las peticiones lo que hace posible mejorar el rendimiento y la escalabilidad. El protocolo HTTP lo implementa con la cabecera Cache Control.

- **Interfaz uniforme.** No se requiere que la interfaz de comunicación entre el cliente y el servidor dependa de ellos, esto garantiza que haya una interfaz establecida, y que no debe importar quien haga o reciba las peticiones.

ELEMENTOS ARQUITECTÓNICOS DE REST

A continuación se describen los elementos que permiten formar el diseño de REST, tal como lo concibió Fielding.

En la arquitectura REST se realiza una conexión entre el cliente y el servidor donde posiblemente se puede tener distintos móviles y clientes Web haciendo solicitudes a través del protocolo de transmisión HTTP a un servidor proxy. El servidor proxy atiende la solicitud y busca darle la respuesta al cliente, entonces inicia un proceso de búsqueda de recursos REST desde la API de autenticación que expone la funcionalidad de identidad de Auth y verifica en el almacenamiento de cache. Los recursos REST necesitan una sincronización de procesos lo que hace posible una restricción de la arquitectura REST llamada HATEOAS la cual

desacopla el cliente y el servidor de una manera que hace que las funciones del servidor sean independientes. El recurso solicitado se devuelve en formato estándar XML [2].

VISTAS DE LA ARQUITECTURA REST

- **Vista de proceso**. La vista de proceso de la arquitectura permite analizar las relaciones entre los componentes desde el punto de vista del camino que siguen los datos a través del sistema.

- **Vista de conector**. Se centra principalmente en los mecanismos de comunicación entre componentes. Para arquitecturas basadas en REST, son las restricciones que definen la interfaz genérica. El conector del cliente examina el identificador de recurso para seleccionar el mecanismo de comunicación apropiado para cada petición.

- **Vista de datos.** Deja ver el estado de la aplicación como un flujo de información a través de los componentes. REST está concebido para sistemas de información distribuidos, por lo tanto, ve una aplicación como una estructura de información y alternativas de control a través de la que un usuario puede realizar una tarea. Cada aplicación define sus objetivos para los sistemas subyacentes. La interacción de los componentes ocurre en forma de mensajes de tamaño dinámico. Los mensajes pequeños o medianos se usan para el control de la semántica, pero la mayor parte de la aplicación trabaja con grandes mensajes que contienen una representación completa del recurso. La forma más frecuente de semántica para peticiones es solicitar la representación de un recurso, (como por ejemplo el método GET de HTTP), que puede ser almacenado en caché para un uso posterior.

1.2 SERVICIOS WEB

La descripción de los Servicios Web que se expone a continuación está basada en un documento público de Working Group Note producido por W3C Web Services Architecture Working Group, que es parte de la W3C Web Services Activity.

La World Wide Web Consortium define un Servicio Web como "… un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red. Tiene una interface descrita en un formato procesable por una máquina (específicamente WSDL). Otros sistemas interactúan con el servicios web en una manera prescrita por su descripción usando mensajes SOAP, típicamente enviados usando HTTP con una serialización XML en relación con otros estándares relacionados con la web [63]".

Un servicio Web es una interfaz que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML estandarizada. También se puede definir como un conjunto de tecnologías de software para el intercambio de datos entre aplicaciones tales como SOAP (Simple Object Acces Protocol), WDSL (Web Services Description Language) y UDDI (Universal Description, Discovery and Integration). Pueden ser desarrollados en muchos lenguajes e implementados sobre muchos tipos de redes. La interoperabilidad se consigue por la utilización de protocolos y estándares abiertos. La Web Services Protocol Stack es el conjunto de servicios y protocolos para los servicios web que son utilizados para definir, localizar, implementar y hacer que un servicio interactúe con otro. Está compuesto por: Servicio de transporte, Mensajería XML, Descripción del servicio y Descubrimiento de Servicios.

El servicio de transporte porta los mensajes entre aplicaciones sobre la red. Incluye varios protocolos del nivel de aplicación: HTTP, (File Transfer Protocol) FTP, SMTP (Simple Mail Transfer Protocol), BEEP (Block Extensible Exchange Protocol) y JMS (Java Message Service).

El servicio de mensajería XML es el encargado de los mensajes en XML estándar por lo que puede ser interpretado en cualquiera de los nodos de la red. Los más utilizados son: REST (Representational State Transfer); RPC (Remote Procedure Calls), XML-RPC, XML, SOAP.

La descripción del servicio que es un servicio de la web cuenta con una interfaz pública descrita en un formato WSDL.

El descubrimiento de servicios conocido como UDDI es un marco independiente de la plataforma para describir servicios, negocios e integrar servicios de negocios.

2. SEGURIDAD WEB

Al aumentar el uso de los servicios web, se plantean amenazas para atacarlos y, por lo tanto, se necesita políticas de seguridad para defenderlos contra los piratas informáticos. El cifrado XML1 y la firma XML son introducidos por W3C2 en 2002, para proteger los documentos XML contra los piratas informáticos mediante el uso de la criptografía. En 2007, el resultado de las investigaciones sobre la seguridad del servicio Web se reunió en el WS-Security estándar (Seguridad en Servicios Web). Esta norma se concentra en la seguridad de los mensajes y el uso de cifrado XML y la firma XML para obtener la confidencialidad y la integridad. WSDL3, es un documento que proporciona la información necesaria para llamar a los métodos de un servicio Web, es un punto vulnerable de los servicios Web. En algunos casos, los hackers usan los WSDL para obtener la información necesaria para atacar al servicio Web.

Aspectos funcionales: Se aplica a la observación de las normas de seguridad y se utiliza en aplicaciones tradicionales. Estos son: autenticación o identificación de los consumidores, autorización o decisión sobre la autoridad para realizar un acto, por un consumidor identificado, en una fuente específica, confidencialidad de los datos o para proteger los datos, la integridad de los datos o la detección de la falsificación en los datos y garantizar que cualquier remitente o receptor no puede modificar los datos y la protección contra ataques o asegura que los atacantes no puedan controlar los datos.

Aspectos no funcionales: Estas características, en contraste con las funcionales, se utilizan para garantizar que las soluciones orientadas al servicio funcionen adecuadamente en las organizaciones.

Interoperabilidad: Este concepto es particularmente útil para la arquitectura orientada al servicio e implica que las diferentes soluciones de seguridad no deben violar el ajuste de los servicios [3].

2.1 SEGURIDAD EN SERVICIOS WEB REST

La seguridad en servicios Web REST es un conjunto de mecanismos, procesos, mejoras, modelos, técnicas que ayudan a mitigar el nivel de impacto de los tipos de vulnerabilidad que sufren las organizaciones y personas que manejan servicios Web. Las principales amenazas estudiadas [55] son inyección de SQL, pérdida de autenticación y administración de sesión, cross-site scripting (XSS), referencias inseguras a objetos directos, configuración errónea de seguridad, exposición de datos sensibles, falta de control de acceso de nivel de funciones, cross-site request forgery (CSRF), uso de componentes con vulnerabilidades conocidas y redireccionamiento y reenvío no validado.

- **PROCESO DE SANITIZACIÓN CONTRA ATAQUES DE INYECCIÓN SQL**

Sanitización es un tipo particular de validación de entrada. Por lo general, elimina elementos potencialmente maliciosos de las fuentes de entrada. Se realiza antes del uso de los parámetros de entrada externos en las operaciones de destino. La mayoría de las soluciones y medidas preventivas pueden ser que una expresión regular o función de validación incorporada es aplicada a la fuente de entrada, el resultado es seguro de usar. Por desgracia, este no es siempre el caso. Por ejemplo, es posible aplicar una función de sanitización a ciertos valores maliciosos, sin embargo, esto no ofrece una protección completa contra todos los ataques de inyección SQL. En el análisis basado en el modelo de comprobación, los ataques falsos negativos que tienen éxito, pueden ocurrir porque las definiciones del patrón de ataque están completas.

Las herramientas generales, como un proxy o detección de intrusiones en el sistema IDS (sistema de detección de intrusos), también son en gran medida ineficaces contra SQLIA (Ataque de inyección SQL), que se realizan a través de los puertos utilizados para el tráfico Web regular. Las técnicas de armonización estructural, como un árbol de análisis, también sufren de falsos negativos cuando un árbol de análisis de la consulta de ataque coincide con una estructura esperada. De este modo, existe una técnica que estima el tamaño de resultado de la consulta para evitar el ataque de inyección de SQL. Este método estima automáticamente un costo de consulta seguro sustituido en cada ubicación de la consulta SQL. La esencia de la técnica es:

Determinar la información necesaria para comprobar los patrones de ataque de las consultas generadas por una aplicación, y
Aplicar una técnica de estimación para comparar estas consultas utilizando el tamaño del resultado de la consulta.

Por lo tanto, la técnica primero utiliza la propagación de la mancha para analizar el código de la aplicación y automáticamente las consultas legítimas que podrían ser generados por la aplicación. La técnica estima el tamaño de los resultados de la

consulta generada. Las consultas que tienen diferentes tamaños de resultado se clasifican como ilegales y se previenen de ser ejecutado en la base de datos.

Para verificar la sanitización de vulnerabilidades de consultas, se necesita información sobre el conjunto de valores retenidos por las variables del programa, en el enfoque, se define la ejecución simbólica, una técnica en la verificación del programa. Durante el tiempo de ejecución, la consulta SQL generada por un programa puede representarse como una expresión simbólica del conjunto de entradas del programa[4].

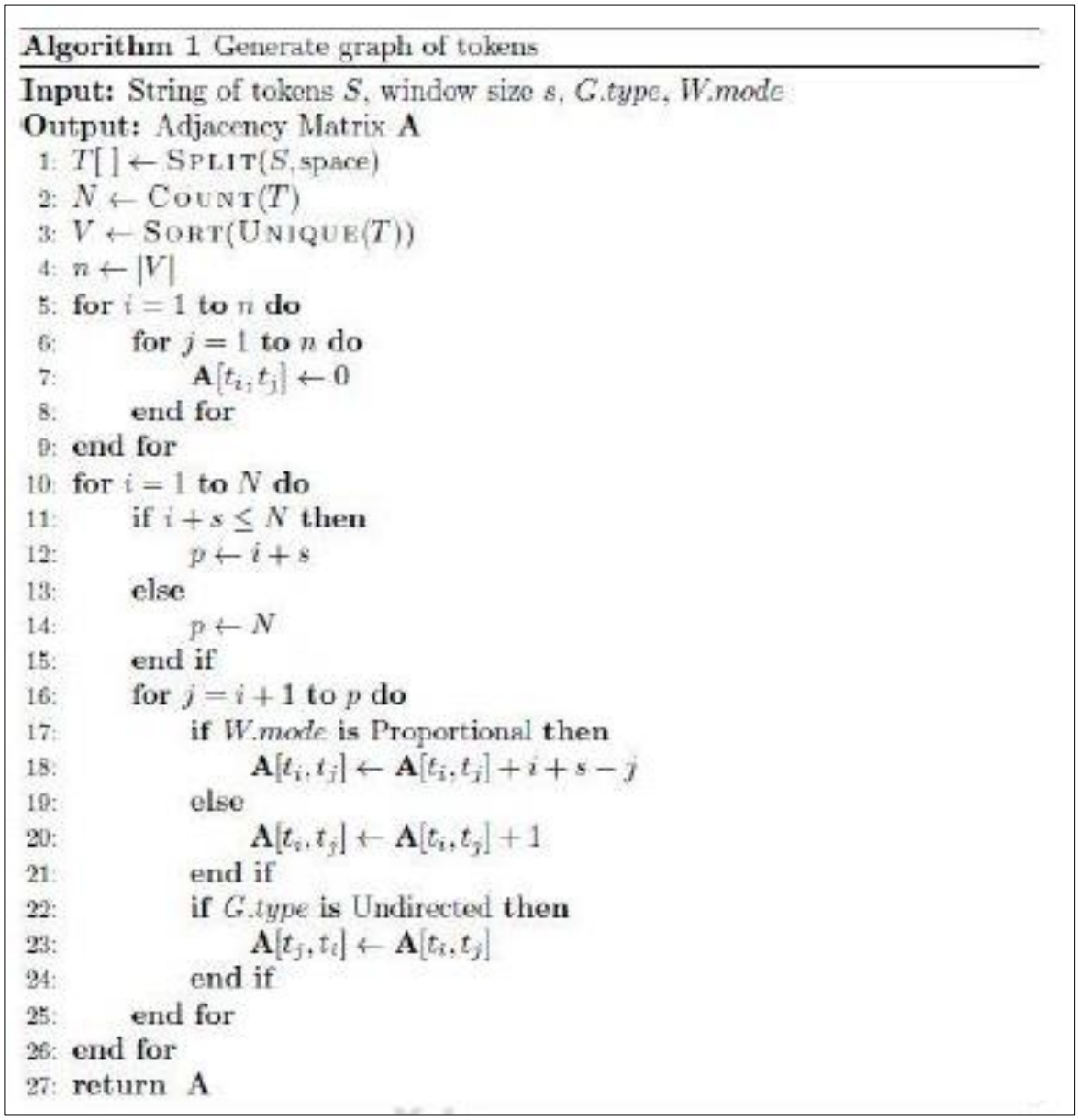
- **GRAFO DE TOKENS PARA DETECTAR SQLIA**

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

El grafo de tokens consiste en n nodos $t_1, t_2, \dots, t_n$ es representado por una matriz de adyacencia A de tamaño $n\,x\,n$. Las filas y las columnas de la matriz de adyacencia son indexadas por los nombres de los nodos (i.e., único tokens), y cada elemento $A[t_i, t_j] = w_{ij.}$ Si no hay un arista entre t_i y t_j entonces $w_{ij} = 0$. En un grafo no dirigido, la matriz de adyacencia es diagonalmente simétrica. En el caso de un grafo dirigido, las filas son los nodos de origen y las columnas son los nodos de destino. La matriz de adyacencia de un grafo dirigido es generalmente asimétrica. Dado que permite auto-bucles, en ambos tipos de grafos, algunos de los elementos diagonales $A\,[t_i, t_j], i = 1 \dots n,$ puede ser distintos de cero.

Los grafos se clasifican en cuatro tipos de grafos: no dirigido uniforme ponderado, no dirigido proporcional ponderado, dirigido ponderado uniforme, y dirigido proporcional ponderado. Una combinación de algoritmos para generar estos cuatro tipos de gráficos se presenta en el algoritmo 1 (véase Figura 1). El algoritmo toma la secuencia de tokens S, tamaño de la ventana s, y dos parámetros adicionales G.type y W.mode que especifican el tipo de grafo y el método de ponderación respectivamente.

Figura 1. Grafo generador de tokens



Algorithm 1 Generate graph of tokens

Input: String of tokens S, window size s, $G.type$, $W.mode$
Output: Adjacency Matrix **A**

```
1: T[ ] ← SPLIT(S, space)
2: N ← COUNT(T)
3: V ← SORT(UNIQUE(T))
4: n ← |V|
5: for i = 1 to n do
6:     for j = 1 to n do
7:         A[t_i, t_j] ← 0
8:     end for
9: end for
10: for i = 1 to N do
11:     if i + s ≤ N then
12:         p ← i + s
13:     else
14:         p ← N
15:     end if
16:     for j = i + 1 to p do
17:         if W.mode is Proportional then
18:             A[t_i, t_j] ← A[t_i, t_j] + i + s − j
19:         else
20:             A[t_i, t_j] ← A[t_i, t_j] + 1
21:         end if
22:         if G.type is Undirected then
23:             A[t_j, t_i] ← A[t_i, t_j]
24:         end if
25:     end for
26: end for
27: return A
```

Fuente. KAR, Debabrata. PANIGRAHI, Suvasini. SUNDARARAJAN, Srikanth. SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM. 2016.p.219

La cadena de tokens se divide en una matriz $T = (t_{1,}\, t_{2,} \ldots, t_n)$, para obtener el conjunto de n vértices V. Los elementos de la matriz de adyacencia $\mathrm{A}[t_i, t_j], i, j = 1 \ldots n$, se inicializan a cero. La ventana corredora abarcando s tokens se mueve de izquierda a derecha por un testigo a la vez hasta que haya al menos dos tokens

izquierda, es decir, al menos una arista es posible. Las líneas 11 a la 15 (véase Figura 1), impiden que el puntero de token p se mueva más allá del final de la cadena. En cada posición de la ventana deslizante, las aristas son consideradas entre las fichas que se producen dentro de él, y el peso según la ponderación especificada, se añade al elemento correspondiente de la matriz de adyacencia. Para grafos no dirigidos, la matriz se hace diagonalmente simétrica copiando $A[t_i, t_j]$ a $A[t_j, t_i]$. Finalmente, el algoritmo retorna la matriz de adyacencia.

- **MEJORES PRÁCTICAS DE CÓDIGO CONTRA LA INYECCIÓN SQL**

Prácticas manuales de codificación defensiva. Consultas parametrizadas o procedimientos almacenados.

Una consulta parametrizada es un tipo de consulta que tiene algunos marcadores. En estas consultas en lugar de realizar consultas dinámicas al concatenar los parámetros con la instrucción SQL, se reemplaza los marcadores de posición con el valor de los parámetros de tiempo de ejecución. El uso de procedimientos almacenados también puede ser eficaz en la lucha contra la inyección de SQL, porque ellos comprueban el tipo de parámetros, si el atacante pasa un tipo de valor incorrecto al procedimiento almacenado, habrá una excepción.

De hecho, los procedimientos almacenados no pueden eliminar la inyección de SQL, pero ocultan la estructura de la base de datos del atacante.

```
MySqlConnection conn = new
MySqlConnection (connectionString);
MySqlCommand  Command  =
Conn.CreateCommand ( );
MySqlParameter Parameter = new
MySqlParameter ("? Id", id);
Command.Parameters.Add (Parameter);
Command.CommandText = "SELECT * FROM news where id = ?id";
```

Escapar.

Se refiere al escape de las palabras clave del lenguaje SQL. Primero, refiriendo al manual de DBMS una lista negra donde se crean palabras clave peligrosas y posteriormente se aplican a la lista negra. Cada base de datos tiene su propio escape, funciones y bibliotecas. Por ejemplo, en PHP Mysql_real_escape_string () controlará el escape proceso.

El siguiente ejemplo es un script PHP que muestra la función de escape. Como resultado de la llamada la función de escape escapará de todos los caracteres como con \ char[6].

$ Att = "This is Amir's laptop";
$ Escaped_att = mysql_real_escape_string ($Att);
Printf ("Escapedstring:", $ escaped_att);
OUTPUT: Escaped string: This is Amir \'s
Laptop

Validación de tipo de datos.

En este método, el desarrollador debe comprobar los datos que vienen de los campos del formulario. Por ejemplo, si el campo es un número de teléfono, debe comprobar que no contiene una cadena. Este método no puede garantizar que la inyección de SQL no se lleve a cabo, pero hace el proceso más difícil para el atacante.

Filtrado de listas blancas.

Es lo contrario del filtrado de la lista negra. Sólo permite las entradas que se ven legítimas para ser aceptadas y ejecutadas en la base de datos. El punto negativo sobre esta técnica es que son difíciles de implementar porque la entrada podría ser normalizada antes de someterse a la detección del algoritmo.

SQL DOM.

SQL DOM es una técnica propuesta por McClure R.A y I.H Kruger para dar la capacidad de usar declaraciones dinámicas sin problemas de seguridad. Consiste en la aplicación de un archivo principal llamado "sqldomgen". En la primera etapa, el desarrollador necesita ejecutar este archivo, para generar un archivo DLL. Los archivos DLL de salida contienen clases poderosas que pueden ser desarrolladas para hacer consultas dinámicas con ellos.

Todos los datos de campo tipos y estructura de base de datos estarán disponibles en el archivo DLL. En caso de que posteriormente se desarrollara la base de datos o cualquier cambio en la estructura de la base de datos, "sqldomgen" debería ejecutarse en la base de datos para proporcionar las nuevas clases de desarrollo[1].

[1] McClure R.A y I.H Kruger. SQL DOM: compile time checking of dynamic SQL statements. in Software Engineering, 2005, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

Inserción de consulta parametrizada.

Esta solución puede encontrar las consultas vulnerables SQL dentro del código fuente y los reemplaza con parámetros seguros SQL. El inconveniente de esta técnica es que sólo funciona con estructuras SQL creadas con cadenas explícitas. En el siguiente ejemplo, la primera línea muestra la declaración dinámica vulnerable y la segunda línea hacia delante de la variable limitada reemplazada por esta solución.

```
$ Result = mysql_query ("SELECT title FROM news
WHERE nid = '$ id');

$ Conn = new PDO ("mysql: host = localhost;
Dbname = MyDB; "," DBusername "," DBpassword ");
$ UserInput [ ] = Array ();
$ Query = "SELECT title FROM news WHERE nid =? ";
$ UserInput [ ] = $ id;
$ Stmt = $ conn-> prepare ($ query);
$ J = 1;
Foreach ($ UserInput como $ item) {
$ Statement-> bindParam ($ j ++, $ item);
}
$ Result = $ statement-> execute ();
```

SQLUnitGen.

SQLUnitGen es la forma abreviada de "Pruebas de inyección SQL utilizando el análisis estático y dinámico ", que proponen SHIN, Y., WILLIAMS L. y XIE T. La solución utiliza el análisis estático para el seguimiento del flujo de entrada del usuario para las pruebas de ataque. El núcleo de sus herramientas se basa en "JCracher". Hacen algunos cambios para crear los casos de prueba para el ataque[2].

Music.

MUSIC es la forma abreviada de "Mutación basada en verificación de vulnerabilidades de inyección SQL" que proponen SHAHRIAR, H. y ZULKERNINE, M. Utiliza el método de pruebas basadas en mutaciones para pruebas de vulnerabilidad de inyección de SQL. "La mutación es un error de base, método de prueba que se inyecta sintaxis para ver si cualquier mutante existe. Luego, comparando la salida, se puede determinar si la declaración contiene un mutante

[2] SHIN, Y., WILLIAMS L., y XIE T. “Sqlunitgen: Sql injection testing using static and dynamic analysis, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

". Su método utiliza nueve operadores de mutación para hacer la inyección en el código[3].

Vulnerabilidad e inyección de ataque.

Esta solución es propuesta por FONSECA, J., VIEIRA, M. y MADEIRA, H. [6 Presentan un método para atacar la aplicación mediante vulnerabilidades de inyección de SQL. Esto prepara un ambiente para probar la contramedida de herramientas de seguridad tales como sistemas de detección de intrusos, firewalls y escáneres de vulnerabilidad. Los resultados usaron datos recolectados de ataques y parches reales. El análisis de esta información ayuda a comprender los códigos vulnerables y sus distinciones que no son vulnerables].

La herramienta propuesta se compone de dos partes principales: "Herramienta de Inyección de Ataque" y "Herramienta de Inyección de Vulnerabilidad", que trabajan automáticamente juntas. El programa de inyección de Vulnerabilidad se utiliza para vulnerabilidades de inyección SQL en el código de la aplicación. Revisa el código fuente y busca los posibles lugares que son adecuados para la inyección.

La herramienta de inyección de ataque es una aplicación que funciona casi de la misma manera que la parte anterior. La diferencia es que después de la inyección de la vulnerabilidad, se intenta atacar la aplicación. Por la ayuda de un Proxy HTTP esta herramienta rastrea el tráfico entre la aplicación Web y el sistema de gestión de bases de datos. Estos datos se utilizan para lanzar el ataque contra archivos vulnerables.

Utilizando la herramienta "Detector de éxito de ataque", se verifica que el ataque se lanza con éxito. La herramienta de inyección de ataque tiene una debilidad que el error es un valor procesado antes de usar en el SQL. Por ejemplo, el número de teléfono del usuario puede dividirse en una sección de código de país y número de teléfono y se utiliza contra dos columnas de la base de datos. Esto lleva a este algoritmo a cometer un error en la detección[4].

SUSHI.

SUSHI es un "solucionador de restricciones de cadenas" propuesto por Fu X. y C. Li. Proponen un algoritmo recursivo que puede resolver una ecuación lineal simple

---

[3] SHAHRIAR, H. y ZULKERNINE, M. “MUSIC: Mutation-based SQL Injection Vulnerability Checking, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

[4] FONSECA, J., VIEIRA, M. y MADEIRA, H. Vulnerability & attack injection for web applications, citado por citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

(SISE) de una manera eficiente. Rompen la restricción SISE con operaciones atómicas de cadenas. Demostraron que su solución es eficaz para encontrar complicados ataques de inyección de SQL[5].

Ardilla.

Ardilla es un método para crear ataques de inyección de SQL propuesta por Kieyzun A. y sus colegas. Esta herramienta es capaz de generar ataques para usar como entrada la aplicación Web, para detectar las vulnerabilidades de inyección de SQL. Consideran que su solución no tiene sobrecarga en tiempo de ejecución ni tampoco es necesaria la modificación del código fuente. Ardilla genera algunas entradas y ejecuta la aplicación con entrada para la prueba. Posteriormente recibe la salida y hace el análisis para ver si hay datos enviando a la base de datos entre el tiempo de recepción de la entrada hasta la ejecución de la consulta[6].

Analizador de cadenas.

Wassermann G. y Z. Su, proponen un analizador de cadenas que es un algoritmo basado en la gramática que modela la cadena como gramática libre de contexto y operaciones de cadenas como el idioma de transductores después de Minamide. Esta solución etiqueta las cadenas que vienen del lado del usuario como No terminal, se asigna la "etiqueta directa" a esas cadenas que vienen directamente desde el lado del usuario como peticiones GET. Se le asigna "etiqueta indirecta" a las cadenas que vienen desde el lado de la base de datos. Luego, resumen las cadenas marcadas para encontrar los contextos y después usar lenguajes regulares y lenguajes libres de contexto que comprueban la seguridad de cada cadena en el aspecto de la sintaxis[7].

PHPMiner.

PHP Miner es una herramienta propuesta por Khin Shar y Kuan Tan, que mira los atributos de código estático y luego hace un modelo de predicción de vulnerabilidad basado en los datos recogidos en la fase anterior.

---

[5] FU, X. y LI, C.-C. A string constraint solver for detecting web application vulnerability, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

[6] KIEYZUN, A. Automatic creation of SQL Injection and crossite scripting attacks. citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

[7] WASSERMANN, G. y SU, Z. Sound and precise analysis of web applications for injection vulnerabilities, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

- **PREVENCIÓN EN TIEMPO DE EJECUCIÓN DE ATAQUES DE INYECCIÓN SQL**

SQLrand.

Boyd S.W y A.D Keromytis propusieron una técnica para SQL para la prevención de inyección que utiliza la consulta al azar SQL para detectar declaraciones maliciosas y abortarlas. Para este propósito realizan instancias aleatorizadas de la consulta SQL, para aleatorizar la consulta de plantilla dentro del script CGI y la base de datos parse. Por ejemplo, la palabra clave SELECT podría ser reemplazada por SELECT921 que es un nombre aleatorio que genera la ejecución actual. Después, se desarrolla mediante un proxy, una intercepción de tráfico entre la aplicación y la base de datos, y si cualquier palabra clave sin aleatorización encuentra que es una inyección de SQL. El atacante no puede hacer la inyección de SQL sin saber la tecla aleatoria. El punto positivo de esta solución es que no afecta el rendimiento.

AMNESIA.

Halfond W.G.J y A. Orso proponen el modelo AMNESIA que es la combinación de análisis estático y análisis de tiempo de ejecución de comportamiento de la aplicación. AMNESIA crea un modelo de consultas originales en la parte estática. Después en la parte dinámica supervisa "consultas generadas dinámicamente" en tiempo de ejecución y compara con el modelo legítimo. Las declaraciones SQL que no pueden cumplir los requisitos del ataque de inyección de SQL entonces la herramienta se detendrá antes de enviar a la base de datos. La debilidad principal de esta técnica es que no pueden soportar consultas segmentadas[8]

WASP.

Halfond W.G.J y sus colegas proponen WASP, Aplicaciones Web con tinción positiva y sintaxis de Evaluación, que funciona basándose en la dinámica contaminación. Su enfoque es la versión mejorada de tintura clásica. La primera mejora es el uso de contaminación positiva que se basa en hacer la confianza. La contaminación se basa en datos no confiables. La siguiente mejora es la exactitud y eficiencia de la contaminación mediante el rastreo de confianza haciendo a nivel

[8] HALFOND, W.G.J. y ORSO, A. AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

de personaje. La tercera mejora es el bloqueo de consultas que contienen palabras clave y operadores SQL. Sin hacer la confianza. El último es el mínimo requisito de aplicación que hace que este método sea práctico[9].

SQLprob.

SQLprob es la forma corta del bloqueador basado en Proxy SQL,sistema de detección de inyección SQL propuesto por Liu Anyi y colegas. Este enfoque extrae la entrada del usuario de la consulta generada por la aplicación Web. Luego valida estos datos en contexto de la estructura sintáctica de la consulta generada. Para ello utiliza un algoritmo genético. Este sistema tiene pocas ventajas. La primera es que, no requiere código fuente de la aplicación. La siguiente es que la validación del proceso no necesita aprendizaje. La tercera mejora es que esta técnica utiliza un proxy que necesita el mínimo requisito para la implementación. El último es la independencia del lenguaje de programación[10].

CANDID.

CANDID es un soporte para (Evaluación de candidatos para descubrir dinámicamente la intención) que propone Bisht P., P. Madhusudan y V.N. Venkatakrishnan. Esta técnica minera dinámicamente la estructura de consulta prevista del programador en el tiempo de ejecución con entradas válidas. Luego la compara con la declaración de consulta legítima. Si el resultado no es el mismo, es un ataque de inyección de SQL[11].

Minimización de los privilegios.

Estos aspectos de seguridad deben ser priorizados y las medidas adecuadas deben ser tomadas durante la etapa de desarrollo en lugar de dejar esos asuntos al final del ciclo de desarrollo. Los desarrolladores se deben familiarizar con el marco de seguridad para que resulte un producto seguro. Es conveniente crear primero una cuenta con pocos privilegios y luego dependiendo de las necesidades de permisos se puede añadir aún más.

---

[9] HALFOND, W.G.J., ORSO, A. y MANOLIOS, P. "WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

[10] LIU, Anyi, YUAN, Yi, WIJESEKERA, Duminda y STAVROU, Angelos. SQLProb: a proxy-based architecture towards preventing SQL injection attacks, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

[11] BISHT, P., Madhusudan, P. y VENKATAKRISHNAN, V.N., "CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection Attacks, citado por ZAMANI Mazdak, MANAF Azizah Abd., SADEGHIAN Amirmohammad. A Taxonomy of SQL Injection Detection and Prevention Techniques.2013.

Aplicación de las normas de codificación uniformes.

Se debe establecer una infraestructura bien planeada con políticas y normas de seguridad. Los desarrolladores por lo general utilizan cualquier método para acceder a los datos resultantes, que es una preocupación de seguridad grave. Por lo tanto, tiene que haber ciertas políticas, reglas según las cuales los desarrolladores pueden acceder a los datos, de modo que hay una cierta similitud entre la rutina de los desarrolladores. En segundo lugar, las políticas de codificación también deben realizar en el servidor mejores prácticas tanto para la seguridad, como para la facilidad de mantenimiento.

Cortafuego o firewall de seguridad del servidor SQL.

El servidor SQL debe contar con un sistema de acceso no autorizado conocido como cortafuegos para que sólo los clientes de confianza pueden ser contactados. Los cortafuegos necesitan estar conectados con el servidor SQL, la red administrativa y el servidor Web. Este firewall debe rechazar todas las entradas que contengan datos binarios, las secuencias de escape, y comentar caracteres. Las múltiples capas de validación deben ser implantados y la entrada de usuario que no se valida no deben ser concatenados[7].

Los requisitos de seguridad están comúnmente protegidos por un WAF (Firewall de aplicaciones Web). En la arquitectura general del sistema, se coloca un WAF delate de la aplicación Web que debe proteger. Cada solicitud que se envía a la aplicación es examinada por el WAF, antes de que llegue a la aplicación Web, solo se hará la entrega si la solicitud cumple con el conjunto de reglas del cortafuego.

Un enfoque común para definir el conjunto de reglas del cortafuego está dado por la lista negra. Una lista negra contiene patrones de cadena, normalmente definidas como expresiones regulares. Las solicitudes reconocidas, son probable que los patrones sean ataques maliciosos, por ejemplo, SQLIA y, por lo tanto, están bloqueados. Por ejemplo, las siguientes expresiones describen la sintaxis de los comentarios SQL, como / ** / o #, y existen otras expresiones que se utilizan con frecuencia en ataques SQLIA, como son las siguientes:

/*!?|*/|[';]--|--[\s\r\n\v\f]|(?:--[^-]*?-) | ([^\-&])#.*?[\s\r\n\v\f]|;?\\x00

Hay varias razones por las que un WAF puede proporcionar protección, incluyendo configuración errónea. Una forma de asegurar la resiliencia de un WAF contra ataques es basarse en un procedimiento de prueba automatizado que detecta exhaustiva y eficientemente las vulnerabilidades. Este papel aborda este desafío para las inyecciones SQL, uno de los tipos de vulnerabilidades en la práctica[8].

- **MODELO DE ÁRBOL DE DECISIONES CONTRA SQLIA**

Un enfoque práctico para manejar el problema del ataque de inyección SQL es escanear cada consulta que viene o se ingresa en la base de datos de esa manera se obtienen las vulnerabilidades presentes. Un modelo de trabajo que permita ejecutar lo anterior es el uso del modelo de árbol de decisiones en paralelo con un modelo estadístico o algoritmo de clasificación para obtener reglas comunes en diferentes ataques de inyección SQL y usar estas reglas para la prevención de futuros ataques.

El modelo propuesto utiliza la técnica de análisis para encontrar el ataque de inyección de SQL que hace uso del árbol de decisión SQL. Una vez creado el recurso de ataque de tal manera que, si las páginas Web no se desinfectan, la URL de la consulta siempre deberá proporcionar problemas de base de datos.

Después de poner la solicitud de ataque la herramienta comprueba la respuesta, si existe cualquier base de datos con problemas entonces puede informar al usuario final. Luego, el usuario final podrá acceder a la base de datos sin problema.

Este ataque de base de datos se establece con las reglas de clasificación que puede ser actualizado por el administrador si es necesario. Si se encuentra algún ataque de base de datos en el servidor Web de la respuesta, entonces se puede decir que existe vulnerabilidad en la consulta de base de datos dada. También se desinfecta la entrada individual.

Según la arquitectura la máquina cliente puede ser n-número y cualquier cliente puede acceder al servidor Web y obtener los servicios. A fin de obtener ese servicio que va a utilizar la solicitud URL mediante métodos get y post. Después de que el usuario publique la consulta si la información es de la base de datos el servidor Web hace uso del servidor de base de datos en el que se puede realizar la transformación de la consulta. Los ingenieros de software confían principalmente en la creación de consultas dinámicas con concatenación de cadena para construir sentencias SQL[9].

En realidad, en el tiempo de ejecución, el sistema forma consultas con insumos recibidos directamente de fuentes externas. La propuesta de este método es permitir diseñar diferentes consultas basadas en condiciones variables establecidas por los usuarios. Sin embargo, como esta es la causa de muchas aplicaciones Web, algunos desarrolladores optan por utilizar consultas parametrizadas o procedimientos almacenados. Mientras que los métodos son seguros, su uso inapropiado puede hacer código vulnerable.

En código HTML hay falta de controles para ataques de SQL que hacen que el sistema sea más atractivo para los ataques. El error más común y grave que los desarrolladores hacen es utilizar entradas en sentencias SQL sin ninguna comprobación. La mayoría de las prácticas manuales de codificación defensiva son la mejor manera para derrotar la inyección de SQL, tal aplicación es necesario más interacciones manuales.

Para disminuir estos problemas, el método propone un sistema de automatización de validación de entrada que es la erradicación de cualquier tipo de interacción del usuario con el sistema de validación de formularios de entrada. Los ingenieros pueden optar por proporcionar su propio esquema de base de datos y construir sentencias SQL utilizando sus API. El Modelo es especialmente útil cuando los desarrolladores necesitan usar consultas dinámicas en lugar de consultas parametrizadas para la flexibilidad. En la mayoría de los casos sólo pueden utilizarlo con un nuevo software, proyectos, y deben aprender un nuevo desarrollo de consultas.

En el modelo propuesto, la solicitud de URL de cliente al servidor es filtrado usando el árbol de decisión que clasifica la consulta SQL como clase de ataque y clase de no ataque basada en información anterior. De acuerdo con el modelo, primero los datos en relación con el ataque de inyección de SQL se adquieren y usando el algoritmo de clasificación de árbol de decisión para la clasificación de inyección de SQL (Algo.A) las reglas de clase para clasificar las consultas SQL como ataques y no-ataques es construido.

Cuando una solicitud llega al servidor Web, el servidor envía la solicitud de ataque de inyección SQL al algoritmo de detección (Algo.B) que utiliza el analizador de árbol de decisión (Algo.C). Para encontrar si la solicitud es más segura o no si la solicitud es seguro entonces lo enviará a la base de datos son enviar mensaje de error al usuario[9].

Algoritmo para la construcción del árbol de decisión de inyección SQL para base de datos.

Paso 1: Comprobar la base de datos de inyección de SQL
Paso 2: Para cada consulta SQL
Paso 3: Encontrar la relación de ganancia de información
Paso 4: Basado en la ganancia de información
Paso 5: Crear un nodo de decisión que se divide con información ganancia
Paso 6: Basado en nodos, construye el árbol para clasificar SQL como ataques y no ataques.

Algoritmo para el procedimiento de detección de ataques de inyección de SQL.

Paso 1: Crear una Estructura de Datos de Cola para almacenar solicitud de URL
Paso 2: Decidir si la URL dada debe acceder a la base de datos o no
Paso 3: Si la URL es para la consulta de destino, envíela al analizado de árbol de decisión y registre la respuesta.
Paso 4: Si la respuesta es positiva directamente a la base de datos.
Paso 5: El otro mensaje de error de informe al usuario.
Paso 6: Finalizar

Algoritmo Analizador de árboles de decisión.

Paso 1: Obtener la solicitud de SQL desde el servidor Web
Paso 2: Vaya al árbol de decisión de inyección de SQL y busque clase de consulta SQL
Paso 3: Si la clase resultante es una clase de ataque, dé resultado negativo
Paso 4: Sino dar la respuesta positiva [9].

ALGORITMO PARA PREVENIR LA BASE DE DATOS BACK-END EN CONTRA DE ATAQUES DE INYECCIÓN SQL

El enfoque se lleva a cabo en tres fases, en la primera fase se discute qué datos deben ser almacenados en la base de datos y la forma en que se deben almacenar. En la segunda fase se discute cómo los datos deben consultarse en la base de datos y en la tercera fase qué datos deben ser recuperados. De esta manera, estas fases son:

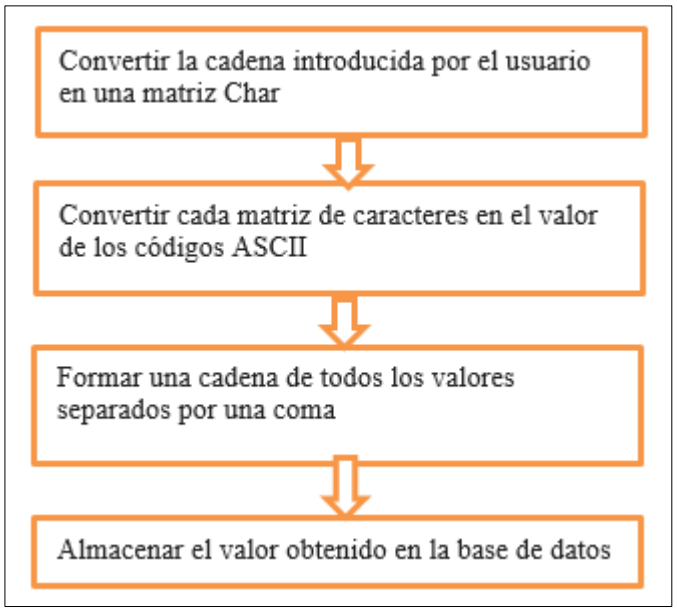
La introducción de datos en la base de datos.

La información introducida por el usuario será una cadena que se convierte en valores ASCII, el cual se almacena en la base de datos. Este es el aalgoritmo para almacenar valores.

Paso 1: Declarar una matriz int, n [];
Paso 2: Almacenar el valor introducido por el usuario en una cadena, str;
Paso 3: Convertir la cadena en una matriz de tipo char ch [];
Paso 4: Almacenar todos los valores ASCII de la matriz en ch, n de la matriz y los respectivos índices;
Paso 5: Formar una cadena utilizando los valores de la matriz de n, cada valor debe ser separado utilizando cualquier símbolo. Se utiliza (,) como el símbolo especial.
Paso 6: Ahora se almacena la actual secuencia obtenida en la base de datos.

A continuación, se detalla la arquitectura para el almacenamiento de valores en la base de datos (véase Figura 2).

Figura 2. Arquitectura para el almacenamiento de valores en la base de datos



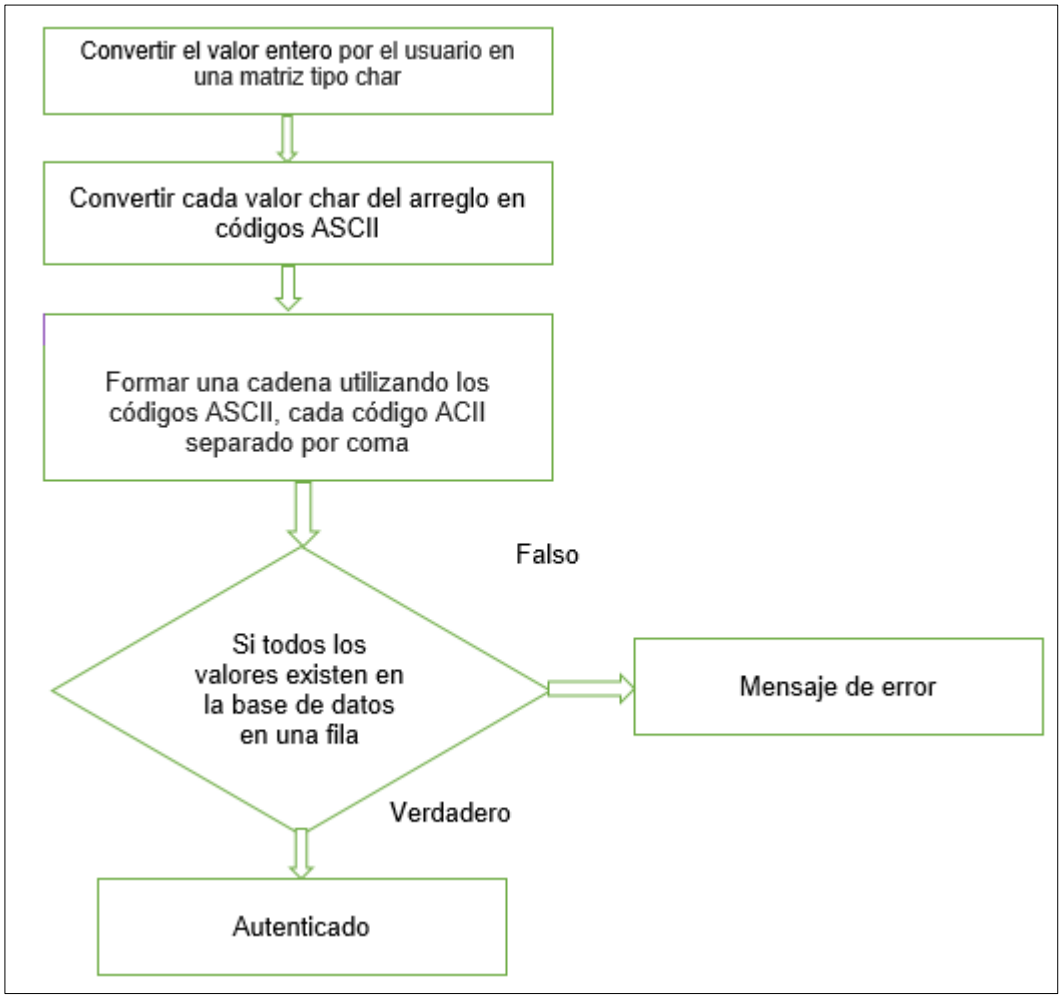
Fuente. SRIVASTAVA Mahima. Algorithm to Prevent Back End Database against SQL Injection Attacks.2014. p. 756

Autenticación.

La autenticación debe coincidir con el valor introducido por el usuario y los valores existentes en la base de datos. Los valores introducidos serán los valores formados por los valores ASCII, como modo de autenticación estos valores son enviados a través de la capa de aplicación. Este es el algoritmo para la autenticación (véase Figura 3)[10].

Paso 1: Almacenar el valor introducido por el usuario en el campo de entrada en una cadena es decir s;
Paso 2: Convertir la cadena s en una matriz de tipo char, s1 [ ];
Paso 3: Convertir cada valor de la matriz s1 en códigos ASCII y almacenarlos en un int en la matriz, n1 [ ];
Paso 4: Formar una cadena utilizando los valores de la matriz n1. Cada valor debe estar separado mediante un carácter especial. Aquí se está usando coma (,).
Paso 5: Si este valor existe en la base de datos, entonces la autenticación se visualiza.

Figura 3. Algoritmo por autenticación



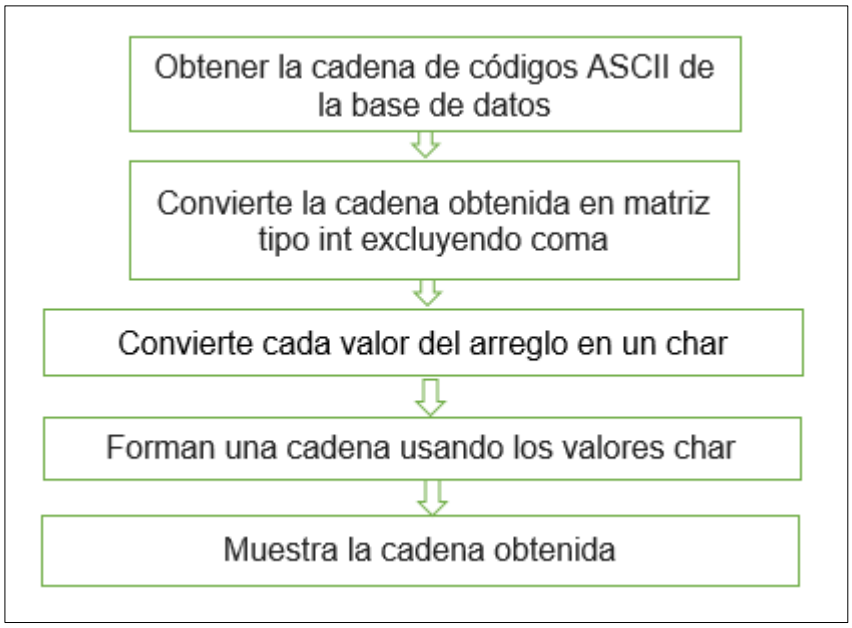
Fuente. SRIVASTAVA Mahima. Algorithm to Prevent Back End Database against SQL Injection Attacks.2014. p. 757

Recuperación de la información.

Algoritmo para recuperar valores de la base de datos (véase Figura 4).

Paso 1: Almacenar la cadena obtenida de la base de datos en una variable, ejemplo a;
Paso 2: Convertir esta cadena en matriz de tipo int excluyendo coma (,), la coma se utiliza para dividir la cuerda.
Paso 3: Convertir los valores ASCII almacenados en el tipo int en la matriz de tipo char.
Paso 4: Ahora forma una cadena usando estos valores char.
Paso 5: Muestra la cadena obtenida[10].

Figura 4. Algoritmo para recuperar los valores de la base de datos



Fuente. SRIVASTAVA Mahima. Algorithm to Prevent Back End Database against SQL Injection Attacks.2014. p. 757

- **ANALISIS DE CAUSA RAÍZ DE GESTIÓN DE SESIÓN Y AUTENTICACIÓN ROTA**

El analizador de RCA (Análisis de causas raíz) de vulnerabilidades de gestión de sesiones se realiza en dos categorías: secuestro de sesión y fijación de sesión. Mientras que el analizador de RCA de vulnerabilidades de autenticación roto sigue un problema de raíz de línea de causas de la gestión de sesiones de vulnerabilidades y soluciones sugeridas.

Razones que generan el secuestro de sesión:

Uso de ID de sesión adivinatoria.
Ausencia de mecanismo de detección para "ensayo de repetición de adivinanzas" ya sea con métodos de fuerza bruta o sistemáticos.
No se pueden detectar ensayos de adivinanzas repetidas mientras hay un mecanismo en su lugar.
Criptografía débil: Una debilidad en el algoritmo de criptografía o una debilidad en la forma en que se utiliza un algoritmo criptográfico fuerte.
Limitación de HTTP: La ausencia de cualquier mecanismo inherente o integrado de gestión de estado.

Métodos de manejo de sesión inseguros.
Uso indebido de la solución: La mala configuración o el uso inadecuado de soluciones básicamente fuertes.
Debilidad en la técnica de gestión de sesiones inactivas.

Causas principales de la fijación de sesión:

Servidor Permisible: Un servidor que acepta ID de sesión generados por el cliente
Tipo de gestión de sesión en uso.
Reutilización de los identificadores de sesión: Generar identificadores idénticos de sesión dos veces o más para diferentes sesiones de clientes iguales o diferentes.

La gestión de sesiones se ha convertido en algo importante para las interacciones de los usuarios los sitios Web cruzados. Sin embargo, todavía dependen del protocolo sin estado HTTP, que carece de una fuerte capacidad de manejo de sesión para inducir el uso de varias soluciones que pueden comprometer aún más la seguridad de los sistemas. Para reforzar la seguridad se sugiere el uso de cookies, políticas y el uso de SSL para cualquier tráfico que incluya ID de sesión y credenciales[11].

Las causas de raíz "Uso de ID de sesión" y "Reutilización de ID de sesión" muestran la falta de comprensión clara de la solución implementada por los programadores. Otra causa estrechamente relacionada que muestra la falta de comprensión son los "Permisos del servidor" para la sesión generada por el cliente. Para tratar estas causas, se sugiere el despliegue de generadores de números aleatorios criptográficamente fuertes para generar ID de sesión.

La falta de atención a los detalles es también una característica común entre las causas, "Métodos inseguros de manejo de sesiones" ejemplificados por la ocultación de la sesión y la "debilidad de gestión de sesiones inactivas". Un problema que aún necesita más investigación es la falta de mecanismo de identificación repetida y confiable.

Actualmente, en todas las redes IP, se utiliza la identificación de origen, sin embargo, con la posibilidad de suplantación de identidad IP, los atacantes pueden intentar utilizar ID de sesión de fuerza bruta mediante ensayos repetidos[11].

Causas de raíz de vulnerabilidades de autenticación rota y soluciones sugeridas:

Falta de métricas: Ausencia de métricas bien desarrolladas que puedan ayudar a tomar la decisión correcta en la selección de los mecanismos de seguridad.
Falta de conocimiento de seguridad entre los programadores para aplicar los mecanismos de seguridad de la información y la comunicación a sus soluciones.

Decisiones o compromisos incorrectos: Tanto los diseñadores como los programadores son propensos a tomar decisiones erróneas debido a la falta de métricas y conocimientos de seguridad.
Uso de módulos auto desarrollados en lugar de módulos probados y bien analizados para servicios de seguridad como autenticación.
Almacenamiento de las credenciales de usuario con otros datos de la aplicación.
Adivinar los intentos: Permite repetidos intentos de adivinar.
Nivel de datos del usuario en el sistema: el nivel de información que el sistema conoce / sostiene sobre los usuarios.
Falta de conciencia de seguridad entre los usuarios.
Los rigurosos requisitos establecidos para fortalecer la seguridad pueden ser poco realistas y difíciles de satisfacer por los usuarios.

El diseño y la implementación de los módulos de autenticación tendrán en cuenta tanto las cuestiones técnicas como las humanas. La implementación de un sistema de autenticación con características de seguridad muy fuertes puede resultar prohibido e intransferible e impedir la usabilidad. El nivel de seguridad requerido para un sistema particular apenas se conoce como la disciplina de seguridad carece de métricas apropiadas.

Lo que en realidad implica que es difícil medir si una solución concreta ha alcanzado o no el nivel de seguridad deseado. Cuando se consideran mecanismos de autenticación basados en contraseña, la capacidad de los usuarios humanos para recordar contraseñas largas y complejas es una limitación bien conocida. A menudo, es posible persuadir a los usuarios a evitar el uso de contraseñas débiles.

Sin embargo, debido al aumento en el número de sistemas disponibles para un solo usuario que pueden iniciar sesión, los usuarios pueden preferir reutilizar contraseñas o utilizar una contraseña bastante memorable que fácilmente se puede adivinar. Cualquier forma de intentar satisfacer los requisitos de contraseña hace que los sistemas sean vulnerables.

Preservar las sesiones de los usuarios autenticados y la gestión a largo plazo de las credenciales de los usuarios son cuestiones importantes. Se debe prestar la debida atención al estudiar y entender cómo el sistema de autenticación maneja numerosas solicitudes cada vez que se ejecuta la aplicación.

Las soluciones puramente técnicas para la seguridad de las aplicaciones Web a partir de problemas de autenticación rotos son el uso de S-HTTP durante la autenticación y la protección criptográfica de las credenciales de usuario (por ejemplo, utilizando hash o cifrado). Estas soluciones protegen los datos de autenticación, pero la solución crea una sobrecarga de comunicación y necesita una mayor optimización[11].

- **MECANISMO DE AUTENTICACIÓN ROBUSTA PARA LA GESTIÓN DE SESIÓN**

Este mecanismo alternativo se llama OTC (one time cookies) cookies una vez, el cual es utilizado para la sesión de autenticación.

Modelo de amenaza.

El objetivo del adversario es tomar el control de las sesiones ya establecidas por los usuarios de una aplicación Web. Hay dos tipos de adversarios. Pasivo y activo. Un adversario pasivo tiene acceso a toda la información intercambiada entre el navegador y la aplicación Web. El adversario pasivo intentará fabricar o reutilizar fichas de autenticación para secuestrar la sesión de un usuario. Un adversario activo tiene el mismo acceso a la información como el pasivo, pero, además, este adversario puede modificar activamente las peticiones, y las respuestas intercambiadas entre el navegador y la aplicación Web.

El adversario activo puede modificar, crear y evitar que los mensajes lleguen a su destino. Además, un adversario activo puede ejecutar ataques a nivel de aplicación contra el navegador y la aplicación Web, incluidos los scripts entre sitios (XSS), y ataques de fijación de sesión. Un adversario activo también puede probar ataques phishing para robar tokens o el archivo de almacenamiento de la computadora del usuario.

OTC se basa en HTTPS para proteger la configuración de sus credenciales durante el inicio de sesión de un usuario. Por lo tanto, OTC supone que HTTPS se establece correctamente y de una manera segura.

Propiedades del protocolo deseado.

Estas son las propiedades para diseñar OTC:

Integridad de la sesión: El mecanismo propuesto debería proporcionar una sólida sesión de autenticación y debería ser intrínsecamente seguro contra el secuestro de sesión, es decir no deberían ser necesarios mecanismos adicionales de protección.
Apaciguamiento: El mecanismo propuesto no debería requerir un estado adicional en la aplicación Web para la verificación de solicitudes.
Robustez: El mecanismo propuesto debería generar tokens de autenticación con fuertes garantías de confidencialidad e integridad.
Performance y escalabilidad: El mecanismo propuesto debería ser tan eficiente y escalable como cookies de autenticación. Rendimiento y escalabilidad de la aplicación Web no debe verse afectada.

Almacenamiento seguro: El mecanismo propuesto debería almacenar las credenciales de autenticación con seguridad en el navegador. En particular, las credenciales de autenticación deben ser aisladas desde otros componentes y funcionalidades del navegador. Por ejemplo, las credenciales deben tener protecciones similares a las de contraseñas y claves privadas[12].
Implementación: El mecanismo propuesto debe requerir cambios mínimos en el Navegador y la aplicación Web. Ningún hardware o software adicional debe ser necesario.
Usabilidad: El mecanismo propuesto debe proporcionar una experiencia de usuario similar a las cookies.
Concurrencia: El mecanismo propuesto debe funcionar con aplicaciones Web que tiene alta concurrencia de solicitudes. Por tanto, los testigos de autenticación deben ser independientes
Asistencia al navegador: El mecanismo propuesto debe aplicarse como parte del Navegador (componente principal o extensión) para proporcionar seguridad y funcionalidades adecuadas.

Descripción del protocolo.

OTC crea un token único por solicitud. Cada token está vinculado a una petición en particular utilizando un secreto de sesión. Por lo tanto, un token no puede ser reutilizado para diferentes solicitudes. Además, OTC almacena la información de estado necesaria para validar el token. Cada ticket está encriptado con una clave a largo plazo compartida entre todos los servidores de aplicaciones Web. Los servidores de las aplicaciones Web pueden acceder a la información almacenada en el ticket. El usuario nunca tiene acceso al contenido. Se definen credenciales como los valores almacenados en el navegador y fichas como los valores adjuntos a cada solicitud[12].

ANALISIS DE SEGURIDAD OTC.

Con OTC, el navegador firma cada solicitud con el secreto de sesión ks, en lugar de adjuntar a las solicitudes como lo hacen las cookies. Por lo tanto, el navegador nunca envía la sesión de secreto a través de la red, reduciendo su exposición. El secreto de la sesión sólo se transmite a través de la red cuando la aplicación Web la envía al navegador durante el inicio de sesión de usuario (Mensaje 2 véase figura 7). Sin embargo, HTTPS se utiliza para proteger el secreto de la sesión durante este paso. Como se indicó anteriormente, OTC supone que HTTPS se establece correctamente de lo contrario un adversario también puede aprender la contraseña del usuario.

OTC se basa en una función HMAC para firmar las solicitudes de los usuarios. La inclusión de los tokens garantiza que cada valor HMAC es único, incluso para idénticas peticiones. Por lo tanto, el HMAC hace cada token OTC único y vincula

cada ficha a una solicitud particular. Como resultado, un adversario activo o pasivo no podrá reutilizar.

La Tabla 1 muestra una lista de las principales amenazas que afectan a las cookies de autenticación y se aplican a OTC. A excepción de los ataques de denegación de servicio, los ataques de red no afectan a OTC porque el navegador nunca envía el secreto de sesión OTC a través de la red. En el OTC es resistente a la mayoría de los ataques que afectan a las cookies porque las credenciales OTC son almacenados y gestionados de forma segura en el navegador de forma predeterminada[12].

Tabla 1. Principales amenazas que afectan a las cookies de autenticación

Amenazas en la Red	**Cookies**	**OTC**
Divulgación debido al uso de HTTP no cifrado	X	-
Divulgación debido a errores de configuración / errores de software	X	-
SSL fractura ataques	X	-
Ataques de renegociación SSL	X	-
Ataques de bestia SSL	X	-
Ataques denegación de servicios	X	X
Amenazas en el navegador y aplicaciones Web	**Cookies**	**OTC**
Ataque Cross-Site Scripting (XSS)	X	-
Ataque Cross-Site Tracing (XST)	X	-
Cross-Site Request Forgery (CSRF)	X	X
Ataque relacionado con el dominio	X	-
Ataque Clickjacking	X	-
Ataque fijación de sesión	X	-
Cookie clobbering protegido	X	-
Débil generación de tokens	X	-
Cookies que roban el malware	X	-
Malware que controla el navegador	X	X
Ataques de ingeniería social	X	X

Fuente. DACOSTA Italo, CHAKRADEO Saurabh, AHAMAD Mustaque, TRAYNOR Patrick. One time cookies: Preventing session hijacking attacks with stateless authentication tokens. 2012.

- **AUTENTICACIÓN CON CLAVES DE SESIONES PARA LA GESTIÓN DE MEMORIA DINÁMICA**

Mediante el uso de la gestión de claves de sesión y con la ayuda de la autenticación automática, la memoria pueda ser ocupada dinámicamente y la eficiencia del espacio puede ser obtenida.

Las claves de sesión se utilizan en la red para almacenar los datos en la memoria para la asignación dinámica de memoria. La clave de sesión desempeña el papel

principal en la autenticación de las ubicaciones de memoria cuando sea necesario. Los usuarios envían las solicitudes y se manejan continuamente en la red inalámbrica cuando se procesan los datos. Una vez el búfer de la petición está lleno, entonces la memoria está sobrecargada.

La red inalámbrica no puede manejar ninguna solicitud cuando está sobrecargada. Para evitar sobrecargas en la red, se implementa el concepto de gestión de memoria para que, cuando el almacenamiento se llene, necesite buscar dinámicamente espacio en la red con la ayuda de las claves de sesión.

Las claves de sesión son llamadas, así como las claves simétricas, que tienen la funcionalidad para proporcionar las claves para la sesión siempre que sea necesario. Las claves de sesión se utilizan cuando la memoria está sobrecargada a la red inalámbrica identificada y las claves se comparten a través del canal inalámbrico para obtener la ubicación de memoria cercana dinámicamente[13].

Existen situaciones donde se utiliza memoria en la red inalámbrica con la autenticación apropiada, las solicitudes se manejan de vez en cuando. Sólo si hay congestión en la red, peticiones en forma de datos no pueden ser manejadas y las solicitudes se descartan. Las claves privadas se comparten entre las principales dispositivo de memoria y los manejadores de solicitudes. Cuando se realiza la solicitud, los datos se protegen las claves de sesión que se cifran y luego la comunicación en el canal se establece con el intercambio de las llaves.

La Figura 5 es el pseudo-código que explica la gestión de la memoria con la ayuda de las claves de sesión. Para manejar las solicitudes incluso con la congestión, las memorias dinámicas se buscan en la red, donde los búferes están disponibles.

Figura 5. Gestión de memoria Pseudo código

Fuente. NAGARAJA Arun, SARAVANA Kumar R. A Session Key Utilization Based Approach For Memory Management in Wireless Networks. 2015

La utilización de la clave de sesión inicialmente, la conexión inalámbrica se realiza con la ayuda de la autenticación, si se pasa la autenticación, los datos son transferidos con la solicitud y el proceso se continúa. Si se ha producido un error en la autenticación, se descarta la conexión.

Las solicitudes se envían a través del enrutador y el enrutador es responsable para la recolección de datos y la transferencia de los datos. Para evitar la pérdida de datos, se puede utilizar la memoria externa para aceptar las solicitudes cuando la memoria está llena, el enrutador eenvía peticiones dinámicamente para una nueva ubicación de memoria y esto se puede obtener con las claves de sesión[13].

- **GESTIÓN SEGURA DE LA SESIÓN**

Es necesario analizar el comportamiento de una aplicación segura y cuáles son los parámetros que deben ser atendidos, mientras se resuelve la vulnerabilidad de fijación de sesión. El primer paso es de nuevo el mismo, donde el cliente inicia una

comunicación con el servidor, y el servidor responde con una cookie que se almacena en el navegador del cliente como información de identidad.

Cuando el cliente intenta acceder a las páginas autenticadas usando credenciales legítimas, el servidor seguro desvía la cookie de sesión generada anteriormente y genera una nueva cookie generada pseudo aleatoriamente, que contiene la información de autenticación del usuario legítimo / autenticado.

Detección automatizada de la vulnerabilidad de la fijación de la sesión.

El método propuesto para la detección automatizada de vulnerabilidades de fijación de sesión implica un mecanismo de análisis de cookies de sesión y las iteraciones de código fuente de la aplicación de prueba afectada. Los siguientes son los pasos involucrados en el mecanismo de detección automatizado:

Se envía la solicitud GET al formulario de inicio de sesión de la aplicación Web que se va a probar.
Webserver emitirá un identificador de sesión SID utilizando un indicador Set-Cookie invocado sólo en el caso de la primera comunicación con un cliente que se captura para un análisis posterior.
Prepara una solicitud POST con credenciales de aplicación legítimas y envíe al formulario de inicio de sesión.
A continuación, se captura el código fuente de la página autenticada.
Prepara una solicitud POST con credenciales de aplicación legítimas de ID de sesión generadas en el paso 2 y la envía al servidor de aplicaciones vulnerable, lo que da como resultado un acceso autenticado satisfactorio
El código fuente de la transacción del Paso 5 se captura utilizando un script de automatización.
Los valores de código fuente y cookie de los pasos 4 y 6 se comparan.
8) Si el código fuente y el valor de cookie de los pasos 4 y 6 coinciden, se confirma que la aplicación Web se ha iniciado correctamente utilizando un ID de sesión que se utilizó en caso de un acceso no autenticado al mismo servidor. Por lo tanto, se demuestra que la aplicación es vulnerable a la vulnerabilidad de fijación de sesión[14].

- **TÉCNICAS DEFENSIVAS DE XSS EN REDES SOCIALES EN LÍNEA**

La Tabla 2 destaca las fortalezas y limitaciones de las Técnicas defensivas existentes de ataques XSS. Este método que restringe el ataque XSS aplicando la sanitización sensible al contexto en la Web. Los componentes principales son: Clasificador de tipo de contexto y motor de contexto sensible auto sanitización (CSAS)[15].

Tabla 2. Detalles de las técnicas defensivas XSS recientes

Fortalezas	Limitaciones
Tiene sobrecarga de bajo rendimiento. Protege la aplicación Web del ataque XSS por construcción a través de aplicar la desinfección consciente del contexto en las plantillas de la aplicación Web.	Determinación de contexto de la variable no fiable en plantillas Web. En algunos lugares no se hace efectivamente debido a las peculiaridades de Navegador.
Se garantiza la escritura menos construcción del HTML en el lado del navegador a pesar de tener un explorador que analiza las peculiaridades. Asegura que la aplicación Web y el navegador tienen una comprensión idéntica del contenido generado HTML.	Incurre en gastos generales de rendimiento. Requiere modificaciones en ambos lados, es decir, el lado del cliente y del servidor. No protege contra los ataques No-scripting.
Permite a la aplicación Web utilizar HTML enriquecido Contenido en sus páginas Web. Intenta detectar la presencia de guiones al observar el comportamiento de infectada y benigna respuesta HTTP.	El uso de esta técnica perturba los analizadores del navegador para deducir código HTML inseguro que puede ser susceptible a ataques XSS basado en el uso de las peculiaridades de análisis de navegador.
Este método utiliza enfoques de análisis tanto estático como dinámico para identificar la existencia de código de script JavaScript ilícito en el código fuente de la aplicación Web.	El costo de implementación de esta técnica es alto debido al uso de componentes dinámicos.
Minimiza la trayectoria de propagación del gusano XSS limitándolo a una vista particular.	Esta técnica no proporciona protección contra la unidad por descarga, ataque de phishing y no protege cada vista de Infectados.
Refuerza la restricción de integridad en los contenidos en la aplicación Web para defenderse contra la inyección del código JavaScript malicioso.	No es eficaz contra la detección de los DOM basados en XSS. Requiere modificaciones en el lado del cliente y del lado del servidor.
Utiliza las funciones para detectar el código JavaScript Ofuscación que se utiliza para el ataque XSS. Tiene alto tasa de detección y menos tasa de falsas alarmas.	Hay algunos scripts benignos para los que la aplicación Web utiliza Ofuscación para enviarlo al usuario. Por lo tanto, clasificar en este caso no funciona correctamente.
Posee la capacidad de detectar la inyección de script malicioso calculando la desviación entre la solicitud HTTP y la Web de respuesta HTTP.	Debido a la ausencia de script malicioso reciente en la base de datos, aumenta la tasa positiva al detectar el ataque XSS almacenado.
Hace la coincidencia de cadenas aproximada para detectar la Inyección parcial de guion para prevenir contra el ataque XSS.	La coincidencia parcial de la cadena maliciosa puede convertirse en la causa de las tasas de falsos negativos. Inicialmente tiene un alto falso de tasa positiva.
Utiliza el método de aleatorización en el espacio de nombres XML, prefijos de etiquetas en cada documento. Es altamente impredecible para que el atacante no pueda adivinarlo. Básicamente elimina todas las dificultades que se producen durante el proceso de sanitización.	No es eficaz proporcionar protección contra los Código JavaScript desde el sitio Web remoto.

Fuente. CHAUDHARY Pooja, GUPTA B.B., GUPTA Shashank. Cross-Site Scripting (XSS) Worms in Online Social Network (OSN): Taxonomy and Defensive Mechanisms. 2016. p. 2134

- **IMPLEMENTACIÓN DE LA DETECCIÓN ESTÁTICA INVERSA SOBRE ATAQUES XSS**

Después de analizar un código fuente de la aplicación java, se diseña el proceso de detección de defectos XSS.

Programa de pre procesamiento: El sistema utiliza el código fuente de la aplicación Web Java, incluyendo archivos JSP, archivos Java Servlet y archivo web.xml, que contiene información de llamada. Dado que los defectos XSS reflejados pueden existir en el archivo JSP, se traduce archivos JSP a archivos Java Servlet, para que se pueda analizar. Luego usa el simulacro Método de prueba para utilizar el archivo web.xml para generar llamadas para todos los códigos fuente de Java, para que el análisis estático pueda cubrir mientras reduce falsos negativos.

Análisis de información de llamadas del código fuente: Utiliza el analizador de código Java para generar la llamada de programa, la información forma el método de la entrada de la prueba simulada generada.

Análisis de la información del programa: Utiliza el analizador basado en consultas de base de datos y aplica las reglas de análisis sensibles al contexto para analizar la información del programa, que se generó en el análisis 3, Para simplificar la información del programa.

Análisis de concordancia de reglas de defectos: Aplica las reglas de análisis de defectos XSS inverso para analizar la información que se generó en 3, y emitir el resultado de análisis de defectos XSS[16].

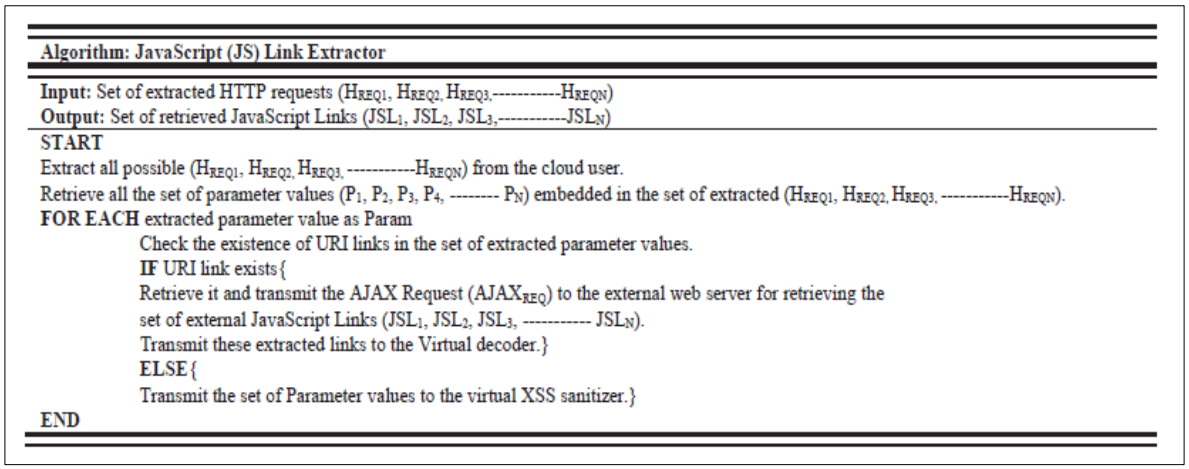
- **MARCO DE TRABAJO DEFENSIVO CONTRA ATAQUES XSS**

Este marco defensivo XSS es utilizado para las plataformas de la nube que detecta la transmisión de gusanos XSS reflejados en las aplicaciones Web implementadas en las máquinas virtuales del entorno de la nube. La técnica es de doble base: En primer lugar, detecta los enlaces URI incrustados en la solicitud HTTP. La existencia de enlaces URI apunta hacia los archivos JavaScript externos (JS) ubicados en los servidores remotos de las plataformas en la nube. Por lo tanto, el objetivo clave de la primera fase es extraer el conjunto de archivos JS externos mediante la referencia de sus enlaces URI. La segunda fase extrae el conjunto de scripts incrustados en el documento generado de respuesta HTTP.

Esos scripts generalmente surgen debido a la presentación del método GET / POST en la solicitud HTTP. Ambos conjuntos de scripts extraídos en estas dos fases deben ser decodificado de manera recursiva hasta que todos los caracteres JS de ambos conjuntos se transformen en forma legible. Ahora, ambos conjuntos de guiones decodificados serán explorados por cualquier similitud entre sus personajes. Si alguna semejanza se observa en estos dos conjuntos de secuencias de comandos se declara que hay la presencia del gusano XSS reflejado.

Los módulos principales del marco defensivo XSS basado en la nube propuesto se implementan en Virtual Cloud Server (VCS). Cuenta con seis módulos principales: External JavaScript (JS), Extractor de enlaces, Generador de documentos, extractor de secuencias de comandos, Decodificador virtual, Detector de semejanza y Virtual XSS Sanitizador.

Figura 6. Algoritmo para extraer archivos JavaScript externos (JS)



Algorithm: JavaScript (JS) Link Extractor

Input: Set of extracted HTTP requests (H_{REQ1}, H_{REQ2}, H_{REQ3},----------H_{REQN})
Output: Set of retrieved JavaScript Links (JSL_1, JSL_2, JSL_3,----------JSL_N)

START
Extract all possible (H_{REQ1}, H_{REQ2}, H_{REQ3}, ----------H_{REQN}) from the cloud user.
Retrieve all the set of parameter values (P_1, P_2, P_3, P_4, -------- P_N) embedded in the set of extracted (H_{REQ1}, H_{REQ2}, H_{REQ3}, ----------H_{REQN}).
FOR EACH extracted parameter value as Param
    Check the existence of URI links in the set of extracted parameter values.
    IF URI link exists{
    Retrieve it and transmit the AJAX Request ($AJAX_{REQ}$) to the external web server for retrieving the
    set of external JavaScript Links (JSL_1, JSL_2, JSL_3, ---------- JSL_N).
    Transmit these extracted links to the Virtual decoder.}
    ELSE{
    Transmit the set of Parameter values to the virtual XSS sanitizer.}
END

Fuente. GUPTA Shashank, GUPTA B.B. Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment.2016. p. 1598

Extractor de vínculos JavaScript (JS) externo: Este módulo es responsable de recuperar los archivos JS externos ubicados en los servidores remotos de las plataformas en la nube. Esto se hace generalmente haciendo referencia a los enlaces URI incrustados en la solicitud HTTP. El algoritmo mostrado (véase Figura 6) ilustra el proceso de recuperación de la JS externa, archivos ubicados en servidores remotos de entornos de nube computacional.

Generador de documentos: Este módulo es responsable de generar el documento HTML o página Web.

Script Extractor: Este componente extrae los scripts inyectados incrustados en el mensaje de respuesta HTTP. Los Script extraídos pueden ser a través de método GET o método POST. Este componente también extrae la lista de posibles funciones maliciosas que pueden utilizarse para ejecutar las operaciones maliciosas.

Descifrador Virtual: Este segmento es responsable de ejecutar el procedimiento de decodificación recursiva en ambos scripts obtenidos en el módulo extractor de vínculo JS externo y el módulo extractor de script.

Detector de semejanza: El objetivo clave de esta unidad es detectar el conjunto similar de caracteres JS en ambos Scripts. Cualquier tipo de semejanza observada en ambos conjuntos será declarado como un posible XSS gusano[17].

- **DESINFECTANTE XSS PARA USO DE NAVEGADOR INDEPENDIENTE PARA LA PREVENCIÓN DE ATAQUES XSS**

El diagrama del navegador propuesto se llama BIXSAN desinfectante. El XSS desinfectante presente en el servidor, se invoca cuando el usuario inyecta el código en un campo particular de una aplicación Web. El código inyectado se desinfecta y se convierte en DOM, modelo de objetos del documento. Cuando este código es inyectado se devuelve al lado del cliente, el generador de árbol de análisis genera un árbol de análisis sin problemas.

Funcionalidad del Algoritmo:

El contenido HTML creado por el usuario se pasa al desinfectante XSS.
El desinfectante XSS analiza el contenido HTML proporcionado por el usuario.
Durante la fase de análisis, se comprueba el contenido HTML
Las etiquetas estáticas se conservan dónde como el resto de las etiquetas (no estático) se filtran.
Aunque las etiquetas estáticas están destinadas a no invocar contenido dinámico, hay algunas razones de análisis, que utilizan etiquetas estáticas para invocar contenido dinámico.
Ejemplo: la etiqueta <img> Es una etiqueta estática pero
<Img src = "javascript: alert ('doevil')"> Invoca el JavaScript. Esta es una razón de análisis.
Para filtrar estas razones de análisis, la solución propuesta utiliza el probador de JavaScript. Las etiquetas estáticas se envían al probador de JavaScript.
El ensayador (Tester) JavaScript devuelve true, si el contenido dado es un JavaScript, sino un valor falso si el contenido no es un script.
Después de filtrar el contenido HTML inseguro, el contenido retenido se convierte en DOM[18].

- **INGENIERÍA INVERSA PARA LA DETECCIÓN DE VULNERABILIDADES XSS**

LigRE es un enfoque de ingeniería inversa que guía a la Fuzzing hacia la detección de vulnerabilidades XSS, primero se construye un modelo de flujo de datos de control, y luego se genera un modelo para guiar el fuzzing.

Inferencia de Control y Flujo de Datos: Paso A, aprenda el flujo de control de la aplicación, utilizando un estado consciente Crawler (o araña de la Web), para maximizar la cobertura. El paso B anota lo deducido en el Modelo con flujos de datos observables de valores de entrada en salidas para producir un modelo de control más flujo de datos.

Slicing y Fuzzing: Para conducir la aplicación al origen t_{src}, para enviar una aplicación maliciosa Valor x_{src}, y luego guiar el fuzzer para navegar hacia T_{dst}, para observar los efectos de x_{src}[19].

- **PROTECCIÓN DE LA SEGURIDAD DE LAS APLICACIONES WEB CONTRA EL ATAQUE DE REFERENCIA A OBJETO INSEGURO.**

Las sugerencias se refieren a tres aspectos: Funciones técnicas, detección de seguridad y la administración de seguridad.

Funciones técnicas.

Los administradores deben escribir un módulo de seguridad universal, que es independiente del sistema particular. En consecuencia, como componente del sistema, el módulo de seguridad permitirá que el sistema pueda tener una política de seguridad flexible si está bien configurado. Entonces, los usuarios no necesitan considerar el problema de seguridad por separado cuando desarrollan un sistema de aplicación Web, y no se pierda los problemas de seguridad debido a la omisión involuntaria.

Fortalecer la seguridad de los servidores de aplicaciones Web. Los administradores pueden desplegar el módulo de seguridad. En la programación de seguridad los desarrolladores deben cumplir con algunas normas particulares de la programación para evitar los problemas de seguridad de la aplicación. Por ejemplo, cada función o archivo sólo realiza una función básica, porque cuanto más conciso es el programa, menos error producirá.

Detección de seguridad.

La detección de seguridad es un paso importante para la seguridad de las aplicaciones Web. Se puede comprobar la seguridad de las aplicaciones Web mediante el examen del registro de las aplicaciones Web. Los siguientes son los elementos que se deben inspeccionar en el registro de la Web.

El registro de la Web: Las palabras clave de sentencias SQL tales como seleccionar, insertar y soltar, algunos caracteres especiales tales como "-", "1", "\" y "=";

El registro del sistema de protección Web: Las páginas Web que son atacados, el IP original de ataque.

El registro de dispositivos de red como firewall e IDS: La conexión desde el servidor a la red externa, la conexión de los puertos; La distribución de la PI y las peticiones.

El registro de la base de datos: El registro de la creación de Biblioteca o tabla por la base de datos, el registro del proceso de almacenamiento, el registro de los datos de importación y exportación.

La comprobación de la lista Web: El registro de la modificación del fichero, el último tiempo de acceso del archivo; subir archivo.

Administración de Seguridad.

La mayoría del personal involucrado suele prestar más atención a la técnica y descuidar las vulnerabilidades que son causadas por la administración de seguridad descuidada. Por lo tanto, se debe fortalecer la educación para la seguridad e introducirla. De esta manera se puede prevenir el potencial problemas de seguridad. Otra cosa importante es construir un sistema de gestión, como la aplicación de la formación y el personal al sistema operativo y de mantenimiento, y el acceso a la seguridad de las aplicaciones Web periódicamente[20].

- **DESARROLLO DE ARQUITECTURA DEL SISTEMA SCAAMP**

El sistema mostrado es una aplicación Web y consta de tres componentes principales y un auxiliar para la herramienta SCAAMP. Los principales componentes son: La configuración de seguridad del Auditor, Fixer y configuración

de Módulo de clasificación de seguridad. El componente auxiliar contiene, los módulos de utilidades.

El flujo de trabajo típico de SCAAMP es un usuario desarrollador Web o administrador que desea una auditoría o configuración de seguridad de un determinado servidor AMP mediante el módulo Web UI y el inicializador, el usuario suministra las credenciales utilizadas para inicializar el sistema, por ejemplo, detectar rutas de acceso a la plataforma y al archivo de configuración para cada componente de AMP, tokens de permiso.

Para el módulo Fixer, un punto culminante de su funcionamiento interno, en primer lugar, es recoger valores. Luego busca la configuración del archivo con el nombre como una clave de búsqueda. Cuando encuentra una coincidencia y el valor actual es diferente del valor del menú de cambio, asigna el valor enviado vía el menú del cambio al valor actual. Esto se repite como siempre y cuando los archivos cambiados no se agoten.

El servidor HTTP, el intérprete y el servidor de base de datos se adjuntan a sus respectivos archivos de configuración para que el Auditor pueda obtener los valores actuales y modificar con el Fixer para cambiar los valores de configuración. El módulo Resetter se utiliza para facilitar el reinicio del subyacente del servidor ya que la mayoría de los cambios en los ajustes de configuración requieren restablecer el entorno del servidor.

La arquitectura del sistema es razonablemente genérica en el sentido de que podría adaptarse fácilmente a pequeñas modificaciones a otros entornos de servidor reemplazando las instancias reales del servidor Web, el script del lado del servidor es un intérprete y servidor de bases de datos[21].

- **SOLUCIÓN PARA UNA CONFIGURACIÓN ERRÓNEA**

Un atacante puede identificar vulnerabilidades dentro de la pila de aplicaciones debido a una configuración incorrecta y utilizar incorrectamente estas vulnerabilidades para comprometer la aplicación, el sistema de archivos y / u otros sistemas.

Estructura.

Hay tres tipos de servidores: Base de datos, Web y Aplicación. Las aplicaciones pueden ser personalizadas o administrativas. El atacante opera en la aplicación. Una aplicación puede tener configuraciones diferentes. Los usuarios tienen cuentas para utilizar.

Un diagrama de secuencias para el caso de uso describe una amenaza que se aprovecha de las configuraciones erróneas de seguridad. El atacante compromete una aplicación cuya seguridad está mal configurada. El atacante debe tener acceso a la aplicación. La aplicación ha configurado incorrectamente su seguridad. La mala configuración de seguridad debe ser accesible para el atacante.

El atacante explora la aplicación y encuentra la configuración errónea de seguridad. A continuación, el atacante utiliza incorrectamente la configuración incorrecta de seguridad para comprometer la aplicación. La aplicación está comprometida, lo que puede conducir a una variedad de usos erróneos[22].

- **TÉCNICAS PARA LA IDENTIFICACIÓN DE CONFIGURACIONES ERRÓNEAS**

Para identificar posibles configuraciones incorrectas de políticas, se debe utilizar primero la minería de reglas de asociación para detectar patrones estadísticos o reglas, a partir de una base de datos central de accesos. Luego se analiza los datos usando estas reglas para predecir potenciales configuraciones erróneas o instancias de los datos para los cuales las reglas no son válidas. Una vez que se determina si una predicción era o no correcta, se incorpora el resultado. En un mecanismo de retroalimentación para promover el uso continuado de reglas que reflejen con exactitud políticas y reglas que no lo hacen.

Para ilustrar la utilidad de esta técnica, considere el siguiente escenario. Chelín es un nuevo estudiante que es asesorado por Alice, un profesor. Bob y Alice trabajan donde el mismo sistema controla el acceso a la oficina de Alice, la oficina de Bob que es compartido con algunos de los otros estudiantes de Alice, un laboratorio compartido, y una sala de máquinas.

Cuando el departamento asigna a Bob una oficina, configura la política de control de acceso para permitir a Bob que acceda a su oficina. La primera vez que Bob intenta acceder al espacio compartido del laboratorio, se le niega el acceso como resultado de la mala configuración, momento en el que debe ponerse en contacto con Alice o el departamento para corregirlo. Este proceso es intrusivo y podría tomar minutos o incluso horas.

Sin embargo, las acciones pasadas de los compañeros de la oficina de Bob sugirieron que las personas que acceden también accedan al espacio compartido del laboratorio y a la sala de máquinas. Las técnicas que se describen permiten inferir del acceso de Bob a su oficina que Bob es probable que necesite acceso al espacio de laboratorio y sala de máquinas.

Identificar esto por adelantado permite corregir la configuración errónea antes de que resulte un acceso denegado y tiempo perdido. La detección de la mala

configuración se logra utilizando sólo el historial de accesos en el sistema. La técnica es independiente del control de acceso subyacente, mecanismo, política y lenguaje de especificación[23].

- **MODELO DE DETECCIÓN DE FUGAS DE DATOS QUE PRESERVA LA PRIVACIDAD**

El método de detección de fugas de datos que preserva la privacidad como servicio y minimiza el conocimiento que un proveedor de DLD puede obtener durante el proceso. Se enumera las seis operaciones ejecutadas por el propietario del proveedor de DLD en el protocolo. Incluyen pre procesamiento por el propietario de los datos para preparar los resúmenes de datos, el lanzamiento para que el propietario de los datos envíe al proveedor de DLD, monitor y detectar para que el proveedor de DLD pueda recopilar el tráfico saliente de la organización, calcular los resúmenes del contenido de tráfico, e identificar fugas potenciales.

El proveedor de DLD puede devolver alertas de pérdida de datos al propietario de los datos donde puede haber falsos positivos, es decir, falsas alarmas y un proceso posterior para que el propietario de los datos muestre la verdadera fuga de datos. El protocolo se basa en datos de cómputo estratégico específicamente la similitud cuantitativa entre la información sensible y el tráfico de red observado. La similitud indica fugas de datos potenciales.

Para la detección de fugas de datos, la capacidad de tolerar un cierto grado de transformación de datos en el tráfico es importante. La idea clave para una comparación rápida y tolerante es el diseño y el uso de un conjunto de características locales que son representantes de patrones de datos locales, por ejemplo, cuando el byte b2 aparece en los datos sensibles, suele estar rodeado de bytes b1 y b3 formando un patrón local b1, b2, b3.

Características locales para preservar patrones de datos incluso cuando las modificaciones inserción, supresión y sustitución se hacen a parte de los datos. Por ejemplo, si un byte b4 se inserta después de b3, el patrón local b1, b2, b3 se retiene a través del patrón global, por ejemplo, un hash de todo el documento. Para lograr la privacidad el propietario de los datos genera un tipo especial de digerir, que se llama huellas digitales borrosas. Intuitivamente, el propósito de las huellas dactilares difusas es ocultar los datos sensibles verdaderos en una muchedumbre. Evita que el proveedor de DLD aprenda su valor exacto[24].

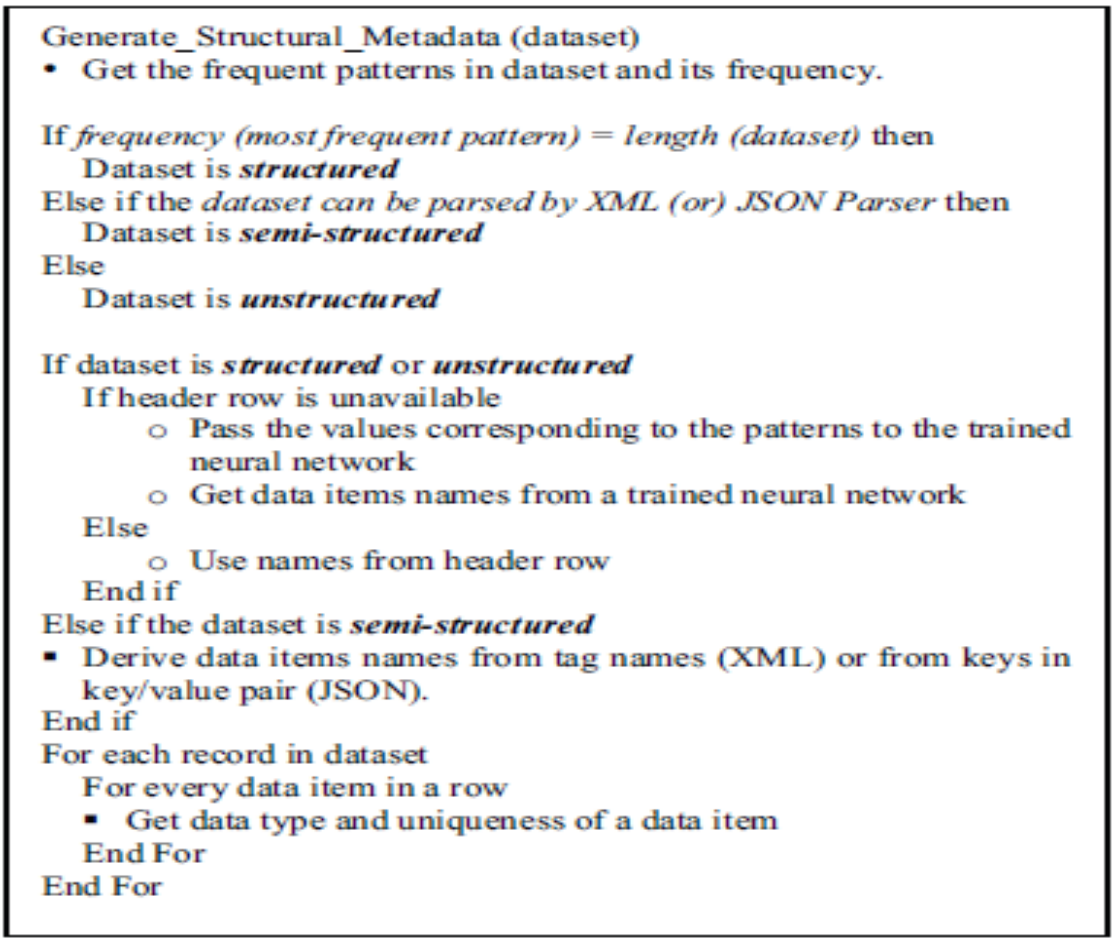
- **MARCO DE CONTROL DE ACCESO HADOOP PARA IDENTIFICAR ELEMENTOS DE DATOS CONFIDENCIALES**

El crecimiento reciente en los grandes datos está aumentando los problemas de seguridad y privacidad. Las organizaciones que recopilan datos de varias fuentes corren el riesgo de contraer obligaciones legales o comerciales debido a la violación de la seguridad y a la exposición de información delicada. Sólo el control de acceso a nivel de archivo es factible en la implementación actual de Hadoop y la información sensible sólo se puede identificar manualmente o de la información proporcionada por el propietario de los datos.

Un marco propuesto está compuesto por un generador de metadatos mejorado (EMG), el cual genera tanto metadatos estructurales y descriptivos. Los metadatos estructurales contienen información sobre los elementos de datos del conjunto de datos considerando que los metadatos descriptivos proporcionan un contenido en el conjunto de datos. Siempre que se almacene un nuevo conjunto de datos en Hadoop, EMG genera metadatos relevantes y supresión del conjunto de datos, los metadatos correspondientes son eliminado, con el fin de hacer que los metadatos generados sean confiable y disponible, se almacena en el Hadoop sistema de archivos distribuido (HDFS).

El pseudo-código para el Algoritmo para el generador de metadatos estructurales se muestra en (véase Figura 7).

Figura 7. Pseudocódigo para generador de metadatos estructurales



Generate_Structural_Metadata (dataset)
- Get the frequent patterns in dataset and its frequency.

If *frequency (most frequent pattern) = length (dataset)* then
  Dataset is ***structured***
Else if the *dataset can be parsed by XML (or) JSON Parser* then
  Dataset is ***semi-structured***
Else
  Dataset is ***unstructured***

If dataset is ***structured*** or ***unstructured***
  If header row is unavailable
    - Pass the values corresponding to the patterns to the trained neural network
    - Get data items names from a trained neural network
  Else
    - Use names from header row
  End if
Else if the dataset is ***semi-structured***
- Derive data items names from tag names (XML) or from keys in key/value pair (JSON).

End if
For each record in dataset
  For every data item in a row
  - Get data type and uniqueness of a data item

  End For
End For

Fuente. TK Ashwin Kumar, LIU Hong, THOMAS Johnson P, MYLAVARAPU Goutam. Identifying Sensitive Data Items within Hadoop. 2015.p.1310

La mejora realizada al generador de metadatos es la adición de una red neuronal para identificar los datos e implementación de técnicas de encadenamiento léxico para generar metadatos descriptivos. El rastreador de uso de datos (DUT) en el marco propuesto implementa el uso de tracker propuesto. DUT rastrea los usuarios, patrones de uso, que consisten en identidad de usuario, marca de tiempo, conjuntos de datos y elementos de datos a los que accede. Todos los patrones de uso generados almacenados en HDFS como metadatos.

El Analizador de similitudes de datos (DSA) mide la similitud entre conjunto de datos al ir combinando la similitud de contexto y la similitud de uso, a partir de los metadatos generados por EMG y patrones de uso generado por DUT. La puntuación de similitud es entre 0 Y 1. Los elementos de datos disjuntos tienen sus puntuaciones de similitud cero, mientras que los datos similares tendrán una

mayor puntuación. La arquitectura de DSA tiene dos componentes principales, es decir, el patrón analizador de uso de datos y analizador de similitud de contexto de datos[25].

- **MODELO RBAC PARA CONTROL DE ACCESO A NIVELES DE FUNCIONES**

El modelo control de acceso basado en roles RBAC para sistemas operativos basados en Linux, apoya jerarquías en ambos roles y permisos. Las jerarquías de roles permiten a las organizaciones modelar sus funciones organizativas existentes en el sistema de control de acceso con un esfuerzo mínimo. Las jerarquías de permisos agregan soporte para fácil mantenimiento de permisos en el sistema RBAC. Sobre el contrario, un modelo de permisos planos requerirá un rol de gerente para ser asignado a la mayoría de los permisos en el sistema directamente y las nuevas asignaciones son necesarias siempre que un nuevo permiso es introducido en el sistema, mientras que en un permiso en el rol de gerente debe ser asignado a los pocos padres Nodos que representan todos los permisos descendentes previstos.

Los roles más altos en la jerarquía heredan todos sus descendientes funciones. Por ejemplo, el rol raíz tendrá implícitamente todo. Tener los permisos más altos en la jerarquía significa tener acceso a todos sus descendientes. Por ejemplo, tener el permiso de root es igual a tener acceso a todos los permisos en el sistema implícitamente. Este modelo es adecuado para el control de acceso a nivel de función, como es muy rápido en los cheques de permiso, pero relativamente lento en modificaciones a las jerarquías de rol / permiso.

El nivel de funciones en los sistemas de control de acceso se utiliza para autorizar acciones en un sistema, tales como la ejecución de funcionalidades individuales. En contraste con el control de acceso a nivel de función, control de acceso a datos o recursos requiere una semántica diferente y un modelo que sea eficiente con frecuencia añadiendo y eliminando roles y / o permisos.

Una verificación de permisos es la operación fundamental RBAC, verificando si un usuario tiene un permiso específico o no, y necesita ser lo suficientemente eficiente para ser realizado antes de ejecutar cada acción en un sistema, sin incurrir en gastos indirectos significativos. Los siguientes son los pasos básicos involucrados en una verificación de permisos:

Seleccionar todos los roles directos del usuario U, que son todos los roles U se asigna explícitamente a.
Seleccionar todos los descendientes de los roles directos, como indirectos de U.
Seleccionar todos los permisos asignados directamente a los roles seleccionados

Seleccionar y comprobar si todos los roles descendientes tienen permisos indirectos.
Este proceso puede requerir memoria significativa, ya que muchos controles implican la mayoría de las funciones.

2.2 ESTRUCTURA DE LOS TIPOS DE VULNERABILIDAD QUE AFECTAN LA SEGURIDAD WEB

INYECCIÓN DE SQL

- **Atacante.**

La inyección de SQL es una técnica para explotar de manera malintencionada aplicaciones que utilizan datos proporcionados por el cliente con sentencias SQL. Los atacantes engañan las operaciones de destino que llegan al receptor con la ejecución de comandos SQL no deseados mediante el suministro de fuentes de entrada que son los insumos provistos por el usuario u otro programa. De esta manera, los atacantes obtienen acceso no autorizado a una base de datos, permitiéndoles ver o manipular datos restringidos.

El uso de la vulnerabilidad de la inyección de SQL, es donde un atacante puede leer, modificar o incluso eliminar la información de la base de datos. En muchos casos, esta información es confidencial o sensible. Las causas de la inyección de SQL son las vulnerabilidades que son relativamente simples y bien comprendidas. Un ataque de inyección de SQL (SQLIA), es una clase de inyección de código que aprovecha la falta de validación de las fuentes de entrada.

Estos ataques ocurren cuando los desarrolladores combinan las cadenas codificadas con el usuario proporcionado de entrada para crear consultas. Si las fuentes de entrada no están correctamente validadas, los atacantes pueden ser capaces de cambiar las cadenas codificadas de SQL con la inserción de nuevas palabras clave SQL a través de cadenas de entrada especialmente diseñadas[4].

- **Herramienta.**

Este tipo de vulnerabilidad generalmente el atacante utiliza como herramienta scripts o programas maliciosos. Se da un ejemplo para trabajar con un $\boldsymbol{if}$ de lenguaje de programación simple que tiene dos variables de dominio: Números y cadenas.

Se fija un conjunto de variables numéricas ***N*** y un conjunto de variables de cadena ***S***, se usa ***m*** para denotar una variable numérica y ***s*** para denotar una cadena de variables, se arregla un conjunto de funciones ***fi***, cada uno de los cuales toma una

tupla de número /valores de cadena y devuelve un número. Se estima un conjunto de funciones ***gi*** que toman una tupla de valores de cadena / número y devuelven una cadena.

Estas funciones, son funciones nativas y personalizadas. Una cadena str es una cadena de consulta especial, y un comando especial Query_execute() es una operación objetivo, se supone que la consulta se produce exactamente una vez, cada uno sin entrada la variable numérica se inicializa a 0 y cada cadena sin entrada de variable se inicializa a cadena nula (véase Figura 8)[4].

Figura 8. Sintaxis de programa

P	:=	defn; stmt; query_execute(str_1,…,str_k)	(program)
defn	:=	number *m* | string *s* | defn	(variable declaration)
stmt	:=	if (expr) {stmt} else {stmt} | stmt	(statement)
		| *m* := *ae* | *s* := *se* | *term* assop *term* | skip	
expr	:=	*term* relop *term* | *term*	(expression)
term	:=	id | digit	(term)
assop	:=	= | += | −= | /= | *=	(assignment operators)
relop	:=	< | <= | == | <> | >= | >	(relational operators)
ae	:=	*m* | f_i (t_1,…,t_k)	(arithmetic expressions)
se	:=	*s* | *str* | g_i (t_1,…,t_k)	(string expressions)
id	:=	letter (letter | digit)*	(identifier)
letter	:=	[A-Za-z]	(letter)
digit	:=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9	(digit)

where $m \in N$ is any number constant,
$s \in S$ is any string constant,
each t_i is either *ae* or *se*, depending on the parameters for f_i, g_i, respectively.

Fuente. YOUNG, Su Jang. JIN, Young Choi. Detecting SQL injection attacks using query result size.2014.p.110.

El siguiente programa es un ejemplo que muestra el funcionamiento del ataque, la variable ***m*** representa el precio y la variable ***title*** el título de un libro, éstas son las variables de entrada, y generan diferentes declaraciones de consultas dinámicas dependiendo del valor de ***m***. La entrada de valor de ***m*** determina la consulta, las secuencias de las declaraciones de consulta son ejecutadas por el programa (véase Figura 9).

Figura 9. Programa ejemplo

```
1.  Program P
2.  number m, price;
3.  string title, str;
4.  if (m == 0) {
5.      str = "Select * From BOOK Where title ='"+title+"'";
6.  }
7.  else {
8.      str = "Select * From BOOK Where title ='"+title+"' AND price < "+price;
9.  }
10. query_execute(str);
```

Fuente. YOUNG, Su Jang. JIN, Young Choi. Detecting SQL injection attacks using query result size. 2014. p.111.

Definición y verificación de vulnerabilidades de consulta.

Definición 1: Operación de destino asociada con la consulta control path.

Sea V el conjunto de variables de entrada. Para cualquier variable de entrada V el programa P toma una operación de destino asociada a la consulta control path. Por lo tanto, la consulta control path PATHv finalizará con la operación de destino query_execute () y la operación de destino toma una consulta control path y una declaración consulta.

Definición 2: Vulnerabilidad de variables de entrada

v_π denota el conjunto de variables válidas. Se supone que se tiene una función de representación de entrada válida $VF: VV_\pi$. Con v_π se hace una sustitución benigna y segura que corresponde a v, $\{(v\ VF(v))|v\ \in DOM(VF)\} and\ VF(v) = v_\pi$. Por lo tanto si la entrada de la variable v es diferente de la variable v_π una entrada de variable v es vulnerable.

Dada una variable de entrada v, se puede ejecutar la operación destino y extraer la consulta control path. En consecuencia, se define la función $VF(v)$ a ser la variable v_π que mapea cada número y variable. Se debe tener en cuenta que $VF(v) = v_\pi * v_\pi$ es benigno y una variable segura que corresponde a v para alguna declaración de consulta. Una variable de entrada v para P es vulnerable si el objetivo de la operación de la declaración de la consulta q que P calcule en v es diferente de que la declaración de la consulta q' que P calcule en $VF(v)$[4].

Eso es, si se conoce la variable v_π, ejercer el objetivo de operación v hace que se pueda validar la diferencia entre v y v_π. Considerando las siguientes consultas, Query1 (ataque) y Query2 (Válido). Permitir que $\propto$ y β dos variables de entrada en el objetivo de operación: $\propto$ es una variable de entrada del título y β es una variable de entrada del precio.

La entrada v: $(m = 1, title =$ "'Ants' OR 1<>2-", $price = 20)$ ejecuta y genera una consulta: (véase Figura 9)

Select * From BOOK Where title = $\propto$ AND price < β, sustituyendo la entrada en las expresiones.

Query1: Select*From BOOK Where title = 'Ants' OR 1<> 2 - AND Price < 20**.**

Considere la variable v_π. El programa genera una consulta cuya declaración es muy diferente a la declaración de consulta anterior (Query 1). Sustituyendo v_π en el rendimiento de la sentencia de la consulta.

Query2: Select*From BOOK Where title = 'Ants' AND price < 20

En las instrucciones de consulta anteriores, las dos consultas tienen las mismas estructuras de árbol de derivación de consulta. Sin embargo, aunque las estructuras de consulta coinciden en la cláusula condicional, Query1 tiene una condición adicional después del conector "OR", que hace que el resultado de la ejecución de la consulta sea diferente.

Definición 3: Estimación del tamaño del resultado de la consulta

Sea q una instrucción de consulta correspondiente a la variable de entrada v sobre PATHv. Sea q_π una instrucción de consulta correspondiente a la variable v_π. Si la sentencia de la consulta q y q_π son iguales al tamaño del resultado de la consulta de q y q_π son también iguales.

La definición 3 es utilizada para estimar los tamaños del resultado de la consulta entre el objetivo de la operación. Se puede obtener el tamaño del resultado de la consulta de las operaciones de destino ejecutadas en PATHv[4].
Considere el seguimiento de las consultas, Query3 (ataque) y Query4 (valido). Sea:

$\propto$ y β dos variables de entrada, y sea la consulta string str algunos comentarios del desarrollador adicionales. Dado la entrada v: $(m = 1, title =$ "Ants' OR 1<>2-", $price = 20)$, el programa ejerce y genera una sentencia de consulta:

Select*From BOOK Where title = $\propto$ AND price < β – Esto es un comentario de la consulta.

Sustituyendo la entrada dentro de las expresiones.

Query3: Select*From BOOK

Where title = 'Ants' OR 1<>2-AND price <20 - Esto es un comentario de consulta
Considere la variable v_π. Sustituyendo v_π dentro del rendimiento de la sentencia de la consulta:

Query4: Select*From BOOK

Where title='Ants' AND price <20 – Esto es un comentario de la consulta

En las instrucciones de consulta anteriores, las dos consultas tienen las mismas estructuras de árbol de derivación de consulta con un token de comentario. Sin embargo, la vulnerabilidad potencial es que el desarrollador puede asumir que, debido a que la consulta analiza correctamente, el valor almacenado en el precio del objeto de la solicitud es el mismo que el utilizado para generar la consulta.

Incluso si las estructuras de consulta coinciden en la cláusula condicional, Query3 tiene una condición de comentario de consulta adicional después del "OR" conectivo, lo que hace que el tamaño del resultado de la consulta sea diferente. Es decir, la comparación de la estructura de árbol de derivación de consulta considera sólo la parte estructural de su árbol de derivación.

Si una semántica existe en una vulnerabilidad de consulta, esto no se detectará comparando la estructura de árbol de derivación de la consulta. Esta limitación existe porque la estructura de árbol de derivación de la consulta no puede detectar vulnerabilidades de consultas maliciosas de semántica[4].

- **Vulnerabilidad.**

Las aplicaciones Web son conocidas por múltiples ataques de inyección de código SQL, que pueden presentarse por un diseño inadecuado o una implementación no adecuada. La inyección SQL se clasifica en clases, amenazas, ataques y tipos que son mecanismos o canales explotados por atacantes como cookies[26].

Tipos de inyección SQL.

Tautologías.

Este tipo de ataque de inyección de SQL funciona haciendo la cláusula "WHERE" siempre verdadera, y esto resultará omitiendo la condición dentro de la instrucción SQL. Los atacantes usan la inyección SQL de tautología para la autenticación. También agregan comentarios en línea para ignorar la parte restante de la declaración para lograr la máxima cantidad del resultado a cambio con el rango más bajo de condiciones.

La mayoría de las tautologías de SQL son útiles cuando el atacante intenta obligar a una sentencia SQL a devolver todos los registros, al ignorar todas las condiciones WHERE. La tautología común es "o 1 = 1". Pondrá otra condición concatenando los criterios "OR" y "1 = 1" que siempre es verdad así que el resultado de la condición entera será verdadero.

SELECT * FROM TABLE_USERS WHERE
USERNAME ='ADMINISTRADOR' y PASS = 'ROOT'

Se muestra una consulta SQL para obtener todas las columnas que su Nombre de usuario es igual a ‘Administrador’ y su contraseña ‘root’, luego hay un valor de variable que se inyecta como un ataque SQL 'OR 1 = 1. La consulta SQL final es el resultado de la concatenación de la variable externa y de la declaración SQL principal.

La ejecución de esta consulta devolverá todas las filas de la "TABLE_USERS" que su nombre de usuario es "ADMINISTRADOR" sin tener en cuenta su contraseña[27].

SELECT * FROM TABLE_USERS WHERE USERNAME=’ADMINISTRADOR’ and PASS=’ OR 1=1 –‘

Consultas ilegales incorrectamente lógicas.

En esta técnica el objetivo del atacante es recopilar información sobre el esquema y la estructura de las tablas dentro de la base de datos. El atacante usa los datos recopilados para lanzar otro ataque. Básicamente el mecanismo de este ataque funciona inyectando una consulta SQL incorrecta en la Web. La aplicación Web devolverá alguna información acerca de la base de datos en forma de mensaje de error. El Inapropiado manejo de errores puede conducir a mostrar la base de datos interna.

Este tipo de mensajes de información crítica sobre la estructura de la base de datos, el atacante los usa para llevar a cabo otro ataque con impacto en el sitio Web. Por ejemplo, en el siguiente ataque, el atacante inserta "ABCD" "en el

campo de nombre de usuario y porque el (Doble Cotización) romperá la estructura de la sentencia SQL en el resultado. La base de datos devolverá un mensaje de error que revela otro Nombre de columna dentro de la cadena de consulta. Ahora el atacante sabe que existe una columna en la base de datos denominada "Contraseña".

Entrada (nombre de usuario): 'ABCD "
Entrada (contraseña): 'TEST "
Sql: SELECT * DE LOS USUARIOS WHERE username =
'ABCD' 'Y contraseña =' TEST '
Resultado: "Sintaxis incorrecta cerca de 'ABCD'.
Notas sin cerrar después del carácter
Cadena " Y Contraseña = 'PRUEBA' '. "

En el resultado de ataques de consultas incorrectas hay dos tipos de error lógica y sintáctica. Los errores lógicos se utilizan para buscar el nombre de las columnas o tablas. Pero los errores de sintaxis muestran cuyos parámetros están abiertos a ataques de inyección.

Unión Query.

El operador UNION en lenguaje SQL se utiliza para unir dos consultas independientes. En la consulta de unión, el atacante utiliza "UNION" para extraer datos de otras tablas inyectando otra consulta de selección y unión. Al utilizar este método, el atacante fuerza la base de datos para devolver el resultado de otras tablas distintas a la definida en la consulta SQL legítima.

Para extraer datos de la base de datos mediante el uso de unión, el atacante necesita tener la estructura de la base de datos tal como los nombres de las tablas y nombres de campos que posteriormente la consulta secundaria se basa en esa información y se une a ella con la cadena de consulta original. El atacante inserta una consulta SQL de unión al final de la URL en la barra de direcciones del navegador.

original: http: //www.example.com/news.php?
Newsid = 340
manipulada: http://www.example.com/noticias
.php? Newsid = 340 UNION SELECT CreditcardNo,
PinNo FROM CreditCardTable

Como resultado de la inyección de SQL, la consulta se verá similar abajo:

SELECT NewsTitle, NewsBody FROM Noticias
WHERE NewsID = '340' UNION SELECT
CreditcardNo, PinNo de CreditCardTable;

En consecuencia, la base de datos devolverá dos columnas. Los contenidos de estas columnas se unen a los resultados de la primera consulta y la segunda consulta. En este ejemplo, la primera fila es de la tabla de noticias y la segunda y tercera fila están buscando de la tabla de tarjetas de crédito (véase Tabla 3)[27].

Tabla 3. Ejemplo del conjunto de datos del resultado del ataque unión consulta

Título de las Noticias	Cuerpo de las Noticias
100 340 955 888 333	398
230 110 715 701 325	102

Fuente. SADEGHIAN Amirmohammad, ZAMANI Mazdak, MANAF Azizah. A taxonomy of SQL Injection Attacks. 2013. p.270

Consultas de Piggy-Backed.

En este tipo de ataque, el atacante inyectará una consulta independiente y en el resultado de un ataque exitoso la segunda consulta se ejecutará después de la primera consulta original que ya corría. La diferencia de este ataque con UNION ataque es que las consultas no se unirán entre sí, pero son completamente independientes. Este ataque es llamado piggy back porque la consulta secundaria será enviada a la base de datos bajo la cubierta de la primera consulta.

La implementación de este ataque sólo es posible si la base de datos está configurada de manera que otorgue este permiso al usuario para ejecutar varias consultas en la misma línea. Este tipo de ataque puede ser muy peligroso porque da la capacidad al atacante para añadir cualquier tipo de comando SQL que desee y ejecutarlo en la base de datos, lo que puede causar un incidente de alto impacto.

El punto y coma (;) desempeña un papel importante en este tipo de ataque porque el atacante lo usa como un delimitador para el final de la primera consulta y el inicio de la nueva consulta. Pero en alguna base de datos la existencia de un delimitador no es necesario. En el siguiente ejemplo se pude ver una consulta que busca noticias de la tabla de noticias basadas en 3 condiciones año autor y tipo.

SELECT * FROM noticias WHERE año = '. $ Year.' Y
Autor = '. $ Autor.' Y tipo = '. $ Tipo.'

Suponga que el atacante inserte:

Año de introducción: 2013
Entrada Autor: ; Caída de los usuarios de la tabla --
Tipo de entrada: public

Como resultado de insertar las entradas anteriores, la siguiente consulta se realizará, esta consulta seleccionará todas las noticias del Año 2013 y punto y coma terminar la primera consulta SQL y la segunda consulta eliminará la tabla de información del usuario e "-" ignorará el resto de la consulta[27].

SELECT * FROM noticias WHERE año = '2013' Y
Author = ''; Drop table users - ' Y TYPE='PUBLIC'

Procedimientos almacenados.

Los procedimientos almacenados son la parte prematura de las consultas SQL que están diseñados para realizar una tarea específica. Parte de la base de datos tienen sus propios procedimientos almacenados predefinidos para trabajar con el sistema operativo. También son vulnerables a ataques de inyección de SQL y el atacante puede ejecutarlos para lograr sus metas maliciosas.

Si el atacante ejecuta bases de datos predefinidas los procedimientos almacenadas, también podrán ejecutar comandos del sistema operativo de la máquina servidor (Privilegio escalada). Básicamente los procedimientos almacenados pueden ayudar a evitar la inyección de SQL mediante el imite de los tipos de sentencias que se pueden pasar a los parámetros SQL. En el ejemplo siguiente hay un procedimiento almacenado que recibe una categoría variable del mundo exterior.

```
ALTER PROCEDURE get_news (@category
NVARCHAR (50)) AS
BEGIN
DECLARE @sqlcmd NVARCHAR (MAX);
SET @sqlcmd = N'SELECT * FROM noticias WHERE
News_cat = '' '+ @category +' '' ';
EXECUTE (@sqlcmd)
END
```

Suponga que el atacante inserta la siguiente entrada:
SPORT'; SHUTDOWN; --

Como resultado de ejecutar esta consulta todas las noticias de la Mesa de noticias con la categoría de deporte y después de eso punto y coma finalizará la primera consulta y la segunda consulta se cerrará en el servidor SQL[27].

SELECT * FROM news WHERE news_cat = 'deporte';
SHUTDOWN; -

Inferencia.

En este tipo de ataque, los atacantes inyectan el SQL y observan las diferencias en el retorno de la aplicación Web. Básicamente el ataque lanzado es por hacer preguntas. Por ejemplo, si la respuesta es "A" hacer "M" o si la respuesta es "B" hacer "N". Normalmente este ataque tiene lugar cuando la aplicación Web se endurece en el aspecto del manejo de errores y el atacante no puede utilizar los mensajes de error. Hay dos técnicas principales de ataque:

Ataques de tiempo.

En el ataque de tiempo, la inyección de SQL del atacante tarda en entender la respuesta a su pregunta por el tiempo que tarda en cargar la página de resultados. Este tipo de ataque es probable en aplicaciones Web seguras porque se basan en el retraso que sucede cuando se ejecuta la inyección SQL y no la salida de la aplicación Web. En el siguiente ejemplo, el atacante pregunta desde la Base de datos, si la versión de base de datos contiene el número 4 (como 4) tiene un retraso de 10 segundos antes de reproducir y cargar la página.

Http://www.MyWebsite.com/news.php?id=12
0 y IF (versión () como '4%',
sleep (10), 'false')) –

El ejemplo anterior está en MySQL. El retraso del comando es diferente basado en el proveedor del sistema de administración de base de datos. Servidor Microsoft SQL utiliza el comando "WAITFOR". Oracle y MySQL utiliza el comando "SLEEP" para realizar el retardo.

Inyecciones ciegas.

La inyección SQL ciega es otra técnica de inyección de inferencia. En este tipo de ataque el atacante hace una pregunta verdadera / falsa y observa la respuesta basada en el comportamiento de la aplicación Web en respuesta (también conocido como contenido basado). Esta situación hace más difícil el proceso del atacante, pero no se puede evitar el ataque. Por ejemplo, asumir que la aplicación Web está suficientemente segura para evitar mostrar mensajes de error que contengan bases de datos estructuradas.
Pero si la aplicación sigue siendo vulnerable a inyección de SQL, el atacante le preguntará desde la Web si la primera letra del nombre de usuario de la base de datos es "A" muestra la página o viceversa. En el peor de los casos estos intentos continúan por 26 veces hasta que pueda adivinar la primera letra del nombre de la base de datos y, por consiguiente, debe repetir el mismo procedimiento para recuperar las otras letras del nombre de usuario de la base de datos[27].

En el ejemplo siguiente, el atacante trata de adivinar Nombres de tablas que existen dentro de la base de datos. En este caso, tuvo dos intentos, en el primer intento sobre la base de datos puso "Seleccione la primera fila como una tabla de administración". La tabla "admin" no existe en la base de datos, por lo que el resultado es falso y este resultado es parte de la condición AND. En consecuencia, la página no se cargará porque la condición no se satisface.

Http://example.com/news.php?id=132 AND
(Seleccione 1 del límite de administración 0,1) = 1

Se supone que existe una tabla denominada "usuarios" en la Base de datos, por lo que en el segundo intento la consulta puede seleccionar la primera fila de la tabla. Y esto satisface la condición y la página se cargará en resultado, ahora el atacante sabe que hay una tabla llamada "usuarios" que existe en la base de datos.

Http://example.com/news.php?id=132 AND
(Seleccione 1 del límite de usuarios 0,1) =1

Codificaciones alternativas.

La codificación alternativa no es un tipo de ataque independiente, pero es una técnica que se utiliza principalmente junto a otras técnicas de inyección SQL para evitar el sistema de seguridad Web de aplicación o infraestructura de red desde la detección del ataque. En otras palabras, sólo se utiliza como una cubierta para otros ataques a evadir de los sistemas de detección de intrusos (IDS).

En este tipo de ataque, las técnicas de codificación tales como Base64, ASCII, HEX o Unicode pueden usarse para engañar a la IDS / IPS cambiando el aspecto de la consulta de inyección de SQL. Se supone que se tiene un sistema de detección de intrusos con la siguiente firma. En esta firma, IDS buscará "'or 1 = 1 - ", si IDS encuentra correctamente el patrón, puede borrar la conexión y mostrar un error[27].

Alert tcp any any -> $ HTTP_SERVERS
$ HTTP_PORTS (msg: "SQL Injection attempt";
Flow: to_server, established; Content: "or
1 = 1 - "; nocase; Sid: 1; Rev: 1;)
En los siguientes ejemplos, se puede ver la codificación de la misma cadena de inyección de SQL para la evasión de IDS.

HEX encoding of 'or 1 = 1 – for use in URL:
% 31% 20% 4F% 52% 20% 31% 3D% 31
HEX encoding de 'o 1 = 1 -:
& # X2; & # x2; & # x20; & # x2; & # x3D; & # x

31;
Decimal encoding of 'or 1 = 1 -:
& # 49 & # 32 & # 79 & # 82 & # 32 & # 49 & # 61 & # 49
Base64 encoding of 'or 1 = 1 -:
MSBPUiAxPTE =

El uso de comentarios en el ataque también es una técnica de evasión muy común, los comentarios pueden cambiar la apariencia de la consulta para evitar la detección por IDSs y IPSs. Siguiendo el ejemplo, se utiliza el comentario "/ * * /" para eludir. En este ejemplo lo que esté entre / * y * / no se considerará a ejecutar en el servidor MySQL pero puede cambiar el aspecto del ataque en el resultado el IDS no puede detectarlo.

Los comentarios de DROP / * van aquí * / users =

En el siguiente ejemplo se puede ver que el uso de comentario incluso en el medio del nombre de la función es también posible y no se tendrá en cuenta para su ejecución en MySQL. En este ejemplo DR / ** / OP actuará como función DROP, por lo que esta flexibilidad de lenguaje SQL dará muchas alternativas al atacante para cambiar la apariencia de su ataque para evadir los algoritmos de detección DR / ** / OP users[27].

Amenazas de inyecciones SQL

Los piratas informáticos aprovechan la vulnerabilidad que se puede presentar en las sentencias SQL lo que permite obtener acceso a la información de la base de datos y otra información sensible como número de tarjeta de crédito y otra información monetaria manipulando la lógica del comando sql y permitiendo así a un atacante borrar y actualizar los datos almacenados en el base de datos. La inyección de SQL se puede definir como la técnica en la que el hacker ejecuta consultas SQL malintencionadas en el servidor de base de datos a través de una aplicación Web para obtener acceso a través de la información sensible o de la base de datos (véase Tabla 4)[7].

Tabla 4. Amenazas de inyección SQL

S. No.	Amenaza	Descripción
1	Suplantación de identidad	En este ataque personas son engañadas a creer que el correo respectivo o el sitio Web es genuino aunque realmente no es.
2	Modificar el precio de datos originales	En este ataque los hackers modifican los datos originales por ejemplo entra en portal de compras en línea. En primer lugar reduce los precios de los productos y las compras a la tarifa más barata.
3	Modificación de registros recientes en la base de datos	En este ataque sustituye completamente los datos existentes o borra los datos de la base de datos.
4	Acceso sobre privilegios	Una vez que el hacker obtiene éxito en el acceso en el sistema entonces para tener acceso completo sobre el sistema y la red busca la altura

	administrativos	privilegios que son utilizadas por el número de administrativo.
5	Denegación de servicio	Varias solicitudes falsas se envían al servidor como resultado se detiene temporalmente el servicio y así el usuario es capaz de acceder al sistema.
6	Acceda la información altamente sensible	Una vez que el hacker accede en la red, el atacante obtiene acceso a la información altamente sensible como número de tarjeta de crédito y otra información monetaria.
7	Destruye los datos existentes en la base de datos	Después de obtener el acceso completo sobre el sistema el atacante destruye los datos completamente, resultando en grandes pérdidas.
8	Ataques de rendimiento de máquina	El atacante detiene todas las operaciones importantes que se realiza por el sistema.
9	Modifica los datos existentes en el registro	Una vez que el atacante obtiene acceso completo sobre el sistema, modifica los datos existentes, resultando en grandes pérdidas.

Fuente. NANHAY SINGH Mohit Dayal, R.S. RAW Suresh Kumar. SQL Injection: Types, Methodology, Attack Queries and Prevention.2016. p. 2873

Ataques de inyección SQL

Estos son los principales ataques asociados con la inyección de sql:

Bypass de autenticación.

Este atacante sin proporcionar el auténtico usuario nombre y contraseña puede con éxito obtener acceso en la red mediante la manipulación de la lógica de comandos SQL.

Filtrar información sensible.

Después de ganar acceso no autorizado por la red, el atacante obtiene acceso a la información altamente sensible presente en la base de datos.

Pérdida de integridad de datos.

En atacante no sólo modifica el contenido principal, sino que también añade contenido malicioso a la base de datos comprometiéndose así con la entereza de los datos.

Pérdida de disponibilidad de datos.

Una vez que el atacante obtiene acceso completo a través del sistema, elimina los datos importantes de la base de datos dando lugar a enormes pérdidas y conduce a la indisponibilidad de datos.

Ejecución remota de código.

Permite al atacante comprometer con el sistema operativo del anfitrión. En este ataque, el atacante no sólo modifica los datos existentes, sino que también añade nuevas cuentas con derechos de usuario completos

- **Blanco.**

Un ataque de inyección SQL se presenta cuando un usuario ilegal accede de manera informal al servidor de bases de datos, logrando obtener la manipulación o visualización de datos, procesos, cuentas e información confidencial de millones de usuarios que utilizan aplicación Web para efectuar transacciones, movimientos bancarios, compras, ventas entre otros. Cuando se está ejecutando el programa vulnerable, sea en computadoras de escritorio o en sitios Web, la intrusión ocurre.

- **Resultado no autorizado.**

La técnica de ataque de inyección SQL que el hacker ejecuta son accesos no autorizados que van en incremento, es la divulgación de la información y robo de recursos de una compañía. Esta vulnerabilidad basada en Web permite al atacante falsificar la identidad, destruir, corromper y cambiar los datos presentes en el sistema.

- **Objetivo.**

El objetivo de este método de vulnerabilidad de aplicaciones Web es apoderarse de información valiosa, hacer daño y obtener una ganancia financiera a partir de herramientas basadas en lenguaje SQL las cuales pueden insertar registros, modificar o eliminar datos, autorizar accesos e incluso ejecutar otro tipo de código en el computador, sin proporcionar usuario y contraseña auténtica el atacante podría obtener autenticación con éxito a través de la red sólo con la lógica de manipular comandos SQL. La confidencialidad y la integridad se ve afectada por los objetivos de dichos atacantes[7].

- **Tipo de organización.**

En el escenario de la vida actual las aplicaciones Web como sitios Web, Facebook, twiter, servidores de correo y aplicaciones financieras son organizaciones que se han convertido en el objetivo de ataque.

- **Debilidad de la aplicación.**

Reunir información: El atacante primero recoge toda la información relevante la cual es necesaria por el ataque de inyección SQL tal como se trae de la base de

datos nombre, usuario, privilegio de usuario y nivel de interacción del sistema operativo, obteniendo manejo y permisos impropios sobre archivos y entradas del sistema.

Detección de vulnerabilidades de Inyección SQL: Identifica las vulnerabilidades del sistema y lanza el ataque de inyección SQL. El atacante identifica las vulnerabilidades en los siguientes caminos:

El atacante lista todos los campos de entrada, oculta campos y envía solicitudes. Entonces el atacante inyecta códigos dentro de los campos de entrada y genera un error.

El atacante ingresa (‘), (;), (--) y Y/O en campos de entrada, si esto genera una página de error entonces este medio del sitio Web es vulnerable hacia la inyección SQL.

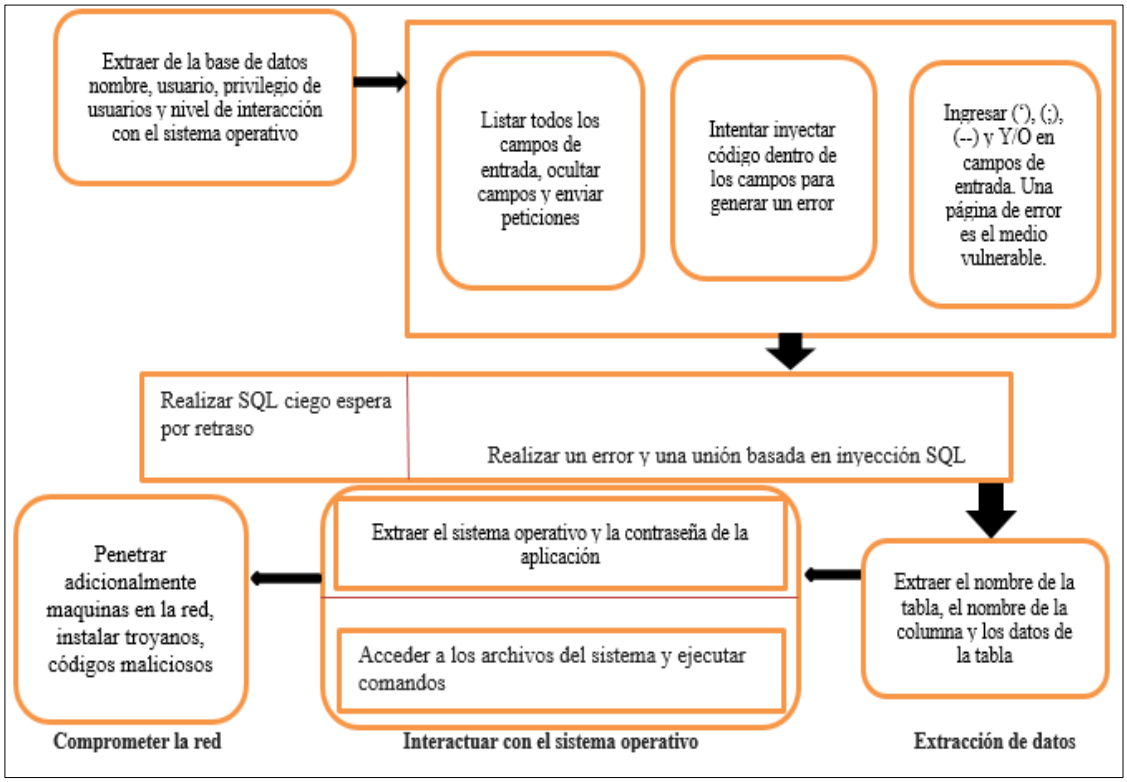
Lanzar ataque: Una de las tantas debilidades es la autenticación insuficiente, la cual explota las reglas de autenticación, el atacante ingresa dentro de la red para inyectar el código malicioso de inyección SQL.

Extracción de datos: Después de obtener el acceso competo sobre todo el sistema, el atacante obtiene acceso de la información altamente sensible presente en la base de datos, haciendo un manejo impropio de las salidas.

Interacción con el sistema operativo: Una vez el hacker obtiene éxito en el acceso al sistema ya tiene el control sobre los privilegios de administrador.

Comprometer el sistema: Permite al atacante comprometer el host de sistema operativo. En este ataque, el atacante modifica los datos existentes y adiciona nuevas cuentas como súper usuarios dependiendo de la intención (véase Figura 10)[7].

Figura 10. Inyección SQL paso a paso



Fuente. NANHAY SINGH Mohit Dayal, R.S. RAW Suresh Kumar. SQL Injection: Types, Methodology, Attack Queries and Prevention.2016. p. 2874

La funcionalidad vulnerable es otra debilidad que aprovecha el atacante para la explotación de este método de ataque. A continuación (véase la Tabla 5), se evidencia herramientas para cumplir con el ataque.

Tabla 5. Herramientas inyección SQL

Herramienta	Descripción
Inyección SQL ciega	Permite a los atacantes explotar la vulnerabilidad de inyección SQL sobre alguna base de datos virtual. Soporta la inyección SQL ciega, inyección SQL profunda e inyección SQL basada en errores. Es rápida, soporta múltiples hilos y el modo GUI.
Herramienta maratón	Esta es usada para ganar tiempo basado en ataques de inyección SQL. Tiene configuración flexible para inyecciones, adiciona variables y valores en cookies.
Inyección de energía SQL	Esta herramienta es usada para explotar la vulnerabilidad SQL sobre una página Web. Soporta la inyección SQL ciega y el servidor SQL, Oracle, MySQL, Sybase / servidor adaptativo, pero cuando se utiliza en modo normal, soporta cualquier base de datos existente.
Havji	Es una herramienta automática de inyección SQL que permite al atacante explotar la vulnerabilidad SQL en una página Web mediante la realización de huellas dactilares de backend, la recuperación de usuarios, hashes de contraseña y la ejecución de comandos en el sistema operativo.

Fuente. NANHAY SINGH Mohit Dayal, R.S. RAW Suresh Kumar. SQL Injection: Types, Methodology, Attack Queries and Prevention.2016. p. 2874

PÉRDIDA DE AUTENTICACIÓN Y ADMINISTRACIÓN DE SESIÓN

- **Atacante.**

La búsqueda de vulnerabilidades en el método de autenticación o administración de sesión puede ser llevada a cabo por todo tipo de usuarios ya sean externos o internos. El ataque de secuestro de sesión es un tipo de ataque donde el atacante roba el token de sesión de un usuario autorizado para ir realizando acciones adversarias. En el ataque de fijación de sesión, el atacante eleva su token de sesión a una persona autorizada, para robar la sesión del usuario.

El ataque de CSRF permite al atacante enviar una solicitud maliciosa a solicitud en nombre de un usuario legítimo. El ataque Clickjacking tienta a un usuario a hacer clic en los objetos colocados en páginas Web maliciosas, lo que puede dar lugar a una acción sin el consentimiento del usuario. El secuestro de sesión y la fijación de sesión ataca el objetivo en el ID de sesión del usuario, mientras que CSRF y clickjacking apuntan al navegador para presentar solicitudes ilegítimas en nombre del usuario [28].

HTTP, cookies, pequeños archivos de datos que mantienen información de estado de sesión en el navegador, fueron diseñados para hacer frente a esta limitación y rápidamente se convirtió en el mecanismo dominante para la gestión de sesiones HTTP. Aunque las cookies son un mecanismo práctico y eficaz para la gestión de sesiones, introducen una serie de riesgos de seguridad, especialmente cuando se emplea como autenticación de sesión. Por ejemplo, la mayoría de las aplicaciones Web se basan en la seguridad que es proporcionada por HTTPS para proteger la contraseña del usuario durante el proceso de inicio de sesión. Durante este paso, la Web genera cookies que el usuario puede emplear más tarde en la sesión.

En consecuencia, las cookies están expuestos a cualquier adversario o espionaje en la comunicación. Dado que las cookies son estáticas, un adversario puede usarlos para obtener acceso no autorizado a la sesión. Varios factores, tales como la proliferación, redes inalámbricas abiertas y de la liberación de herramientas de ataque automatizadas han aumentado el riesgo de esta amenaza. La mayor parte se recomienda utilizar HTTPS para proteger a todas las comunicaciones con la Web. Sin embargo, la implementación activa HTTPS puede ser un reto debido a su rendimiento y costo.

- **Herramienta.**

La fijación de la sesión y la falsificación de solicitud de sitio cruzado (CSRF) son los principales ataques encontrados en los mecanismos de administración de sesiones en las aplicaciones Web. En un ataque de fijación de sesión en el mecanismo de administración de sesión de una aplicación Web, el atacante fija el ID de sesión del usuario antes de que el usuario inicie sesión en el servidor de destino, eliminando la necesidad de obtener el ID de sesión del usuario después. Los pasos involucrados en un ataque de fijación de sesión en un servidor de aplicaciones vulnerable son los siguientes:

En primer lugar, el atacante envía una solicitud GET al servidor de aplicaciones, para iniciar una secuencia de respuesta desde el servidor.
A continuación, el servidor vulnerable responde enviando una ID de sesión al cliente atacante. El atacante envía ahora un enlace malicioso a la víctima que contiene el mismo ID. de Sesión que fue enviado por el servidor de aplicaciones, con el fin de aplicar el mismo Cookie de Sesión en el navegador de la víctima para el servidor de aplicaciones vulnerable.Tan pronto como la víctima cae en la trampa de ese enlace malicioso enviado por el atacante, el ID de sesión generada para el atacante se impone en el navegador de la víctima.Cuando la víctima intenta acceder a la aplicación autenticada alojada en el servidor de aplicaciones vulnerable utilizando credenciales legítimas, se autentica correctamente.
El ID de sesión original enviado al atacante, ahora se activa. El atacante ahora envía una solicitud para acceder a la página autenticada de la aplicación vulnerable. El atacante obtiene el control de las páginas autenticadas de la víctima con éxito[14].

Ahora, cuando el cliente intenta acceder a una aplicación autenticada en el servidor utilizando el ID de sesión ya comunicado por el servidor, continúa utilizando el mismo ID de sesión después de una solicitud satisfactoria, en lugar de generar un nuevo ID de sesión aleatorio [14].

Los identificadores de sesión pueden ser adivinados utilizando los algoritmos públicamente disponibles para varios marcos o lenguajes de programación, o pueden ser interceptados, utilizando por lo tanto un identificador de sesión generado previamente en el servidor de aplicaciones vulnerable. El escenario que se presenta a continuación de ataque es un modelo estricto de gestión de sesión. Cuando se envía una solicitud al servidor Web incrustado con una ID de sesión generada previamente en la cookie del cliente, responde aceptando de nuevo el mismo ID de sesión enviado.

Ahora, cuando el cliente intenta acceder a una aplicación autenticada en el servidor utilizando el ID de sesión ya comunicado con el servidor de aplicaciones de modo estricto, continúa utilizando el mismo valor de cookie incluso después de una solicitud satisfactoria, en lugar de generar una nueva cookie aleatoria[14].

- **Vulnerabilidad.**

El método de vulnerabilidad perdida de autenticación y gestión de sesión se da por factores o decisiones a nivel de diseño entre ellos se tiene:

Por no poseer una política de contraseñas fuerte, es decir, por permitir contraseñas cortas.
Mal control o gestión de los mecanismos de modificación o recuperación de las contraseñas.
Mala gestión de la finalización de sesiones, dificultades del usuario para cerrarla correctamente, tiempos prolongados de vida de la sesión tras el cierre de la aplicación.
Manejo de las contraseñas a través de la red sin cifrar, es decir, en texto plano, haciendo posible su interceptación mediante un sniffer u otro tipo de método.

- **Blanco.**

Busca obtener los datos de la cuenta del usuario, la cual es una vista simplificada de un ataque de secuestro de sesión, (1) después de inicio de sesión, la víctima envía peticiones a la aplicación Web que utiliza una cookie para la autenticación; (2) porque esta solicitud se envía a través de HTTP, un adversario puede espiar a la solicitud y capturar la cookie; (3) Por último, el adversario puede utilizar para enviar esta cookie peticiones arbitrarias a la aplicación Web, el secuestro fue con éxito en la sesión de la víctima[12].

- **Resultado no autorizado.**

El método es un acceso incrementado, es una corrupción de la información debido a que el atacante toma la información confidencial del usuario y de esa manera se pierde la autenticación ante el sistema por parte del usuario legítimo, donde se obtiene como respuesta una negación de la información y de recursos.

- **Objetivo.**

Obtener el control y la manipulación de contraseñas, claves, token de sesión, para tener acceso a identidades de usuarios, esta vulnerabilidad está relacionada con la seguridad de la información de los usuarios de los sitios Web.

- **Tipo de organización.**

Las vulnerabilidades relacionadas con la perdida de autenticación y gestión de sesión se consideran delicadas en cuanto a la seguridad de las aplicaciones y en especial de las aplicaciones Web. En la actualidad cualquier tipo de organización empresarial maneja aplicación Web y se puede ver afectada.

- **Debilidad de la aplicación.**

La administración de la sesión en sitios Web puede presentar debilidades, como es el manejo inadecuado del proceso de registro del usuario, insuficiente expiración de sesión o insuficiente recuperación de contraseña. Se puede evidenciar que existen algunas páginas seguras de acceso, pero una página de inicio de sesión HTTP puede poner todas las cuentas del usuario en riesgo. Los sitios Web enlazan sesiones a dispositivos, para eso es importante determinar tokens de sesión a dispositivos cliente para prevenir el secuestro de sesión.

Por otra parte, se tienen los procedimientos de cierre de sesión es cuando un usuario hace clic en los sitios Web de cierre de sesión donde debe invalidar la sesión activa del lado del servidor. Muchos sitios Web populares eliminan la cookie de sesión desde el navegador, pero no invalida la sesión en el servidor. En consecuencia, un atacante que de alguna manera obtiene la sesión del usuario puede seguir realizando transacciones en nombre del usuario después de cerrar la sesión[29].

SCRIPTING ENTRE SITIOS (XSS)

- **Atacante.**

Muchos cibercriminales aprovechan las vulnerabilidades en diferentes servidores, para alojar sus ataques de phishing o malware, es importante tener en cuenta que los atacantes explotan la confianza que un usuario tiene en un sitio en particular. En el caso del malware XSS, los comportamientos de los usuarios pueden caracterizarse por la tendencia a visitar los perfiles de los amigos frente a extraños, asumiendo que el perfil de un usuario siempre es accesible para todos sus amigos. Sin embargo, el perfil de una persona puede no estar disponible para todos los extraños[30].

El ataque XSS permite a un atacante vacunar a los clientes JavaScript malicioso en el código fuente de las páginas Web visitado por los usuarios benignos. Cuando este script es generado por el Navegador se ejecuta con los mismos permisos que el código benigno de secuencia de comandos. Por lo tanto, se procesa por el navegador resultante. En el robo de la información personal este se conduce a un lanzamiento exitoso del ataque XSS donde es un tipo de implementación en la capa de aplicación de la jerarquía de red.

El ataque se dirige a los scripts que se ejecutan en el lado del cliente, en lugar del servidor. Surge debido a las lagunas de seguridad. Las tecnologías avanzadas para desarrollar el sitio Web como HTML, XML, JavaScript, AJAX y así sucesivamente. es activado por la modificación de los scripts del lado del cliente

en la aplicación Web en una forma que puede ser anticipada por el atacante. Tiene impacto, su magnitud se puede medir por la presencia de la sensibilidad de la información en el lado del cliente y el robo de estos datos[15].

- **Herramienta.**

El gusano XSS es un tipo de programa malicioso que no depende de vulnerabilidades específicas del sistema. Los gusanos se esconden en los enlaces Web, y luego los códigos maliciosos se ejecutan cuando alguien hace clic en los enlaces, pronto su perfil personal será infectado por los gusanos[17].

- **Vulnerabilidad.**

La vulnerabilidad XSS es el resultado de una configuración en la falta de aplicación Web en la desinfección de las entradas de los usuarios. Estos insumos pueden ser de diferentes fuentes como campos de formularios HTML y URL. Usando estas entradas no desinfectadas, los atacantes pueden inyectar scripts maliciosos en páginas Web de aplicaciones Web. Así que XSS es un ataque de inyección de código donde los scripts se inyectan como código malicioso.

Estos scripts inyectados pueden prepararse con casi todo tipo de scripts del lado del cliente como JavaScript, VBScript y ActionScript. Los ataques XSS poseen varios riesgos graves de seguridad. Eso incluye el robo de identidad, el acceso a la información sensible o información restringida, alteración de la funcionalidad del navegador y denegación de servicio[31].

Los scripts de política permiten que se pueda acceder en las páginas procedentes del mismo sitio a los métodos y a las propiedades de cada uno, sin restricciones específicas, pero esto impide el acceso a la mayoría de los métodos y propiedades a través de las páginas de diferentes sitios. Esta política falla en la prevención del ataque XSS, el XSS es cuando el código se inyecta en la consulta HTTP que devuelve un mensaje de error personalizado que contiene el código XSS.

XSS es un tipo de vulnerabilidad de aplicaciones Web, que es causado por el fracaso de la aplicación en la comprobación en la entrada del usuario antes de devolverlo a los navegadores Web del cliente. Sin una validación adecuada, la entrada del usuario puede incluir código malicioso de secuencias de comandos que puede enviarse a otros clientes e inesperadamente ejecutadas por sus navegadores, provocando explotación de la seguridad[16].

XSS es una inyección de código que el atacante aprovecha para ejecutar secuencias de comandos maliciosas en el cliente del navegador Web. Se produce cuando una aplicación Web hace uso de los insumos suministrados por el usuario

final sin sanitización adecuada. Cuando el usuario visita una página Web explotada, el navegador ejecuta los scripts maliciosos. Esto se conoce como ataque XSS. El ataque XSS lleva a consecuencias como la sesión de secuestro, fuga de datos sensibles, robo de cookies y degradación del contenido Web. Los ataques XSS son de tres tipos (véase Tabla 6):

XSS reflejado. Se produce cuando las entradas de usuario que contienen script se refiere inmediatamente en la respuesta de la página Web Sin validación adecuada.
Almacenado. Siempre que la entrada de usuario no sea validada y contenga scripts se deben almacenar en la base de datos de la aplicación.
Basado en DOM. Se produce en el lado del cliente de la aplicación, el ataque hace que el script del lado del cliente se comporte de forma imprevista, cuando el script utiliza información no validada de la estructura DOM, modelo de objeto de documento para procesar en la aplicación[28].
XSS basado en la Mutación. Se produce al lado del cliente y se activa cuando se altera el ataque XSS basado en DOM[17]**.**

Tabla 6. Clases de ataques y su descripción

S.No	Clase de ataque XSS	Descripción	Ubicación de explotación	Nivel de Ingeniería social para explotar
1	Ataque XSS almacenado o ataque XSS persistente.	La secuencia de comandos malintencionados se transmite en la ejecución de la página Web.	Script del lado del servidor.	Bajo
2	Ataque XSS reflejado o ataque no persistente.	Son secuencias de comandos transmitidos por la Web del cliente, se refleja directamente hacia atrás o en ataque no persistente.	Script del lado del servidor.	Alto
3	Ataque XSS basado en DOM.	Se ha activado la secuencia de comandos malintencionados, mediante clientes inseguros en código fuente.	Script del lado del cliente.	Alto
4	Ataques XSS basado en la mutación.	Script malicioso activado al alterar el DOM.	Script del lado del cliente.	Alto

Fuente. GUPTA, Shashank. GUPTA, B.B. Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment. 2016. p. 1596

Este modelo de propagación de gusanos se puede representar a través de los nodos de la red los cuales se dividen en tres categorías: nodo infeccioso, nodo susceptible y nodo inmune. En realidad, los nodos no siempre están en línea, también se dividirán los nodos en cinco categorías: nodo infeccioso en línea, nodo infeccioso fuera de línea, nodo susceptible en línea, nodo susceptible fuera de

línea e nodo inmunológico. Los siguientes cinco vectores se definen para almacenar el estado de cada nodo en cualquier momento.

Bajo la condición de red real, el estado de un nodo en línea o sin conexión está cambiando dinámicamente. Para simular esta incertidumbre del estado del nodo, se genera aleatoriamente algunos nodos en línea y nodos sin conexión en cada momento. Esto significa que algunos nodos en línea se convertirán en offline, mientras que algunos nodos sin conexión se volverán a estar en línea. Aquí se establece el número de nodos en línea a α y el número de nodos fuera de línea a β. Un nodo susceptible se convertirá en nodos infecciosos cuando esté infectado por el gusano, y ambos nodos susceptibles e infecciosos pueden ser parcheados y convertirse en nodos inmunes[32].

- **Blanco.**

El atacante roba información, datos sensibles de las cuentas o procesos de los usuarios. Sin embargo, no solo ataca aplicaciones Web, sino también a aplicaciones locales y al navegador como tal.

- **Resultado no autorizado.**

Básicamente, una vez realizado el ataque la información almacenada en el servidor o los servicios comprometidos quedan expuestos, esta situación es ocasionada por accesos no autorizados, lo que permite sabotear los controles de autorización y registro de aplicación, al mismo tiempo esta vulnerabilidad puede estar presente de dos maneras directa e indirectamente una es inyectar el código donde la información se propaga a terceras personas que no son los usuarios legítimos, y por el otro lado modificar los valores en los mensajes o en la URL de la páginas Web.

- **Objetivo.**

Los ataques XSS se usan comúnmente para secuestrar sesiones de usuarios Sitios Web que facilitan el comercio electrónico, en los que el script se ejecuta en el cliente del usuario y captura la cookie de información del navegador del usuario que permite el secuestro de la sesión [33].

- **Tipo de organización.**

Los gusanos XSS se encuentran en los sitios Web más populares, especialmente debido a las redes sociales en línea OSN por sus características atractivas para el intercambio de datos. Los usuarios de correo de Yahoo han sido infectados por la

explotación de XSS basada en DOM. En consecuencia, ni los usuarios tienen que hacer clic en el malintencionado enlace o recibir mensajes de correo. Los servicios Web han sido infestados por la vulnerabilidad XSS en donde los usuarios están comprometidos con sólo una vista previa del mensaje de correo electrónico malicioso.

Twitter, el famoso sitio de OSN, contiene un defecto de XSS, que muestra una ventana emergente que exige credenciales de inicio de sesión del usuario. Orkut, popular sitio de OSN, es también atacado por el gusano XSS persistente. En esto, el atacante inyecta scripts maliciosos en su perfil y las personas, cualquiera, visita los perfiles del atacante y se infectaron, las víctimas recientes de gusanos XSS (véase Tabla 7)[15].

Tabla 7. Estadísticas de ataques XSS sobre OSN

S.No	Sitio Web Objetivo	Mes, Año	Consecuencia
1	Parlamento del Reino Unido	Marzo, 2014	Desinformación
2	Yahoo	Enero, 2013	Cuenta Hijacking
3	Hotmail	Junio, 2011	Sesión Hijacking
4	Twitter	Septiembre, 2010	Gusano
5	OrKut	Diciembre, 2007	Gusano

Fuente. CHAUDHARY, Pooja. GUPTA, B.B. GUPTA, Shashank. Cross-Site Scripting (XSS) Worms in Online Social Network (OSN): Taxonomy and Defensive Mechanisms. 2016.p.2132

- **Debilidad de la aplicación.**

Se presenta por funcionalidades vulnerables, permisos impropios sobre archivos del sistema, autenticación insuficiente y procesos de validación insuficientes. Si los atacantes logran conducir con éxito el ataque XSS, en la navegación Web de usuarios finales la sesión puede ser secuestrada y su información personal será a menudo robada por los atacantes. La detección de XSS ha sido importante en los últimos años, porque es un ataque provocado por las secuencias de comandos maliciosas en los sitios Web [34].

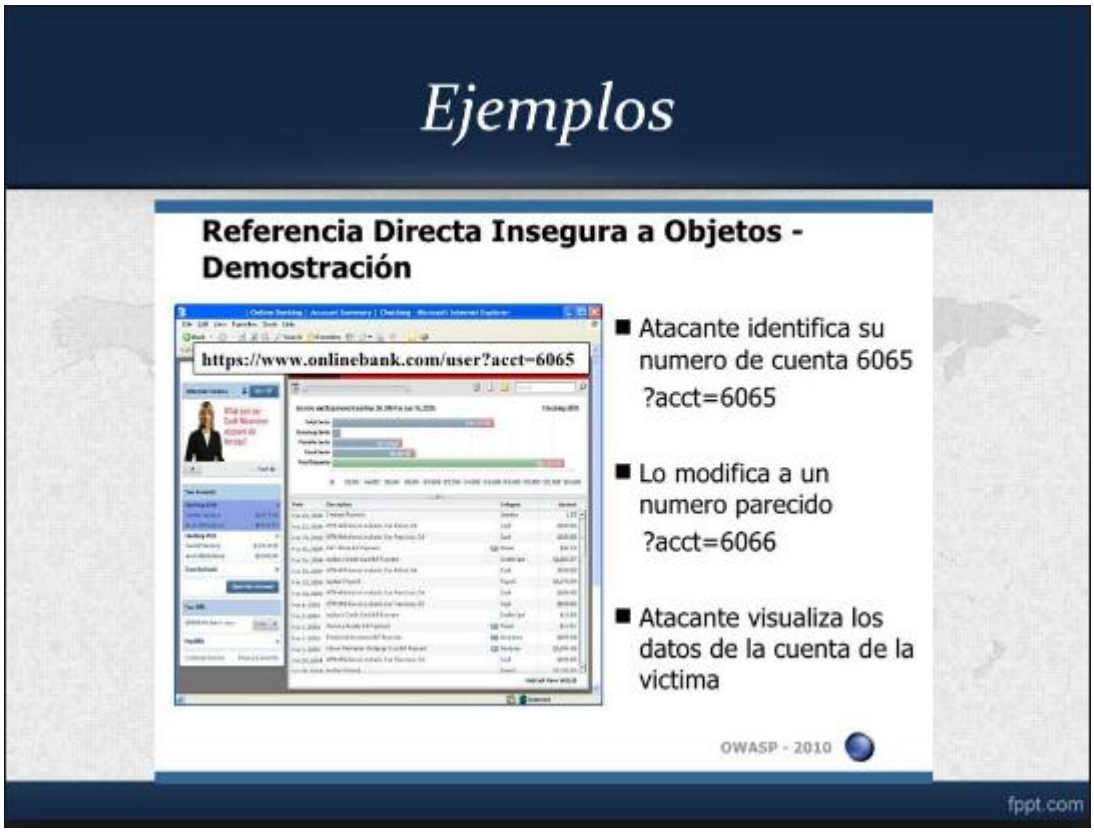
REFERENCIAS DIRECTAS A OBJETOS INSEGUROS

- **Atacante.**

Referencias a objetos inseguros es un problema de autorización, que resulta fácil de explotar. Como resultado el usuario o atacante es capaz de acceder a un objeto, fichero o base de datos del sistema al que no está autorizado. Su impacto es grave dejando la explosión de datos privados hacia el resto de usuarios de la

aplicación, el atacante como usuario autorizado en el sistema, modifica los valores de los parámetros que se refiere directamente a un objeto para que el usuario no se encuentra autorizado (véase Figura 11)[35].

Figura 11. Ataque referencia insegura a objetos



Fuente. http://slideplayer.es/slide/5966797/.6 de mayo 2017

- **Herramienta.**

Una referencia directa a objetos ocurre cuando el desarrollador expone una referencia a un objeto de implementación interno. Por lo tanto, sin un chequeo de control de acceso u otra petición, los atacantes pueden manipular dichas referencias para acceder.

- **Vulnerabilidad.**

La mayoría de los problemas de seguridad en los sitios Web se encuentran a nivel aplicación y que son el resultado de escritura defectuosa de código, se debe entender que programar aplicaciones Web seguras no es una tarea fácil, ya que requiere por parte del programador, no únicamente mostrar atención en cumplir

con el objetivo funcional básico de la aplicación, sino una concepción general de los riesgos que puede correr la información contenida, solicitada y recibida por el sistema. En la actualidad, aunque existen muchas publicaciones que permiten formar un criterio sobre el tema, no existen acuerdos básicos sobre lo que se debe o no se debe hacer, y lo que en algunas publicaciones se recomienda, en otras es atacado[36].

- **Blanco.**

El atacante se aprovecha de todos aquellos objetos que no tienen las protecciones apropiadas y verifica que usuario no está autorizado a acceder al recurso y de ese modo utiliza la cuenta de dicho usuario para comprometer la seguridad del sistema.

- **Resultado no autorizado.**

A pesar de la existencia de patrones de seguridad, protección de datos, resulta para el atacante una explotación fácil accediendo a todos los datos el sistema. Normalmente las aplicaciones utilizan el nombre o clave actual de un objeto cuando se generan las páginas Web y no verifican que el usuario tenga autorización sobre el objeto.

- **Objetivo.**

Realizar referencia a objetos o elementos internos de la aplicación para los que no se tiene autorización, los cuales utilizan la aplicación para almacenar información y que son referenciados a través de parámetros en URL’s o en formularios.

- **Tipo de organización.**

En cualquier caso, este método de vulnerabilidad puede afectar el funcionamiento interno de las aplicaciones de cualquier organización. Las empresas por esta razón, gestionan tratamientos de prevención en busca de mitigar el riesgo de posibles accesos no autorizados que comprometen la integridad de los datos confidenciales.

- **Debilidad de la aplicación.**

Se considera algunas razones que logran debilitar las aplicaciones y hacer este método de vulnerabilidad efectivo, entre ellos están:

La falta de comunicación cifrada en el proceso de acceso a la aplicación.

La utilización de mecanismos de autenticación del tipo “Recordar contraseña”, dado que esta contraseña se almacena para poder ser utilizada y la sustracción de este valor ocasiona la suplantación de usuarios.
La no existencia de un enlace en las páginas de la aplicación para que el usuario pueda cerrar la sesión.
La mala configuración de caducidad de las sesiones ante un periodo de inactividad.

CONFIGURACIÓN ERRÓNEA DE SEGURIDAD

- **Atacante.**

Un atacante atento, puede modificar en PHP, Apache el archivo httpd.conf para administrar configuración global del servidor y también permite método de control de acceso por directorio a través de su archivo .htaccess. Hay directivas que se pueden establecer en httpd.conf o php.ini. En este caso, el valor que se utilizará en tiempo de ejecución depende de si Apache está configurado para permitir o no anular las configuraciones globales. Uno de los muchos ejemplos en httpd.conf que se puede controlar es o no revelar la firma del servidor a través de la directiva de firma del servidor.

Las incidencias causadas por errores de configuración son crecientes. Para lanzar XSS, inyección de SQL y ataques de cruce de directorios, hay muchas razones asociadas con la mala configuración de vulnerabilidades y sus consecuencias en la Web. La primera razón es que Apache, MySQL y PHP (AMP) es sin duda la aplicación Web más utilizada en ambiente de servidor, ya que estos componentes son de fuente abierta, hacen una mezcla suave que aprovecha la facilidad de uso, flexibilidad y portabilidad a través de múltiples plataformas[21].

- **Herramienta.**

El protocolo SSL proporciona autenticación basada en infraestructura de clave pública X.509, protege los datos. La codificación simétrica asegura la integridad de los datos con mensajes digestivos criptográficos. SSL se utiliza comúnmente para proteger sitios Web y servidores de correo, prevenir ataques de red de escuchar o repetir los mensajes del cliente y se considera generalmente la mejor práctica de seguridad para los sitios que utilizan el protocolo HTTPS3.

La validación del Certificado de Navegador básicamente consta de los siguientes pasos: comprobar si el certificado firmado digitalmente por un emisor es de confianza y comprobar si el certificado no se ha cumplido. Los mensajes de advertencia SSL en caso de fallar la validación del certificado X.509, son mostrados por los navegadores modernos. Este mensaje de advertencia podría

implicar que se produjo un ataque donde el atacante intercambió el certificado original del servidor con su propia cuenta, para escuchar al Cliente y servidor[37].

- **Vulnerabilidad.**

La categoría de vulnerabilidades de configuración errónea se puede presentar por malas reacciones del sistema: Cuando ocurre una mala configuración, el sistema debe identificar los parámetros del parámetro mal configurado Nombre / valor o su información de ubicación, por ejemplo, los números de línea en el archivo. De lo contrario se considera que la reacción del sistema como una vulnerabilidad de configuración errónea.

La Tabla 8 clasifica los diferentes tipos de configuración errónea. La primera categoría, el sistema se bloquea y se cuelga, se considera como vulnerabilidades severas, especialmente para aplicaciones de servidor[38].

Tabla 8. La categoría de las reacciones del sistema malo

Reacción	Descripción
Choque o colgar	El sistema choca o cuelga.
Terminación anticipada	El sistema sin identificar el error de configuración inyectado.
Fracaso funcional	El sistema falla la prueba funcional sin identificar el error inyectado.
Violación silenciosa	El sistema cambia las configuraciones de entrada a diferentes valores sin notificar a los usuarios.
Ignorancia silenciosa	El sistema ignora las configuraciones de entrada (principalmente por control-violación de dependencia).

Fuente. XU, Tianyin. ZHANG, Jiaqi. HUANG, Peng. ZHENG, Jing. SHENG, Tianwei. YUAN, Ding. ZHOU, Yuanyuan. PASUPATHY, Shankar. Do not blame users for misconfigurations. 2013.p.251

Políticas de seguridad

El control de acceso es omnipresente dentro de las diferentes capas de TI. Por ejemplo, el filtrado de red y capa de aplicación en las políticas de autorización son diferentes formas de control de acceso que normalmente cooperan en escenarios reales. El proceso de control de acceso distribuye a través de varios componentes de TI, potencialmente operando en diferentes capas de arquitectura y residiendo en diferentes hosts. Por ejemplo, un clásico firewall de red es capaz de tomar decisiones de permitir / denegar solicitudes de red, con parámetros como direcciones IP y puertos TCP / UDP.

Intuitivamente, un firewall en particular, dependiendo de su política, permite que las solicitudes sean procesadas u otros puntos de decisión política bloquearan las decisiones de inmediato[39].

Accesos que no están permitidos por la política implementada pero que comparten similitudes con accesos permitidos, pueden ser indicativo de las configuraciones erróneas de la política de control de acceso. Identificando tales configuraciones incorrectas permite a los administradores resolverlas antes que interfieran con el uso del sistema[40].

Un atacante puede aprovecharse de configuraciones erróneas como software no remendado, funciones adicionales innecesarias como la consola administrativa del servidor de aplicaciones o cuentas predeterminadas activadas, las aplicaciones se implementan utilizando una pila de aplicaciones complejas: servidores Web, servidores de aplicaciones, servidores de bases de datos, componentes de terceros y código personalizado. En muchos casos hay múltiples equipos que son responsables de la seguridad de los diferentes componentes: administradores de bases de datos, administradores de sistemas, equipos de desarrollo.

En este entorno, es fácil que ocurran errores de configuración de seguridad en cualquier nivel de la pila de aplicaciones. Para realizar algún tipo de mal uso es necesario identificar alguna vulnerabilidad dentro de la pila de aplicaciones. Estos ataques pueden realizarse aprovechando las siguientes vulnerabilidades:

- Las contraseñas predeterminadas o débiles.
- Una pila de aplicaciones puede contener software sin revisión. El código de explotación de la vulnerabilidad podría estar disponible públicamente, ser desarrollado.
- Existen varios diccionarios como CVE que publican vulnerabilidades y vulnerabilidades relacionadas, como vulnerabilidades en versiones específicas de bases de datos y marcos. Pueden incluir configuración errónea.
- El surgimiento de mercados de vulnerabilidad proporciona un incentivo económico para que los investigadores busquen y divulguen información sobre vulnerabilidades. Éstos incluyen revelaciones de vulnerabilidad relacionadas con configuraciones erróneas de seguridad.
- La existencia de programas que proporciona un incentivo económico para buscar y revelar información sobre vulnerabilidades. Además, la divulgación de la vulnerabilidad es vista como un símbolo de estado por muchos expertos en seguridad.

- **Blanco.**

La explotación de esta vulnerabilidad permite el acceso no autorizado a datos del sistema o funcionalidades de la aplicación, y afecta a los niveles de la capa de aplicación, desde el código hasta el marco de trabajo. Ocasionando daños en el servidor Web, servidor de aplicaciones y base de datos.

- **Resultado no autorizado.**

La mayoría de vulnerabilidades están relacionadas con errores de configuración en el servidor, versiones de software desactualizadas, ajustes inapropiados y problemas que se presentan con un despliegue inseguro, lo que da como resultado corrupción, difusión y negación de la información.

- **Objetivo.**

Tomar el control de los datos más relevantes del conjunto de componentes del sistema a partir de sesiones que no son autorizadas y aprovechar la falta de políticas de seguridad por parte de los roles implicados en el desarrollo.

- **Tipo de organización.**

El método de configuración errónea se puede presentar en organizaciones de negocios, servicios financieros, instituciones educativas entre otros, tanto para servicios Web, como para aplicaciones locales.

- **Debilidad de la aplicación.**

Un atacante puede obtener el control del sistema comprometido para obtener información confidencial, credenciales o información financiera. Un atacante podría ser capaz de deshabilitar o eliminar la aplicación utilizando la consola administrativa que conduce a una condición de denegación de servicio. El atacante podría ser capaz de expandir el ataque a otros sistemas y obtener información más confidencial. El atacante podría ser capaz de realizar modificaciones de la pila de aplicaciones para obtener ganancias financieras.

Las posibles fuentes de fallas para el atacante incluyen, posibilidades de que el código de explotación no funcione, es posible que el sistema infectado esté suficientemente protegido para evitar que el atacante acceda a otros sistemas. Sin embargo, en este caso la aplicación misma podría estar comprometida.

La configuración errónea de seguridad puede ser detenida por las siguientes contramedidas:

Proporcionar un proceso de endurecimiento repetible, por ejemplo, eliminar cuentas predeterminadas y de prueba para bloquear la configuración del sistema.
Aplicar actualizaciones de software y parches de manera oportuna.
Proporcionar un sistema para supervisar la seguridad de todos los componentes de la pila de aplicaciones en bases de datos públicas, listas de correo y listas de correo de seguridad, y mantenerlos actualizados. Exigir exploraciones y pruebas de seguridad de toda la pila de aplicaciones. Utilizar patrones de seguridad en la

pila de aplicaciones. Diseñar aplicaciones usando metodologías de seguridad apropiada[22].

EXPOSICIÓN DE DATOS SENSIBLES

- **Atacante.**

Las aplicaciones Web no protegen de manera adecuada los datos sensibles como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar esos datos y cometer fraudes, robos de identidad y muchos delitos más.

La exposición de datos sensibles en almacenamiento y transmisión plantea una grave amenaza para la seguridad organizativa. La detección de fugas de datos apunta a escanear contenido en edad o transmisión de los datos sensibles expuestos. Porque el volumen del conjunto de datos necesita ser escalables para una detección oportuna. Esto permite la preservación de la técnica para minimizar la exposición de los datos sensibles. Esta transformación apoya la salida segura de la fuente de detección de fugas de datos no confiable [41].

- **Herramienta.**

Esta vulnerabilidad logra acceder al sistema y tomar los datos, hacen intercambio de información o modificación de datos privados.

- **Vulnerabilidad.**

Existen tres posibles causas para que los datos sensibles aparezcan en el tráfico de salida de una organización, incluido el legítimo uso de datos por parte de los empleados.

• **Caso 1 Fuga de datos inadvertida:** Los datos confidenciales se filtran accidentalmente en el tráfico saliente por un usuario legítimo. Este trabajo se centra en detectar este tipo de fugas de datos accidentales sobre la red supervisada. La fuga inadvertida de datos puede deberse a errores como el olvido de usar el cifrado, el envío descuidado, un correo electrónico interno y adjuntos a personas externas o debido a defectos de aplicación.

• **Caso 2 Fuga de datos malintencionados:** Un intruso o una pieza de software furtivo puede robar información personal u organizativa. Porque el adversario malicioso puede utilizar encriptación privada fuerte, estenografía o encubierta, canales para desactivar la inspección de tráfico basada en contenido,

• **Caso 3 Transferencia de datos legítima y prevista:** Los datos confidenciales son enviados por un usuario legítimo[24].

- **Blanco.**

El atacante se apodera de las cuentas privadas, datos o cualquier otro proceso confidencial de los usuarios.

- **Resultado no autorizado.**

Se considera quien puede obtener acceso a los datos, pero no siempre se cumplen las políticas de seguridad y los accesos no autorizados incrementan, esto representa un conjunto de datos expuestos, maltratados y un robo de recueros en el sistema.

- **Objetivo.**

Acceder a los datos sensibles desde el momento que son facilitados por el usuario, enviados y almacenados por la aplicación, hasta que son devueltos por el navegador.

- **Tipo de organización.**

Bancos que son entidades que manejan aplicaciones Web que guardan números de tarjetas de créditos o datos que se consideran confidenciales o cualquier entidad financiera. Los sitios expuestos a esta vulnerabilidad son aquellos que no utilizan SSL para sus páginas que requieren autenticación.

- **Debilidad de la aplicación.**

La debilidad más común es no cifrar datos sensibles. Cuando se hace cifrado, se detecta que hay claves, algoritmos y técnicas débiles de hashing. El navegador también es débil y son fáciles de detectar, pero son difíciles de explotar.

FALTA DE CONTROL DE ACCESO DE NIVEL DE FUNCIONES

- **Atacante.**

El atacante, que puede ser un usuario legitimo dentro del sistema, lo que hace es coger la URL y modificarla o un parámetro a una función con privilegios. La explotación a esta vulnerabilidad permite el acceso no autorizado a determinadas funcionalidades ofrecidas por la aplicación. Las cuentas con permisos de administración resultan ser las más abusadas.

- **Herramienta.**

En el proceso de autenticación el atacante puede lograr ingresar datos o comando de usuario que controla a los usuarios en la forma en que se conectan a las aplicaciones. Eso debe ser una estructura sólida para la gestión, auditoría y encriptación de la autenticación de datos. La autenticación puede identificar dos tipos de roles en este análisis: Roles de negocios, que determinan la ubicación del usuario dentro de la Jerarquía institucional y los roles de aplicación definidos.

La autorización es a través de la gestión, las funciones procurarán que cada usuario reciba autorizaciones para el rol de negocio y las funciones se desempeñan dentro de la institución, mitigando vulnerabilidades y riesgos para proporcionar confiabilidad, integridad y disponibilidad de información. Definirá y aplicará seguridad, políticas de gestión de la identidad basadas en procesos en el nivel estratégico de acceso.

La auditoría en la gestión del acceso será la referencia al control, monitorear, auditar recursos a través de servicios de intranet o extranet de la institución. El proceso se basa en la definición de las políticas de seguridad que se adoptarán para mecanismos de autenticación, autorización y auditoría. La creación de políticas en el control de acceso responsabiliza a los usuarios sobre la gestión[42].

El acuerdo de claves de sesión no se debe admitir, en algunos casos, es posible que después del éxito de autenticación mutua, haya algunos mensajes secretos de comunicación entre el usuario y el servidor, que deben codificarse para proporcionar la confidencialidad de los datos transmitidos, por lo que es una característica de autenticación que puede hacer que el usuario y el servidor negocien una clave de sesión acordada para asegurar la confidencialidad[43].

- **Vulnerabilidad.**

Este método puede ser vulnerable por el tipo de implementación de controles de acceso en el código, por la falta de auditoría y actualización de los procesos de gestión de accesos y permisos, dicha implementación de controles debería negar

el acceso por defecto, y debe requerir los permisos a roles específicos para acceder a las funcionalidades.

- **Blanco.**

El atacante va en busca de obtener los datos de las cuentas más sensibles del sistema a través de una red o aplicaciones locales.

- **Resultado no autorizado.**

Se debe a accesos de atacantes al sistema, que manipulan la información y roban los recursos importantes en el funcionamiento del sistema.

- **Objetivo.**

Acceder al sistema de manera no autorizada, y hacer uso de todo tipo de funciones principalmente de las administrativas.

- **Tipo de organización.**

Las organizaciones a diario se ven afectadas por estos escenarios de ataque, porque el atacante fuerza la navegación hacia las URL'S objetivo y otros sitios solo pueden acceder los administradores, pero si el usuario puede entrar a ambas direcciones se considera una vulnerabilidad.

- **Debilidad de la aplicación.**

Los permisos impropios del sistema, la falta de procesos de validación insuficiente y la falta de configuración en la aplicación hacen posible el cumplimiento de este método de vulnerabilidad.

CROSS-SITE REQUEST FORGERY (CSRF)

- **Atacante.**

El ataque de solicitud de falsificación de sitio cruzado (CSRF) manipula la sesión HTTP del usuario o el privilegio de inicio de sesión para la aplicación Web específica. El atacante engaña a la víctima en ejecución inadvertida de script malicioso. Para implementar un ataque CSRF, el atacante necesita tener una idea clara de la petición HTTP y el patrón de respuesta de la Web seleccionada. Utilizando esta información, el atacante crea una página Web que genera automáticamente solicitudes falsas HTTP que son idénticas a las solicitudes HTTP

válidas de la víctima. El atacante utiliza el identificador de sesión entre el usuario víctima y la aplicación Web para hacer la solicitud. El atacante utiliza el identificador de sesión entre el usuario víctima y la aplicación Web para hacer la solicitud.

En el ejemplo, el atacante pretende cambiar el perfil del usuario en la información de la aplicación Web de destino. Para hacer eso, el atacante escribe un script que intentará enviar automáticamente un formulario de perfil de usuario a la aplicación Web de destino cada vez. Alguien navega por la página Web falsificado este formulario de perfil de usuario donde debe ser idéntico al usuario de la aplicación Web objetivo de la víctima, el formulario de perfil. El código de script debe incluir el mismo HTTP de respuesta de solicitud utilizado por la aplicación Web de destino para comunicarse con los usuarios.

Para que este ataque funcione, el usuario víctima necesita iniciar sesión en la aplicación Web de destino, los atacantes engañan al usuario víctima para que visite una aplicación Web forjada. Una vez que la víctima visita la página falsa, el script realiza un envío automático, solicitud HTTP al navegador del usuario víctima. Como la solicitud del formulario es idéntica a la forma de aplicación Web de destino, el navegador envía el formulario a la aplicación Web de destino.

El servidor de aplicaciones Web coincide con el id de sesión de la solicitud y los identifica como válidos desde que la solicitud fue enviada desde el navegador de la víctima. El servidor de aplicaciones Web ejecuta la solicitud de formulario malicioso y cambia los perfiles de las víctimas.

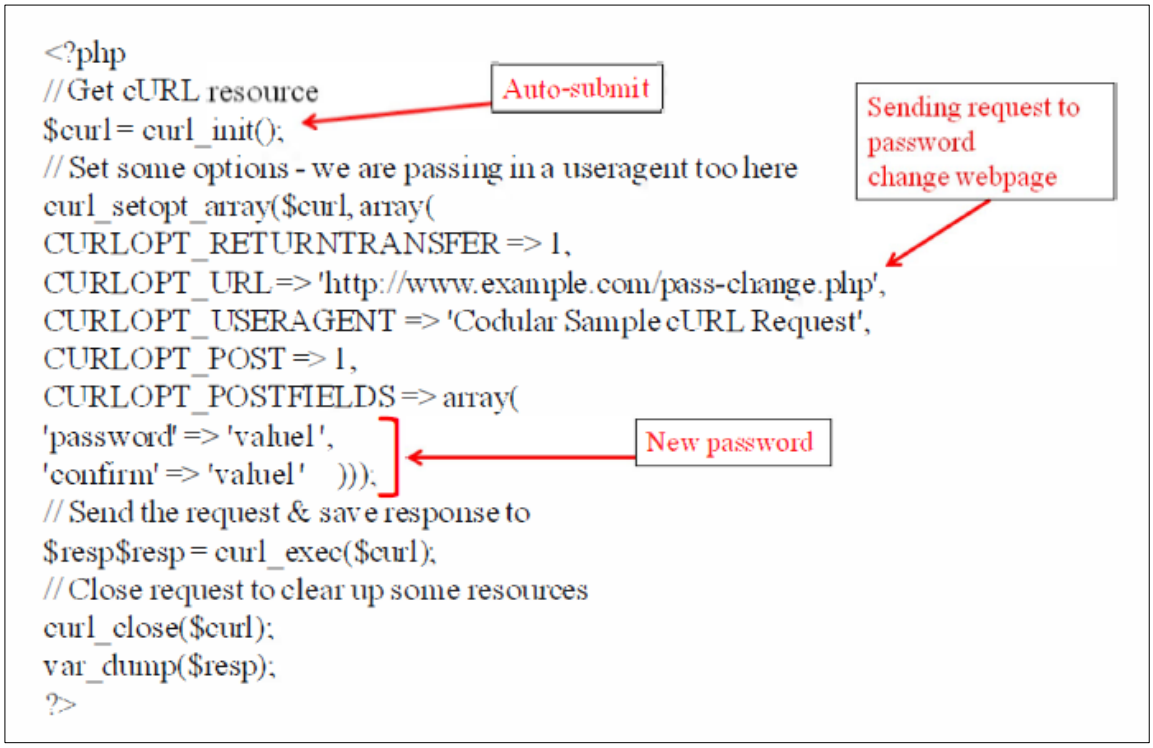
- **Herramienta.**

El principal método para atacar a la CSRF es engañar al usuario víctima para visitar la página Web forjada de CSRF. Una vez que la víctima visita la página falsa el resto del código podría ser el mismo que el script de ataque XSS. Como esta solicitud es recibida por la Web el servidor de aplicaciones procesa porque el servidor Web es engañado a creer que la solicitud es enviada por la víctima usuario ya que tienen el mismo id de sesión[44].

- **Vulnerabilidad.**

Se puede evidenciar vulnerabilidades de tipo diseño, implementación, configuración o falta de políticas en las organizaciones. A continuación, se muestra un Script que hace cambio de contraseñas a través de este ataque (véase Figura 12)[45].

Figura 12. Script para cambiar la contraseña a través de CSRF




```
<?php
// Get cURL resource
$curl = curl_init();
// Set some options - we are passing in a useragent too here
curl_setopt_array($curl, array(
CURLOPT_RETURNTRANSFER => 1,
CURLOPT_URL => 'http://www.example.com/pass-change.php',
CURLOPT_USERAGENT => 'Codular Sample cURL Request',
CURLOPT_POST => 1,
CURLOPT_POSTFIELDS => array(
'password' => 'valuel',
'confirm' => 'valuel'    ));
// Send the request & save response to
$resp$resp = curl_exec($curl);
// Close request to clear up some resources
curl_close($curl);
var_dump($resp);
?>
```

Fuente. FARAH Tanjila, SHOJOL Moniruzzaman, HASSAN Maruf, ALAM Delwar. Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS & CSRF. 2016.p.77

- **Blanco.**

Consiste en enviar una solicitud malintencionada a sitios Web, desde el cliente autenticado desde el servidor.

- **Resultado no autorizado.**

El método CSRF es una vulnerabilidad, como en la mayoría de casos de amenazas, son no autorizadas en el sistema, acceden de manera ilegal, toman los recursos del servidor y la base de datos.

- **Objetivo.**

Borrar, realizar transacciones o cambiar contraseñas. Este ataque es exitoso debido al hecho de que los desarrolladores tienden a confiar que un cliente nunca enviará una solicitud, esto no implica que la interfaz gráfica de usuario no este

diseñada para enviar. A diferencia del ataque XSS, que explota la confianza del cliente en el sitio Web, este ataque explota la confianza del sitio Web en el cliente [46].

- **Tipo de organización.**

Las redes sociales son gran objeto para este tipo de ataque, lo que hace el usuario malintencionado es inyectar código en etiquetas de imágenes para controlar el sitio.

- **Debilidad de la aplicación.**

Solicitud de sitio cruzado de falsificación es también una amenaza de seguridad mayor consistente en enviar una solicitud malintencionada a un sitio Web, normalmente desde un cliente autenticado del servidor, la solicitud maliciosa podría incluir la realización de datos. La causa principal de los CSRF es la prevalencia de la autoridad ambiental en la Web de hoy. La autoridad ambiental significa que los sitios Web envían datos automáticamente por los navegadores como una indicación de autoridad y por lo tanto intención legítima del usuario [47].

USO DE COMPONENTES CON VULNERABILIDADES CONOCIDAS

- **Atacante.**

El uso de componentes con vulnerabilidades conocidas se presenta cuando los desarrolladores no conocen las versiones de las librerías y trabajan con código desactualizado, afectando otros componentes con los que existe dependencia, de esa manera se heredan los problemas de seguridad ya existentes de otro software El trabajo de un hacker es fácil porque logra identificar si hay componentes vulnerables para explotar con herramientas automatizadas.

- **Herramienta.**

El uso de componentes con vulnerabilidades conocidas logra ser la herramienta clave de los atacantes, donde pueden invocar servicios Web con todos los permisos al no proporcionar un token de identidad y la ejecución de código remoto en aplicaciones java con vulnerabilidad de inyección.

- **Vulnerabilidad.**

Es una vulnerabilidad o amenaza de tipo diseño, implementación y configuración, que no cumple con todas las políticas de seguridad. En desarrollo de software los

Ingenieros se enfrentan con el problema de comprender y tomar el código fuente de otros desarrolladores. El desafío clave es entender la especificación subyacente implementada por el sistema de software. Recuperar esta comprensión es más difícil cuando el código fuente es la única fuente confiable de información, la documentación está obsoleta o sólo está presente en fragmentos, y los desarrolladores originales ya no están disponibles.

Desafortunadamente, se encuentran frecuentemente situaciones de este tipo en sistemas de software científico y de ingeniería, desarrollados en la industria. Si un Ingeniero deja la empresa, literalmente deja atrás un sistema heredado para otra persona (o equipo). U soporte de herramientas que combina análisis de programas estáticos y dinámicos puede abordar este desafío. Usando el análisis estático del programa se extrae el comportamiento de la entrada / salida del código fuente del programa y se presenta la información extraída además del código fuente analizado, proporcionando la navegación inconsútil entre ambas vistas.

El análisis dinámico del programa permite a los desarrolladores examinar el comportamiento de entrada / salida para las ejecuciones de un solo programa y así obtener información sobre el comportamiento estándar y casos excepcionales[48].

- **Blanco.**

Es una vulnerabilidad donde el atacante conoce componentes software débiles de una aplicación Web como pueden ser los frameworks, librerías y otros módulos de software, que en muchos casos funcionan con todos los privilegios.

- **Resultado no autorizado.**

Debido al tema de que los desarrolladores utilizan componentes vulnerables, se puede presentar una corrupción de la información o funcionalidad del software. La comprensión del programa es un proceso vital que implica mucho esfuerzo en el mantenimiento del software. Un desafío clave para los desarrolladores es comprender un sistema de software para ser mantenido ya que es difícil y consume mucho tiempo. Hoy en día, los sistemas de software han crecido en tamaño causando el aumento de tareas de los desarrolladores en la exploración y comprensión del código fuente. El código fuente es un recurso crucial para que los desarrolladores se familiaricen con un sistema de software ya que alguna documentación del sistema es a menudo no disponible o anticuado[49].

- **Objetivo.**

Atacar un componente vulnerable que facilite pérdida de datos y una toma del control del servidor.

- **Tipo de organización.**

Todas las empresas que manejen software se ven expuestas a este método de vulnerabilidad, por tal motivo es importante tener en cuenta medidas de prevención.

- **Debilidad de la aplicación.**

En su mayoría de casos los Ingenieros de software deben analizar todo el documento de requisitos y buscar los requisitos que se ven afectados por un cambio, en consecuencia, los Ingenieros suelen aplicar cambios a la implementación directamente y dejar los requisitos sin cambios. Es importante mantener las especificaciones actualizadas cuando los sistemas evolucionan y un mantenimiento de las especificaciones de los requisitos de forma significativa[50].

REDIRECCIONAMIENTO Y REENVÍO NO VALIDADO

- **Atacante.**

Esta vulnerabilidad puede ser explotada cuando el uso de redirecciones no validadas, están accediendo a un recurso perteneciente a un dominio que el usuario confía. Este método de ataque es aprovechado por agentes atacantes que instalan malware o hacen creer al sistema que son usuarios legítimos para obtener información sensible.

- **Herramienta.**

El re direccionamiento de URL es necesario en las aplicaciones Web. Sin embargo, si se utiliza incorrectamente, podría dar lugar a ataques como el phishing. Una técnica de escaneo de caja negra puede ser de gran ayuda para modificar URL y analizar la salida generada para identificar vulnerabilidades en los principales sitios Web[51].

- **Vulnerabilidad.**

Es una vulnerabilidad de tipo diseño, implementación y configuración que deja de cumplir con ciertas políticas de seguridad.

- **Blanco.**

El atacante phisher engaña a la víctima para acceder a sitios no autorizados a partir de peticiones falsas en las aplicaciones Web, ellos buscan ganarse la confianza de los usuarios.

- **Resultado no autorizado.**

Es una amenaza o vulnerabilidad que no tiene ningún acceso permitido al sistema, y por el contrario lo que hace es corrupción de la información, negación y difusión de la misma.

- **Objetivo.**

Crear enlaces a redirecciones no validadas y engañar a la victima de manera que el usuario haga clic en esos enlaces que los llevan a sitios inseguros, pero para el usuario suele ser sitios confiables, evadiendo los controles de seguridad.

- **Tipo de organización.**

La red social Facebook ha sufrido esta vulnerabilidad, entre muchas organizaciones.

- **Debilidad de la aplicación.**

Hay varias fallas en el sistema entre muchas se tiene autenticación insuficiente, falta de configuración, funcionalidad vulnerable, manejo impropio de entradas y procesos de validación insuficiente.

3. VULNERABILIDADES DE LOS SERVICIOS WEB

Los tipos de vulnerabilidad son tipos de ataque conocidos en los servicios Web. Contemplan tres etapas principales la preparación, la activación y la ejecución. Durante la ejecución del ataque el agresor logra violar los principios de seguridad que son integridad, disponibilidad y confidencialidad. Se han realizado diversos intentos de obtener una clasificación de ataques o incidentes de seguridad que realmente aporten en el entendimiento del problema, buscando cumplir con la mayor cantidad de las características propuestas por Cohen [58]. Según Cohen una clasificación o taxonomía de vulnerabilidades debe cumplir con las siguientes características: que sean mutuamente excluyentes, exhaustivas, repetibles, aceptadas, útiles y que no sean ambiguas.

3.1 PRINCIPALES TAXONOMÍAS DE VULNERABILIDADES

Las principales taxonomías de vulnerabilidades permiten desde distinto enfoque hacer una clasificación sobre las vulnerabilidades. Ortiz y Galindo [59] presentan y explican las siguientes taxonomías o clasificaciones de incidentes en los servicios Web:

Lista de términos. Es una lista simple de términos a los que se les da su correspondiente definición. Incluye entre sus términos: negación de servicios, piratería de software, copia no autorizada de datos, degradación de servicios, análisis de tráfico, virus y gusanos, ataques de tiempo, caballos troyanos, bombas lógicas, entre otros.

Lista de categorías. Es una lista que introduce un elemento de mayor estructura. Tiene siete categorías para clasificar los ataques: robo de contraseñas, ingeniería social, errores de software y puertas traseras, fallas de autenticación, fallas de protocolos, fugas de información y negación de servicio.

Categorías de resultados. Es una lista que agrupa los ataques de acuerdo con los resultados obtenidos.

Listas empíricas. Es una lista de resultados con categorías basada en datos de clasificación empírica. Es una lista de ocho categorías que comprende: robo externo de información, programas de plaga, sobrepaso, autenticación o autorización, abuso de autoridad, abuso a través de la inacción y abuso indirecto. La desventaja de esta clasificación es que es difícil de recordar, es menos intuitiva y no hay una estructura que muestre relaciones entre las categorías.

Matrices. Es un esquema de clasificación basado en dos dimensiones. Las dos dimensiones pueden ser: vulnerabilidades (destrucción física, destrucción de información, robo de servicios, exploración y robo de información) y atacantes

potenciales (operadores, programadores, encargados de captura de datos, internos, externos, intrusos).

Taxonomías basadas en procesos. Permite el análisis de los ataques e incidentes. Se parte de los procesos normales que se tienen en el manejo de la información. Stallings [60] presenta un modelo, enfocado en la información en tránsito, que clasifica las amenazas de seguridad en cuatro categorías interrupción, intercepción, modificación, fabricación, vistas estas como ataques pasivos o activos. Este es un modelo simplificado que sin embargo, brinda una claridad conceptual, adecuada para el análisis de los ataques.

Taxonomía basada en procesos – Orientación al evento [52]. Se clasifica según tipo de vulnerabilidad y permite identificar el atacante, la herramienta, la vulnerabilidad, el blanco, el resultado no autorizado, el objetivo, el tipo de organización y la debilidad de la aplicación de cada tipo.

Taxonomía de amenazas a la seguridad de la información [53]. Clasifica el tipo de vulnerabilidad en cuatro grupos: interrupción, interceptación, modificación, fabricación. Tiene como desventaja que no permite desarrollar la estructura del tipo de vulnerabilidad.

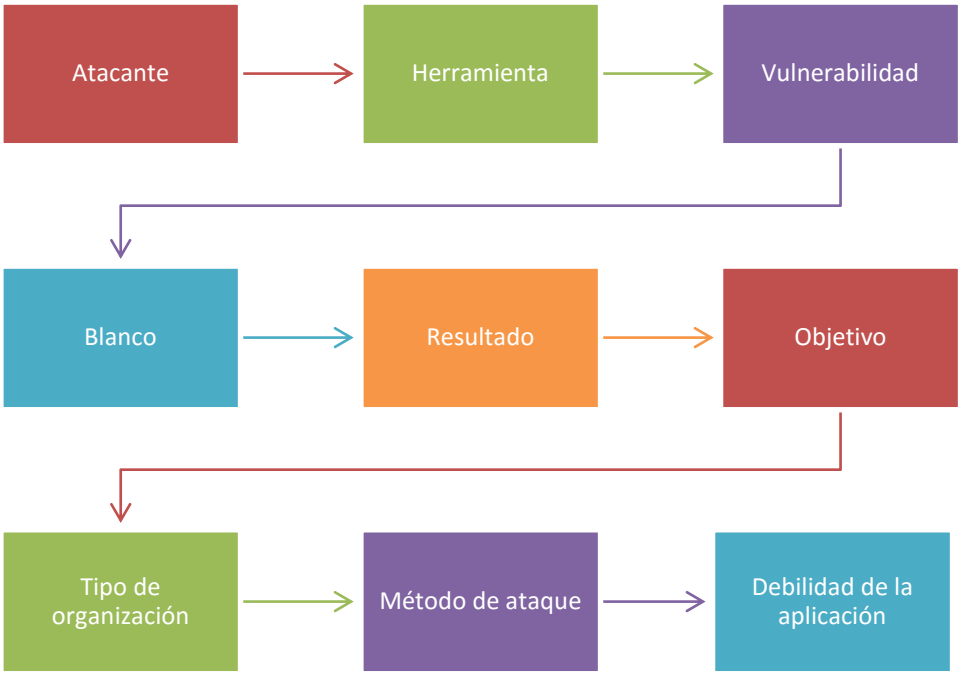
Taxonomías basadas en procesos – Orientación al acceso. En esta taxonomía un ataque se puede ver como un proceso que permite a un atacante alcanzar unos objetivos. Para lograrlo requiere un medio, un camino y un fin, que se pueden describir también como un atacante, una herramienta, un acceso, un resultado y un objetivo.

Taxonomía de gestión de incidentes para RedIRIS [54]. Clasifica el tipo de vulnerabilidad en: contenido abusivo, contenido malicioso, obtención de información, acceso/intrusión, disponibilidad, seguridad/confidencialidad de la información, fraude, helpdesk, otros. Su principal desventaja es que no permite desarrollar la estructura del tipo de vulnerabilidad.

3.2 TAXONOMÍA BASADA EN PROCESOS – ORIENTACIÓN AL EVENTO

Con base en el análisis de las características de las anteriores taxonomías se considera que la clasificación más útil para estudiar las vulnerabilidades de los servicios web REST es la taxonomía basada en procesos –Orientación al evento- ampliada cuyas características se describen en la publicación “Extensión de taxonomía y tratamiento de valores faltantes sobre un repositorio de incidentes de seguridad informática”, definida por el Instituto Sandia en el artículo “A Common Language for Computer Security Incidents”,[61] que propone el modelo que se presenta en la Figura 13.

Figura 13. Taxonomía basada en procesos – orientación al evento



Fuente. El Autor

La taxonomía propuesta ayuda a desarrollar la estructura de los tipos de vulnerabilidad, identificando el atacante, la herramienta, la vulnerabilidad, el blanco, el resultado no autorizado, el objetivo, el tipo de organización y la debilidad de la aplicación. La taxonomía muestra que los tipos de vulnerabilidad actúan con el objetivo de quebrantar los principios de seguridad.

La taxonomía propuesta define unos criterios de clasificación (véase Figura 14), que permiten hacer el desarrollo de los tipos de vulnerabilidad seleccionados, a partir de esto se define un criterio de evaluación que ayuda a identificar los principios de seguridad vulnerados. Los criterios son:

Atacante: Describe el tipo de autor que comete el tipo de actos indebidos.

- **Hackers:** Es cualquier individuo con conocimientos avanzados en las computadoras que se dedica a encontrar los puntos débiles y atacar con diferentes fines malintencionados**.**
- **Espías:** Se dedican a recopilar información confidencial de una computadora.
- **Terroristas:** Realizan operaciones ilícitas a través de la internet para dañar computadoras.

- **Intrusos corporativos:** Intentan vulnerar la seguridad de las empresas a través del internet o las aplicaciones.
- **Criminal profesional:** Es el individuo que se dedica a realizar ataques con el objetivo de obtener ganancia financiera.
- **Vándalo:** Persona que comete acciones criminales informáticas.
- **Voyeristas:** Son aquellas personas que se dedican a acosar financieramente las cuentas de los clientes.
- **Usuario externo e interno:** Estos individuos pueden hacer parte de las empresas ya sea interna o externamente, y aprovechar las vulnerabilidades informáticas.

Herramientas: Es el artefacto informático que el atacante usa para lograr su objetivo.

- **Ataque:** Proceso que diseñan los atacantes para conseguir objetivos malintencionados.
- **Intercambio de información:** El atacante busca información confidencial y la reemplaza por información distinta.
- **Comando de usuario:** Es una instrucción que el usuario proporciona al sistema con mala intención.
- **Agente:** Se describe como un programa agente que se instala en las computadoras para obtener información.
- **Script o programa:** Es un archivo de procesamiento que se instala en los en las bases de datos o servidores para robar recursos.
- **Toolkit:** Es un editor de texto que puede ser utilizado para guardar información confidencial.
- **Toma de datos:** Es la actividad o herramienta que utiliza el atacante para apoderarse de información valiosa.

Vulnerabilidad: Es la capacidad disminuida de un sistema de software.

- **Diseño:** Se presentan en la fase de análisis y diseño del ciclo de software.
- **Implementación:** Se presentan en la fase de implementación del ciclo de software.
- **Configuración:** Se presentan en la fase de desarrollo del ciclo de software.
- **Política de seguridad:** Se pueden presentar en la construcción del ciclo de vida del software.

Blanco: Es el punto de ataque.

- **Cuenta:** Es todo tipo de cuenta de usuario de la cual el atacante busca sacar provecho.
- **Proceso:** Es un tipo de blanco para el atacante reunir todos los datos en una base de datos de procesos determinados.
- **Dato:** El atacante busca obtener la información sensible de los usuarios.

- **Componente:** El atacante busca tener control de los elementos de los sistemas de software de los usuarios.
- **Computador:** El atacante como blanco de ataque logra tener el control sobre las distintas computadoras de los clientes o usuarios.
- **Red:** El atacante toma el control de la red para apoderarse de la información confidencial
- **Intranet**: El atacante busca apoderarse de las redes de las compañías.

Resultado no autorizado: Es la identificación del tipo de resultado al que se asemeja el ataque.

- **Acceso incrementado:** Es el acceso no autorizado en los aplicativos el cual sucede con frecuencia.
- **Difusión de la información:** Proceso del atacante para transmitir información del usuario.
- **Corrupción de la información:** El atacante vulnera la información y daña los datos.
- **Negación de la información:** Esto sucede cuando el usuario no encuentra su información en los sistemas, la cual ha sido robada.
- **Robo de recursos:** El atacante saca la información de las bases de datos sin autorización.

Objetivo: Es el fin al que se dirige una acción u operación.

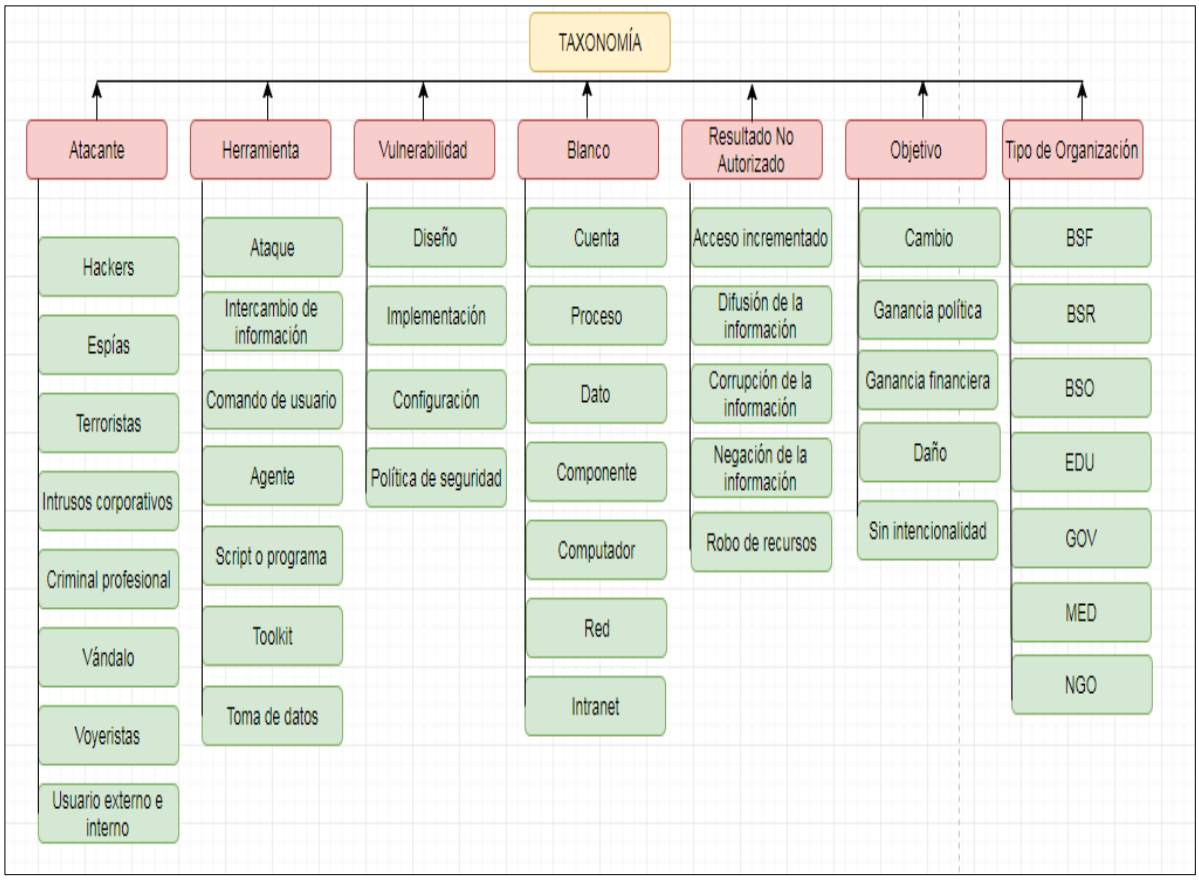
- **Cambio:** El atacante cambia la información de los usuarios.
- **Ganancia política:** Los ataques van dirigidos hacia procesos políticos y así sacar provecho.
- **Ganancia financiera:** Los ataques van dirigidos hacia procesos financieros y así sacar provecho.
- **Daño:** Los ataques también se pueden presentar con el objetivo de hacer daño sin intenciones de obtener dinero.
- **Sin intencionalidad:** Son ataques que se hacen sin causar daño.

Debilidad de la aplicación: Son los puntos de falla en los sistemas de software.

- **Autenticación insuficiente:** Los sistemas no cuentan con suficientes políticas de seguridad, lo que ocasiona que la autenticación sea insuficiente.
- **Autorización insuficiente:** Los sistemas no cuentan con suficientes políticas de seguridad, lo que ocasiona que la autenticación sea insuficiente.
- **Entropía insuficiente:** Es una medida insuficiente de la salida del mensaje hacia el usuario.
- **Falta de configuración en base de datos, infraestructura y servidor Web:** Debilidad que aprovecha el atacante para tener el control sobre la base de datos y servidores.

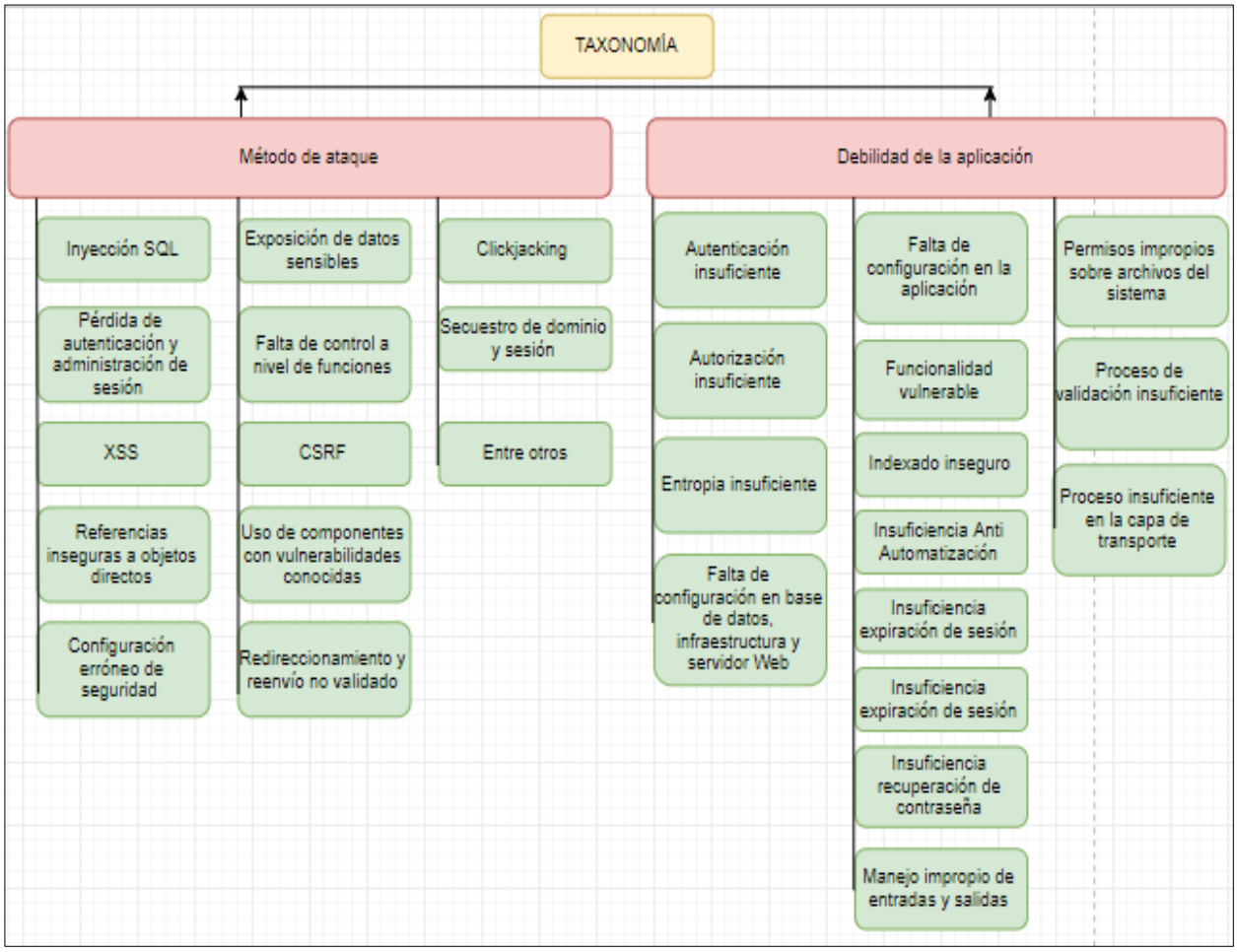
- **Falta de configuración en la aplicación:** Esto se presenta cuando las aplicaciones no tienen un sistema de seguridad estable.
- **Funcionalidad vulnerable:** Dentro de los sistemas se presenta cuando los desarrolladores trabajan con los mismos componentes o funciones sin actualizaciones.
- **Indexado inseguro:** Es un método de accesos al sistema que es inseguro.
- **Insuficiente Anti Automatización:** Los procesos manuales que pueda hacer el usuario en los sistemas pueden ser inseguro.
- **Insuficiencia expiración de sesión:** Dentro de las políticas de seguridad la expiración de sesión debe tener un tiempo,
- **Insuficiencia recuperación de contraseña:** Si el usuario necesita recuperar su contraseña debe pasar por un proceso de validación.
- **Manejo impropio de entradas y salidas:** El atacante vulnera las entradas y salidas de datos en los sistemas.
- **Permisos impropios sobre archivos del sistema:** El atacante obtiene de manera ilegal los permisos para entrar al sistema y hacer uso de los archivos.
- **Proceso de validación insuficiente:** Se presenta cuando no hay buenas políticas de validación de los datos que ingresan los usuarios.
- **Proceso insuficiente en la capa de transporte:** Es el paso de errores de los datos entre el cliente y el servidor.

Figura 14. Taxonomía



Fuente. CARVAJAL, Carlos y BAYONA, Diego. Revista Ingeniería, 2012. p. 31.

Figura 14. (Continuación)



Fuente. CARVAJAL, Carlos. BAYONA, Diego. Revista Ingeniería, 2012. p. 32.

4. METODOLOGÍA

Para cumplir con los objetivos se realizó la revisión sistemática de la literatura existente sobre las debilidades en los servicios Web REST. La revisión sistemática es una herramienta útil que permite, entre otros, obtener información de calidad y actualizada sobre los servicios Web REST. Consta de tres fases principales: planificación de la revisión, ejecución de la revisión y reporte de la revisión que son los resultados obtenidos.

4.1 PLANIFICACIÓN DE LA REVISIÓN SISTEMÁTICA

Formulación de la pregunta.

REST es un estilo de arquitectura de software que obtiene datos o indica operaciones sobre los datos través del protocolo de transmisión HTTP en formatos JSON, XML y otros. Los servicios Web desarrollados con el estilo de arquitectura REST implementan un servicio Web conocido como Restful, que consiste en la aceptación de unos principios para mejorar las comunicaciones cliente – servidor. Sin embargo, los servicios Web Restful presentan problemas de vulnerabilidad en la red tanto en la interceptación de mensajes como en la suplantación de identidad. Por lo tanto, para encauzar la revisión sistemática se formuló la siguiente pregunta ¿Cuáles son las debilidades que se presentan y ponen en riesgo la seguridad en los servicios Web REST?

Adopción de una metodología.

Las revisiones sistemáticas son artículos científicos ampliamente usados desde hace varios años en las ciencias de la salud. Recientemente se han propuesto metodologías para revisiones sistemáticas en las ciencias empresariales, ciencias sociales y economía. La revisión sistemática aplicada en este trabajo está basada en la publicación de Jhon Eder Masso [56] quien presenta una metodología para revisiones sistemáticas en Informática.

Definición de las palabras clave.

Las palabras claves o términos que ayudaron en la búsqueda de información fueron: Cross-Site Request Forgery (CSRF), Cross-site Scripting (XSS), Inyección de SQL, Pérdida de autenticación y administración de sesión y Referencias Inseguras a objetos firectos.

Población.

Todas las organizaciones, usuarios y roles que de una u otra forma diseñan y crean aplicaciones Web REST.

Elección de recursos.

IEEE Xplore, ACM Digital Library y ScienseDirect, se eligieron por ser sociedades científicas de formación tecnológica que promueven la creatividad y el desarrollo.

Definición de la cadena de búsqueda.

SQL Injection AND Broken Authentication and Session Management AND Insecure Direct Object References AND (Cross-site Scripting OR XSS) AND (Cross-Site Request Forgery OR CSRF).

Selección de estudios.

Para la selección de los artículos o estudios, se hizo el proceso de búsqueda a partir de la definición de los siguientes criterios de inclusión/exclusión:

Inclusión.

Artículos científicos con títulos que llevaran una o varias palabras clave, que al realizar la lectura de la introducción se hablara sobre el tema de soluciones de seguridad en los servicios Web o sobre los tipos de vulnerabilidad seleccionados y con fechas de los años 2011, 2012, 2013, 2014, 2015, 2016 y 2017.

Exclusión.

Páginas Web, abstracts, resúmenes, prólogos, documentos que no cuenten con la publicación completa, artículos duplicados, artículos o trabajos de pregrado, puntos de vista, análisis, reflexiones y estudios que no presenten con claridad los datos empíricos generados en las investigaciones.

Revisión de protocolo.

El protocolo o ficha fue recomendado por el Director del Programa de Ingeniería de Sistemas y Computación, y por el asesor del trabajo de grado, quienes lo utilizan en sus trabajos de investigación. Se estudió con el asesor la conveniencia y forma de diligenciarlo. Con esta ficha (véase Tabla 9), se hizo la extracción de los datos de los artículos identificando título, revista, fecha, autores, resumen documental, volumen, fascículo, páginas, ISBN y DOI. De esta manera, se logra conocer en el resumen documental la información relacionada con las vulnerabilidades en REST o los mecanismos de seguridad de las aplicaciones Web.

Tabla 9. Protocolo revisión sistemática

Título del Artículo	
Revista o publicación	
Fecha	
Autores	
Resumen documental	
ISBN	
Páginas	
DOI	
Enlace de publicación	

Fuente. Holman Bolívar. Ingeniero de Sistemas y Computación.

4.2 EJECUCIÓN DE LA REVISIÓN SISTEMÁTICA

Selección de artículos iniciales.

Los 50 artículos científicos seleccionados se obtuvieron de las bases de datos IEEE Xplore, ACM Digital Library, ScienseDirect, y de acuerdo a los criterios de inclusión y exclusión establecidos en la metodología de la revisión sistemática, se hizo la búsqueda por los siguientes temas: SQL Injection, Insecure Direct Object References, Broken Authentication and Session Management, Cross-site Scripting (XSS), Security Misconfiguration, Sensitive Data Exposure, Missing Function Level Access Control, Cross-Site Request Forgery (CSRF), Using Components with Known Vulnerabilities y Unvalidated Redirects and Forwards. El listado de los 50 artículos seleccionados se presenta en la siguiente tabla

Tabla 10. Artículos seleccionados

No.	Artículo	Año	Idioma	Fuente
1	OWASP Top 10 2013: actualización de los riesgos más extendidos asociados a las aplicaciones Web	2013	Español	Revista SIC
2	Extensión de taxonomía y tratamiento de valores faltantes sobre un repositorio de incidentes de seguridad informática	2012	Español	Revista de Ingeniería
3	A taxonomy of SQL Injection Attacks	2013	Inglés	IEEE
4	A taxonomy of SQL Injection Detection and Prevention Techniques	2013	Inglés	IEEE
5	Detecting SQL injection attacks using query result size	2014	Inglés	ScienceDirect
6	Detecting SQL injection attacks using graph of tokens and SVM	2016	Inglés	ScienceDirect
7	SQL Injection: Types, Methodology, Attack Queries and Prevention	2016	Inglés	IEEE
8	SQL Injection Attacks Prevention Based on Decision Tree Classification	2015	Inglés	IEEE
9	Mitigating SQL Injection Attacks Via Hybrid Threat Modelling	2015	Inglés	IEEE
10	Behind an Application Firewall, Are we safe from SQL	2015	Inglés	IEEE

	Injection Attacks?			
11	Algorithm to Prevent Back End Database against SQL Injection Attacks	2014	Inglés	IEEE
12	Root Cause Analysis of Session Management and Broken Authentication Vulnerabilities	2012	Inglés	IEEE
13	One time cookies: Preventing session hijacking attacks with stateless authentication tokens	2012	Inglés	ACM
14	SessionJuggler: Secure Web Login From an Untrusted Terminal Using Session Hijacking	2012	Inglés	ACM
15	A Session Key Utilization Based Approach For Memory Management in Wireless Networks	2014	Inglés	ACM
16	Automated Session Fixation Vulnerability Detection in Web Applications using the Set-Cookie HTTP response header in cookies	2014	Inglés	ACM
17	Securing Web Applications from injection and Logic Vulnerabilities Approaches and Challenges	2016	Inglés	ScienceDirect
18	Modeling the Propagation of XSS Worm on Social Networks	2013	Inglés	IEEE
19	Cross-Site Scripting (XSS) Worms in Online Social Network (OSN): Taxonomy and Defensive Mechanisms	2016	Inglés	IEEE
20	Reverse Analysis Method of Static XSS Defect Detection Technique Based on Database Query Language	2014	Inglés	IEEE
21	A Study of XSS Worm Propagation and Detection Mechanisms in Online Social Networks	2013	Inglés	IEEE
22	Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment	2016	Inglés	ScienceDirect
23	BIXSAN: Browser Independent XSS Sanitizer for prevention of XSS attacks	2011	Inglés	ACM
24	On the Automatic Detection Algorithm of Cross Site Scripting (XSS) with the Non-Stationary Bernoulli Distribution	2012	Inglés	IEEE
25	A Survey On Web Application Vulnerabilities(SQLIA,XSS)Exploitation and Security Engine for SQL Injection	2012	Inglés	IEEE
26	LigRE: Reverse-Engineering of Control and Data Flow Models for Black-Box XSS Detection	2013	Inglés	IEEE
27	On Security Issues in Web Applications through Cross Site Scripting (XSS)	2013	Inglés	IEEE
28	Analysis and suggestions for the Security of Web Applications	2011	Inglés	IEEE
29	Análisis de riesgos de las aplicaciones web de la superintendencia de bancos y seguros, utilizando las recomendaciones top ten de OWASP	2014	Español	ESPE
30	Early Detection of Security Misconfiguration Vulnerabilities in Web Applications	2011	Inglés	IEEE
31	Refactoring Multi-Layered Access Control Policies Through (De)Composition	2013	Inglés	IEEE
32	Discovering Access-Control Misconfigurations: New Approaches and Evaluation Methodologies	2012	Inglés	IEEE
33	Two threat patterns that exploit “Security misconfiguration” and “Sensitive data exposure” vulnerabilities	2015	Inglés	ACM

34	Detecting and Resolving Policy Misconfigurations in Access-Control Systems	2011	Inglés	ACM
35	Why eve and mallory (also) love webmasters: a study on the root causes of SSL misconfigurations	2014	Inglés	ACM
36	Do not blame users for misconfigurations	2013	Inglés	ACM
37	Privacy-Preserving Detection of Sensitive Data Exposure	2015	Inglés	IEEE
38	Identifying Sensitive Data Items within Hadoop	2015	Inglés	IEEE
39	Privacy-Preserving Scanning of Big Content for Sensitive Data Exposure with MapReduce	2015	Inglés	ACM
40	LRBAC: Flexible Function-Level Hierarchical Role Based Access Control for Linux	2015	Inglés	IEEE
41	Analysis to define management of identities access control of security processes for the registration civil from Ecuador	2016	Inglés	IEEE
42	Robust biometrics based three-factor remote user authentication scheme with key agreement	2013	Inglés	IEEE
43	Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS & CSRF	2016	Inglés	IEEE
44	An Analysis of XSS, CSRF and SQL Injection In Colombian Software And Web Site Development	2016	Inglés	IEEE
45	Lightweight Server Support for Browser-Based CSRF Protection	2013	Inglés	ACM
46	SV-AF – A Security Vulnerability Analysis Framework	2016	Inglés	IEEE
47	Identifying Outdated Requirements Based on Source Code Changes	2012	Inglés	IEEE
48	Integration of Static and Dynamic Code Analysis for Understanding Legacy Source Code	2016	Inglés	IEEE
49	Semantic-Based Extraction Approach for Generating Source Code Summary Towards Program Comprehension	2015	Inglés	IEEE
50	URFDS: Systematic Discovery of Unvalidated Redirects and Forwards in Web Application	2015	Inglés	IEEE

Fuente. El Autor

Evaluación de la calidad de los artículos.

Los artículos que cumplieron con los criterios de inclusión definidos en la revisión sistemática, proporcionaron toda la información requerida para diligenciar los protocolos o fichas de la revisión sistemática cuyo contenido tiene que ver con la seguridad y la vulnerabilidad de los servicios web REST. Se consultaron bases de datos reconocidas que tienen el registro de excelentes artículos sobre ingeniería, electrónica, computación, telecomunicaciones, telemática, mecatrónica, tecnología de control, robótica, biomédica y biónica, procesamiento digital de señales, sistemas energéticos, entre otras ramas.

Extracción de los datos de los artículos seleccionados.

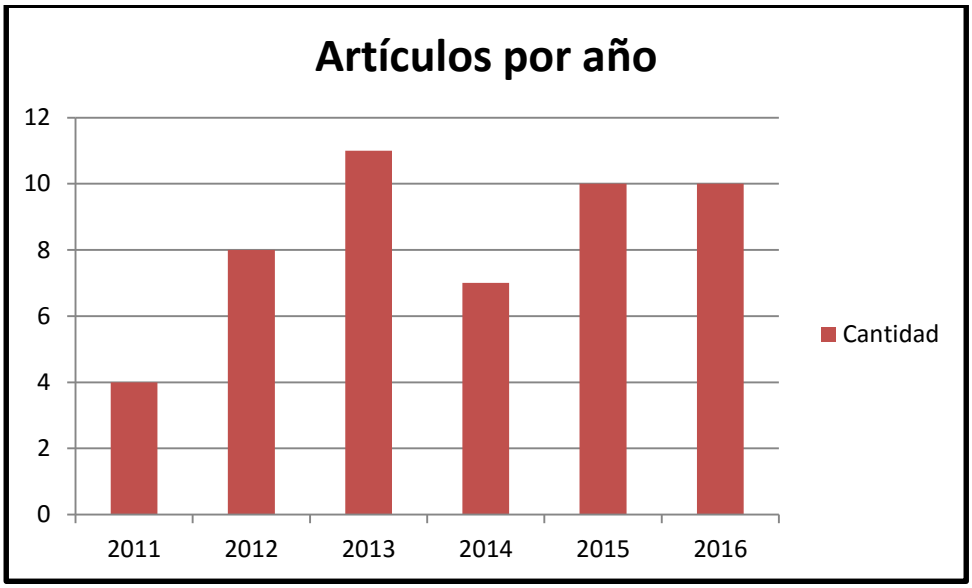
El proceso de análisis y síntesis de los artículos, inicia con la lectura del abstract, la introducción, las secciones, los resultados, las conclusiones y las referencias lo que permitió extraer los datos solicitados en el registro de la ficha o protocolo.

4.3 REPORTE DE REVISIÓN

Análisis de resultados.

Las publicaciones seleccionadas en los diferentes estudios (véase Tabla 11) según lo definido en el diseño experimental de la fase de planificación de la revisión sistemática se cuantificaron por año, idioma y fuente (véase Figura 15, 16 y 17).

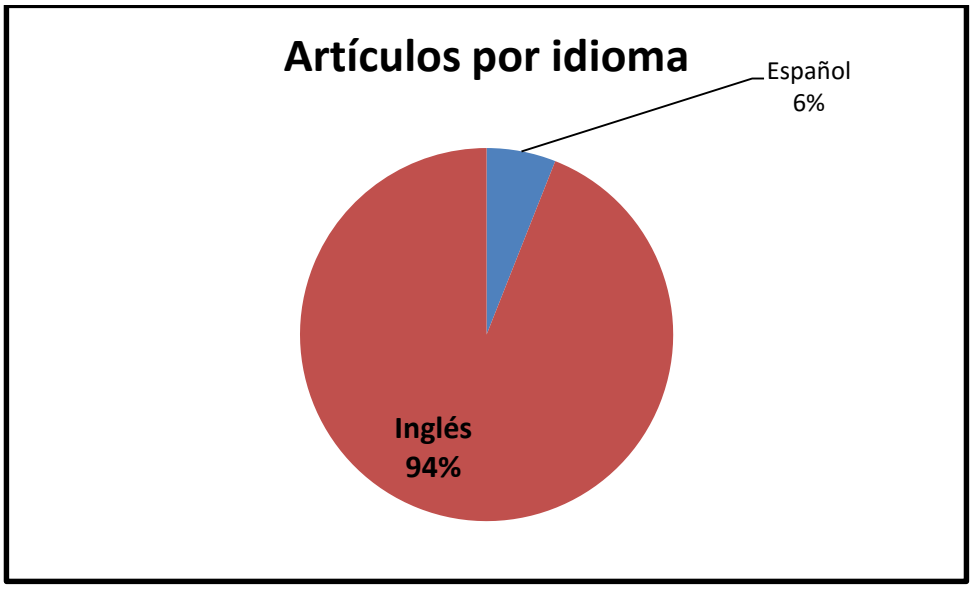
Figura 15. Artículos por año



Fuente. El Autor

La gráfica muestra que más del 75% de los artículos seleccionados se publicaron ente los años 2013 a 2016. Las publicaciones seleccionadas corresponden a los últimos cinco años.

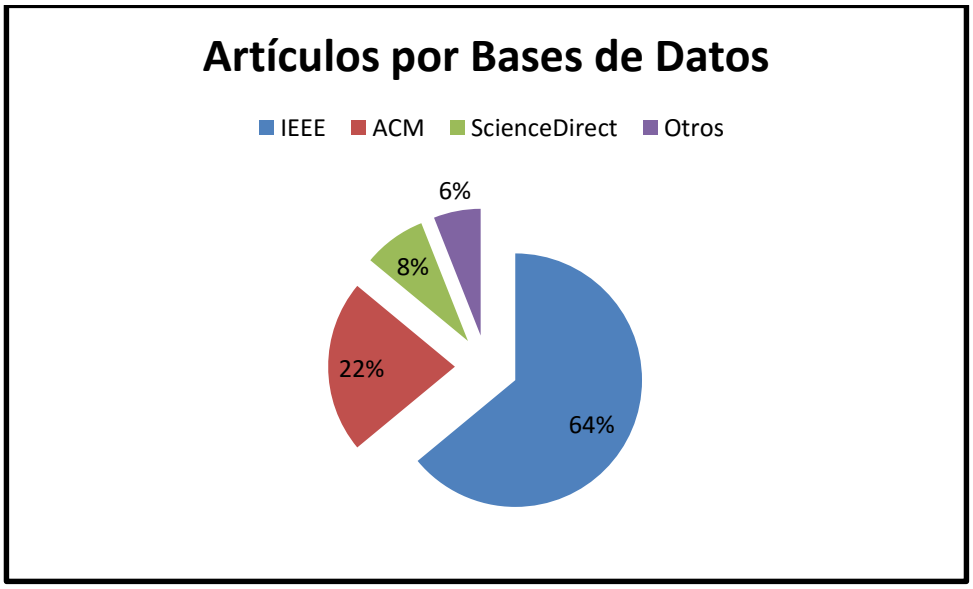
Figura 16. Artículos por idioma



Fuente. El Autor

La figura 16 gráfica muestra que el 94% de los artículos están escritos en idioma inglés, puesto que la fuente de información son las bases de datos IEEE y ACM. Tres publicaciones están en español. Una es la versión en español de la publicación en inglés de OWASP. La segunda es una publicación de un grupo de investigación de la Universidad Distrital Francisco José de Caldas donde se describe la taxonomía utilizada en este trabajo. Y la tercera es un análisis de riesgos de las aplicaciones web de la superintendencia de bancos y seguros, utilizando las recomendaciones top ten de OWASP.

Figura 17. Artículos bases de datos



Fuente. El Autor

La figura 17 muestra que la mayoría de los artículos estudiados fueron obtenidos de la base de datos IEEE que es la base de datos más utilizada en las áreas tecnológicas y en segundo lugar de ACM de la Association of Computing Machinery, que es una base de datos con la información más completa de publicaciones, proceedings y libros en Ciencias de la Computación.

5. RESULTADOS

En este capítulo se presenta el resultado del análisis de los 50 artículos que estudian tipos de vulnerabilidades de los servicios Web REST, seleccionados en la revisión sistemática. El estudio del tipo de vulnerabilidad reportada en cada artículo se hace con referencia a las características definidas en la Taxonomía Basada en Procesos – Orientación al Evento descrita en el capítulo 3, con el propósito de identificar el atacante (autor que comete el acto indebido), la herramienta (recurso informático usado por el atacante), la vulnerabilidad (debilidad del software), el blanco (punto de ataque), el resultado no autorizado (tipo de resultado), el objetivo (fin de la acción), el tipo de organización (identificación), el método de ataque (intruso) y la debilidad de la aplicación (fallas del software). El resultado final del capítulo es la síntesis de los estudios individuales que llevan a concluir cuales principios de seguridad (integridad, disponibilidad y confidencialidad) son vulnerados por los tipos existentes en los servicios Web REST.

Inyección de SQL

Amirmohammad Sadeghian, Mazdak Zamani, Shahidan M. Abdullah [27] afirman que la flexibilidad de SQL lo convierte en un lenguaje potente que le permite al usuario preguntar por la información que desea sin tener ningún conocimiento sobre cómo se va a buscar la información. Sin embargo, el uso de bases de datos basadas en SQL lo convierten en el centro de atención de los hackers. El ataque de inyección de SQL es una amenaza de seguridad bien conocida para las aplicaciones Web impulsadas por bases de datos. Un ataque de inyección de SQL con éxito revela información confidencial crítica al hacker.

Amirmohammad Sadeghian, Mazdak Zamani, Azizah Abd. Manaf [27] afirman que una de las vulnerabilidades de aplicaciones Web más graves y peligrosas es la inyección de SQL. El ataque de inyección de SQL se realizó insertando una porción de consulta SQL malintencionada a través de una entrada no validada del usuario en la instrucción de consulta legítima. En consecuencia, el sistema de gestión de bases de datos ejecuta estos comandos y conduce a la inyección de SQL.

Young-Su Jang, Jin-Young Choi 4] dicen que las aplicaciones Web se están convirtiendo en una parte esencial de nuestra vida cotidiana, actividades dependientes de la funcionalidad y seguridad de estas aplicaciones. Aplicaciones Web son ubicuos, realizan tareas críticas y manejan datos de usuario sensibles. Como la escala de estas aplicaciones crece, las vulnerabilidades de inyección, tales como inyecciones de SQL, se convierten en grandes desafíos de seguridad. La mayoría de estas vulnerabilidades se derivan de la falta de validación.

Debabrata Kar, Suvasini Panigrahi y Srikanth Sundararajan [5] afirman que los ataques de inyección SQL han predominado en las bases de datos Web desde hace 15 años. Explotando las fallas de validación de entrada, los atacantes inyectan código SQL a través del front-end de sitios Web y roban datos desde bases de datos de back-end. Detección de ataques de inyección de SQL ha sido un problema difícil debido a la extrema heterogeneidad de los vectores de ataque.

Nanhay Singh, Mohit Dayal y R.S. Raw, Suresh Kumar dice que la inyección de SQL se puede definir como la técnica en la que el hacker ejecuta consultas SQL malintencionadas en el servidor de base de datos a través de una aplicación Web para obtener acceso a través de la información sensible o de la base de datos. Esta es la vulnerabilidad basada en Web que permite al atacante falsificar la identidad, destruye los datos presentes en el sistema y cambia los registros presentes en la base de datos[7].

B.Hanmanthu, B.Raghu Ram y Dr.P.Niranjan comentan que en el mundo real, la dependencia de las aplicaciones de la World Wide Web aumenta día a día y se transforma en vulnerables a los ataques de seguridad. De todos los ataques diferentes, los ataques de inyección de SQL son los más comunes[9].

Habeeb Omotunde, Rosziati Ibrahim contribuyen que las aplicaciones Web que dependen de las bases de datos de back-end no son actualmente inmunes a los ataques de inyección SQL a pesar de la enorme inversión en artefactos de seguridad y mecanismos defensivos de software desplegados por las organizaciones. Estas formas de ataques implican la inserción de cadenas mal formadas o entrada especialmente diseñada codificada como consulta SQL en formularios Web o solicitudes de encabezado http a servidores Web[26].

Dennis Appelt, Cu D. Nguyen, Lionel Briand piensan que los firewalls en las aplicaciones Web son una capa indispensable para proteger los sistemas en línea de los ataques de inyección SQL. Sin embargo, el ritmo acelerado en el que aparecen nuevos tipos de ataques y su sofisticación requieren que los firewalls se actualicen y prueben periódicamente, ya que de lo contrario serán eludidos[8].

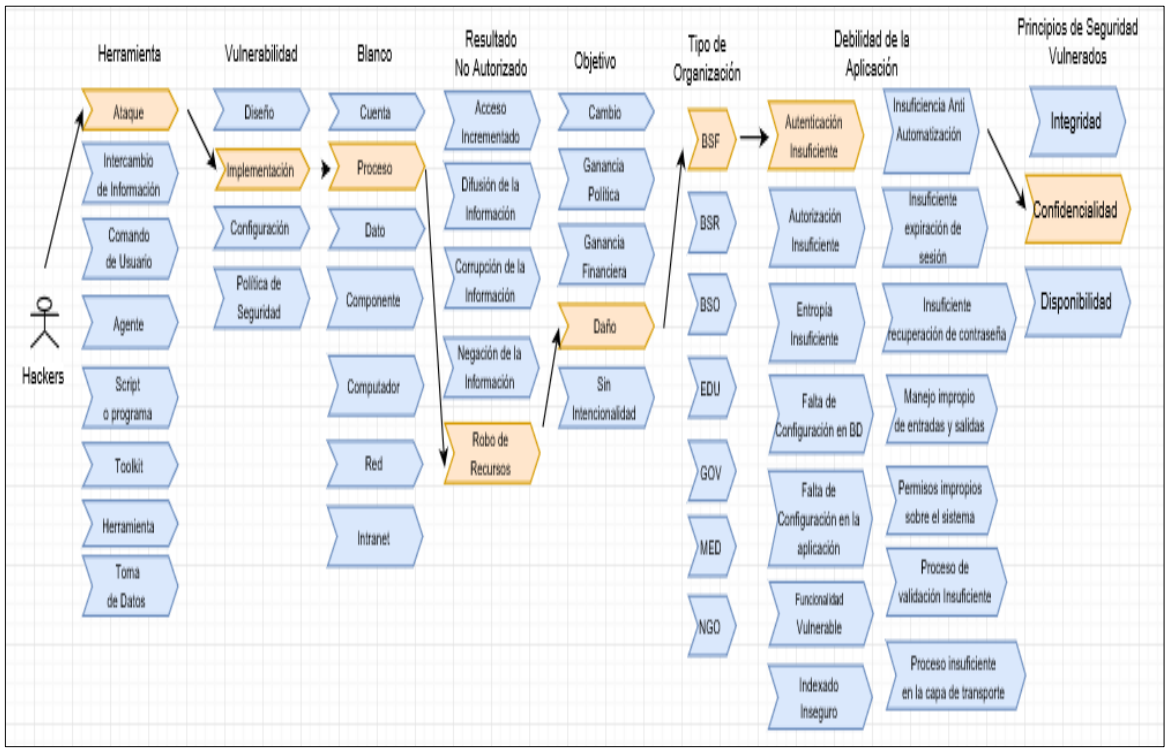
Mahima Srivastava afirma que el ataque de inyección SQL (SQLIA) es una técnica a través de la cual los atacantes acceden a las bases de datos de back-end insertando los códigos maliciosos a través de front-end. En los últimos tiempos, los ataques de inyección SQL (SQLIA) han surgido como una amenaza importante para la seguridad de la base de datos[10].

Síntesis

Los usuarios tienen acceso a través de la internet a los sistemas de información, en la mayoría de sistemas se requiere una autenticación la cual se realiza con un usuario y una contraseña, estos datos son enviados para ser validados en una

Base de datos para obtener una autorización. Un hacker burla este tipo de seguridad con un ataque de Inyección SQL, él no cuenta con un usuario y una contraseña, entonces aprovecha la vulnerabilidad en la Base de Datos y genera una instrucción similar que la Base de datos debe procesar para autenticar el usuario e ingresarla en la Web y como esta instrucción es válida para el sistema, el sistema autoriza el ingreso. Un ataque de inyección de SQL con éxito vulnera el principio de seguridad de confidencialidad, el cual revela información confidencial crítica al hacker (Véase Figura 18).

Figura 18. Clasificación de Taxonomía Inyección de SQL



Fuente. El Autor

Pérdida de autenticación y administración de sesión

Daniel Huluka, Oliver Popov afirma que existen numerosos enfoques para proteger las aplicaciones Web como una de las maneras más frecuentes de aprovechar el potencial de Internet, los atacantes casi todos los días presentan nuevos intentos de explotar diversas vulnerabilidades y comprometer los datos que se encuentran en la red. Uno de los posibles lugares para lograr soluciones sostenibles es seguir enfoques estratégicos basados en un análisis detallado y comprensión de los problemas en lugar de algunos de los métodos tácticos comunes y a menudo reactivos[11].

Italo Dacosta, Saurabh Chakradeo, Mustaque Ahamad Y Patrick Traynor piensan que los riesgos asociados con el uso de cookies como testigos de autenticación de sesión se conocen desde hace años. Han propuesto alternativas más robustas para reemplazar las cookies de autenticación. Sin embargo, no se han desplegado porque no cumplen con los requisitos de las alternativas que requieren una costosa sincronización de estado a través de las aplicaciones Web distribuidas Web 2.0. Específicamente, la mayor parte de la aplicación propuesta, una seria preocupación por los sistemas distribuidos[12].

Elie Bursztein, Chinmay Soman y Dan Boneh afirman que se utiliza las características modernas de los navegadores Web para desarrollar un sistema de inicio de sesión seguro desde un terminal no confiable. El sistema, llamado Session Juggler, no requiere cambios en el lado del servidor ni software especial en el terminal más allá de un navegador Web moderno. Esta importante propiedad hace que la adopción sea fácil que con las propuestas anteriores[29].

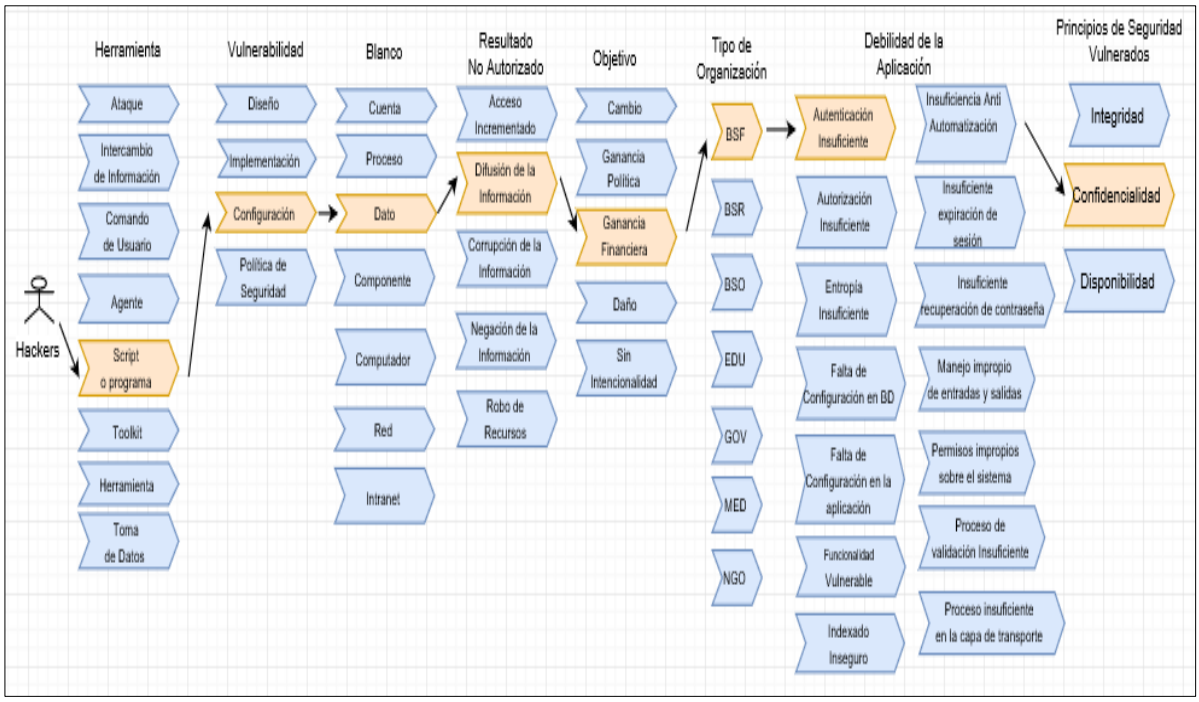
Arun Nagaraja y Saravana Kumar R. piensan que la criptografía simétrica se utiliza en redes inalámbricas, con la ayuda de claves de sesión para hacer que sea utilizado por el público. Las claves que se generan aleatoriamente para proporcionar seguridad se denominan clave de sesión y ésta es la clave de cifrado y descifrado para establecer la comunicación entre un usuario y otro ordenador[13].

Rahul Kumar, Indraveni K y Aakash Kumar Goel afirman que los sistemas son seguros, pero muchos de estas interfaces pueden tener graves problemas de seguridad asociados con que permiten que los ataques tengan éxito en la recolección de datos críticos. Una gestión de sesiones insegura es uno de esos comúnmente encontrado, sin embargo, una falla de seguridad severa en las aplicaciones Web que afectan los datos se gestionan y controlan en estas interfaces Web[14].

Síntesis

Esta vulnerabilidad está directamente relacionada con la seguridad de la información de los usuarios de los sitios Web, funciona con el objetivo de obtener las credenciales de un usuario en el sistema. El usuario ingresa sus datos de usuario y contraseña correctos, los cuales viajan a través del protocolo HTTP para ser verificados en una Base de Datos, el atacante explota esta vulnerabilidad de Pérdida de Autenticación y Administración de Sesiones, por medio de un Analizador de Tráfico de Red donde obtiene las credenciales de un usuario e ingresa de manera fraudulenta. Este método de ataque vulnera el principio de seguridad de confidencialidad (Véase Figura19).

Figura 19. Clasificación de Taxonomía Pérdida de autenticación y administración de sesión



Fuente. El Autor

Cross-site Scripting (XSS)

G. Deepa, P. Santhi Thilagam piensan que, aunque existen diferentes tipos de defectos de inyección, el alcance se restringe a SQL Injection y Cross-site scripting, ya que se clasifican como las principales amenazas por diferentes consorcios de seguridad. Aunque existen varios enfoques disponibles para proteger las aplicaciones Web de SQLI y XSS, siguen siendo frecuentes debido a su impacto y gravedad. Los defectos de la lógica están ganando la atención de los investigadores ya que violan las especificaciones comerciales de las aplicaciones. No hay una sola solución para mitigar todos los defectos. Se necesita más investigación en el área de fijación de defectos en el código fuente de aplicaciones[28].

Ying Zhao y Pingke Yi afirman que en los últimos años, ataques de secuencias de comandos entre sitios se producen normalmente en las redes sociales. Los atacantes ocultan los códigos de script maliciosos en los enlaces Web y obtienen información confidencial personal si alguien hace clic en estos vínculos[32].

Pooja Chaudhary, B. B. Gupta y Shashank Gupta afirman que la propagación de gusanos XSS en los sitios de redes sociales como Twitter, LinkedIn, Facebook,

etc. ha observado un crecimiento exponencial en la era moderna de la tecnología Web 2.0. Según una encuesta reciente, el 43% de las aplicaciones Web son vulnerables a los gusanos XSS. Este insoportable crecimiento de los gusanos XSS ha generado algunas preocupaciones serias de seguridad y privacidad en OSN[15].

Cui Baojiang, Long Baolian y Hou Tingting piensan que junto con el amplio uso de la aplicación Web, la vulnerabilidad XSS se ha convertido en uno de los problemas de seguridad más comunes y ha causado muchas pérdidas graves[16].

Mohammad Reza Faghani y UyenTrang Nguyen piensan que se presenta modelos analíticos y resultados de simulación que caracterizan los impactos de los siguientes factores en la propagación de gusanos XSS en las redes sociales en línea (OSNs):

Comportamientos de los usuarios, es decir, la probabilidad de visitar el perfil de un amigo versus de un extraño.

La estructura altamente agrupada de las comunidades.

Tamaños de la comunidad.

El análisis y resultados de simulación muestra que la estructura agrupada de una comunidad y la tendencia de los usuarios a visitar a sus amigos con más frecuencia que los extraños ayudan a frenar la propagación de gusanos XSS en OSNs[30].

Shashank Gupta, B.B.Gupta afirman que para evitar que las máquinas virtuales sean víctimas de ataques XSS en el entorno de computación en la nube, presentan una mejorada metodología defensiva XSS para las plataformas en la nube. Este marco analiza inicialmente las solicitudes HTTP para enlaces URI integrados que apuntan hacia los enlaces de archivos JS externos y que pueden contener carga útil XSS maliciosa[17].

Sharath Chandra y S. Selvakumar piensan que la proliferación de sitios de redes sociales y las aplicaciones Web que ofrecen contenido dinámico a los clientes han aumentado el contenido HTML creado por el usuario en la World Wide Web. Este contenido HTML creado por el usuario puede ser un vector notorio para ataques XSS. Los ataques XSS tienen la capacidad de dirigirse a sitios Web, robar información confidencial de los usuarios y secuestrar sus cuentas, etc. Los ataques XSS se lanzan para explotar las vulnerabilidades del mal desarrollado código de aplicación y los sistemas de procesamiento de datos. En particular, la validación incorrecta del contenido creado por el usuario y los mensajes de error personalizados no desinfectados introducen vulnerabilidad para los ataques XSS[18].

Daiki Koizumi, Takeshi Matsuda y Michio Sonoda afirman que XSS es un tipo de ataque de inyección desencadenado por los scripts maliciosos en los sitios Web. Si los atacantes lideran con éxito XSS, la sesión de navegación Web de los usuarios finales puede ser secuestrada y su información personal es a menudo robada por atacantes. Una de las razones principales para XSS son las vulnerabilidades en las aplicaciones Web. Sin embargo, esas vulnerabilidades a menudo se dejan para mantener servicios continuos[34].

Rahul Johari y Pankaj Sharma dicen que la inyección estructurada de SQL y XSS es quizás una de las técnicas de ataque de capa de aplicación más utilizadas por el atacante para desfigurar el sitio Web, manipular o eliminar el contenido mediante la introducción de cadenas de comandos no deseadas[33].

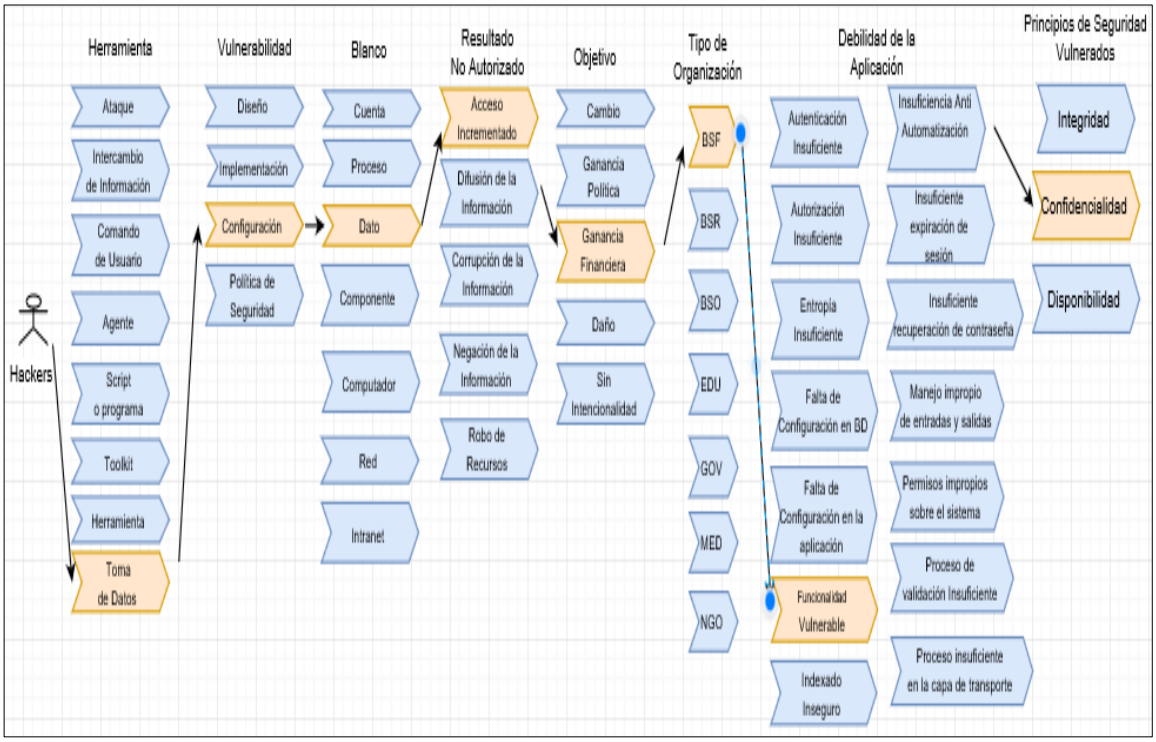
Fabien Duchène; Sanjay Rawat; Jean-Luc Richier; Roland Groz muestran que muestran que LigRE aumenta las capacidades de detección de XSS, un caso particular de vulnerabilidades de inyección de comandos Web[19].

Vikas K. Malviya, Saket Saurav y Atul Gupta afirman que las aplicaciones Web se han convertido en un medio muy popular de desarrollar software. Esto se debe a muchas ventajas de las aplicaciones Web, como la necesidad de instalación en cada máquina cliente, los datos centralizados, la reducción de los costos de negocio, etc. Con el aumento de esta tendencia las aplicaciones Web se están convirtiendo en vulnerables para los ataques. XSS es la principal amenaza para la aplicación Web, ya que es el ataque más básico en la aplicación Web[31].

Síntesis

Los atacantes con XSS explotan la confianza que un usuario tiene en un sitio en particular, lo que ocasiona un gran impacto. Hacen que a través de un buscador se ejecute un mensaje de alerta en JavaScript, para robar las cookies para luego robar la identidad, para esto los cibercriminales suelen enviar correo a sus víctimas donde hacen clic disfrazado y así se produzca el robo. El principio de seguridad vulnerado es la confidencialidad (Véase Figura 20).

Figura 20. Clasificación de Taxonomía Cross-site Scripting (XSS)



Fuente. El Autor

Referencias inseguras a objetos directos

You Yu, Yuanyuan Yang, Jian Gu, and Liang Shen dicen que la seguridad de las aplicaciones Web actuales enumera los ataques más comunes en ellas, como la inyección, las secuencias de comandos de sitios cruzados y las referencias directas a objetos inseguros. Luego, tomando como ejemplo un ataque por inyección, se explica los principios del ataque por inyección y se analiza las razones de la vulnerabilidad. Por último, para evitar estos ataques, ofrecen varias sugerencias valiosas. En general, la popularización de las aplicaciones Web resulta en el surgimiento de los problemas de seguridad en las aplicaciones Web. Estos problemas de seguridad han recibido cada vez más atenciones. Pero hay muchos reconocimientos erróneos sobre cómo protegerlos. La realidad indica que la protección de las aplicaciones Web se ha convertido en indispensable para la seguridad de los sistemas[20].

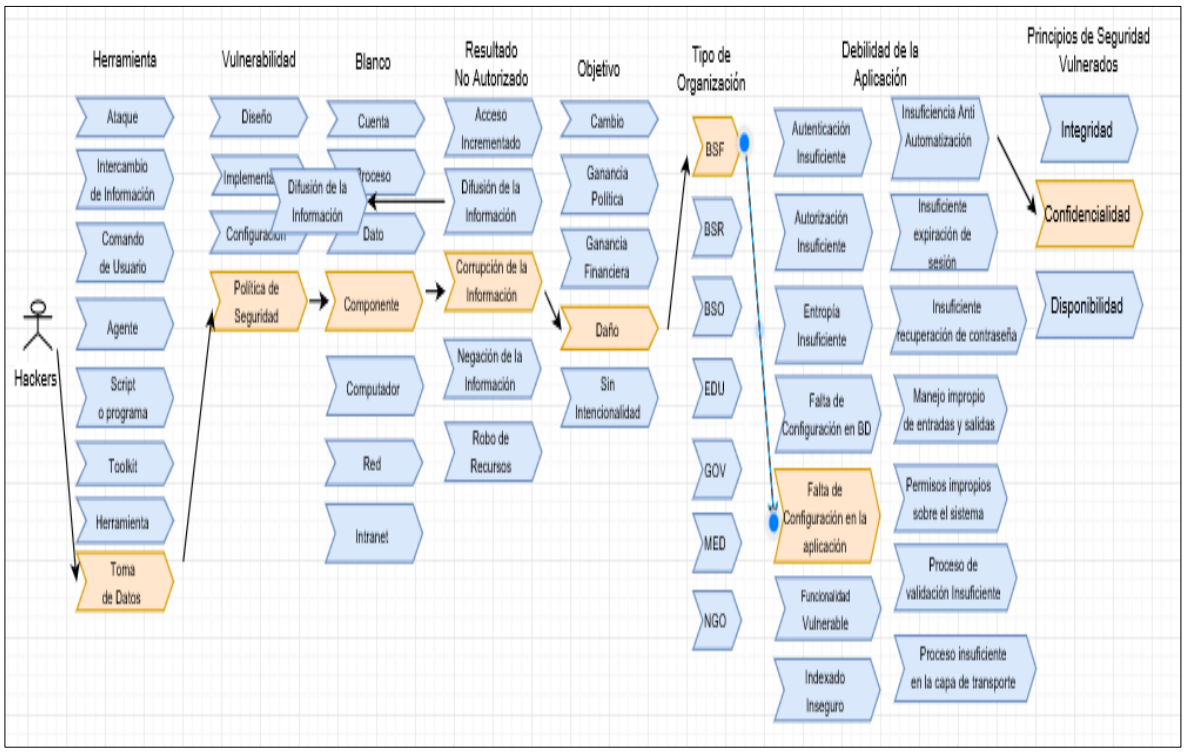
Lenin Salgado, Mario Ron y Fernando Solís dicen que se realiza una evaluación de los riesgos basados en el OWASP Top 10 - 2010, de las vulnerabilidades, amenazas, el impacto, detectándose que en las aplicaciones de la Superintendencia de Bancos y Seguros hay dos riesgos bien claros que presentan

una vulnerabilidad media-alta, al igual que la amenaza, pudiendo llegar el impacto a ser grande en caso de ser explotada, estos riesgos son: Almacenamiento Criptográfico Inseguro y Protección Insuficiente en la Capa de Transporte. Aunque el riesgo Inyección no está presente se recomienda utilizar la función Prepared Statement de Java, debido a que es la primera opción de defensa que propone OWASP para prevenir ataques de inyección[36].

Síntesis

Las aplicaciones Web están conformadas por objetos o elementos internos, como archivos, directorios y registros en bases de datos que utiliza la aplicación para almacenar información y que son referenciados a través de parámetros en las URL's o en formularios. Esta vulnerabilidad realiza una referencia directa a ellos, porque según la información que contengan estos parámetros, se accederá a uno u otro dato. El principio de seguridad vulnerado es la confidencialidad (Véase Figura 21).

Figura 21. Clasificación de Taxonomía Referencias Inseguras a Objetos Directos



Fuente. El Autor

Configuración errónea de seguridad

Birhanu Eshete, Adolfo Villafiorita y Komminist Weldemariam dicen que los ajustes de configuración de seguridad de los entornos de servidor en el desarrollo y la implementación de aplicaciones Web. También ofrece características para ajustar automáticamente la configuración de seguridad y cuantitativamente el nivel de seguridad de los entornos de servidor antes de implementar aplicaciones Web. Utilizando el software, se puede evaluar los paquetes de servidores Apache, PHP y MySQL en tres plataformas[21].

Matteo Maria Casalino y Romuald Thion afirman que el control de acceso basado en políticas es un paradigma bien establecido para asegurar sistemas de TI estratificados. Las políticas de control de acceso, sin embargo, a menudo no se centran en capas de arquitectura dedicadas, sino que emplean cada vez más conceptos de capas múltiples. Los servidores de aplicaciones Web, por ejemplo, suelen soportar el filtrado de peticiones sobre la base de direcciones de red. La flexibilidad resultante se acompaña de una mayor complejidad de gestión y del riesgo de una mala configuración de la seguridad cuando se examinan las distintas políticas de forma aislada[39].

Lujo Bauer, Yuan Liang, Michael K. Reiter, Chad Spensky dicen que los accesos que no están permitidos por la política implementada pero que comparten similitudes con accesos permitidos, pueden ser indicativo de las configuraciones erróneas de la política de control de acceso. Identificando tales configuraciones incorrectas permite a los administradores resolverlas antes interfieren con el uso del sistema. Se mejora el trabajo previo para identificar tales configuraciones erróneas de dos maneras principales[40].

Rohini Sulatycki y Eduardo B. Fernández dicen que los diseñadores deben entender primero las posibles amenazas antes de diseñar sistemas seguros. Sin embargo, identificar las amenazas no es suficiente. Se necesita entender cómo se realiza todo un mal uso aprovechándolos. Los patrones de amenaza y mal uso parecen ser una buena herramienta para entender cómo se realizan malversaciones. Es posible construir un catálogo relativamente completo de patrones de amenazas y mal uso para la seguridad de las aplicaciones[22].

Lujo Bauer, Scott Garriss, Michael K. Reiter dicen que la configuración errónea en la política de control de acceso que causan que las solicitudes sean erróneamente denegadas pueden resultar en pérdida de tiempo, frustración del usuario y, en el contexto de aplicaciones particulares (por ejemplo, atención médica), consecuencias muy graves. Se aplica la minería de reglas de asociación a la historia de accesos para predecir cambios en las políticas de control de acceso que probablemente sean consistentes con las intenciones de los usuarios, de modo que estos cambios puedan ser instituidos antes de que las confusiones interfieran con los accesos legítimos[23].

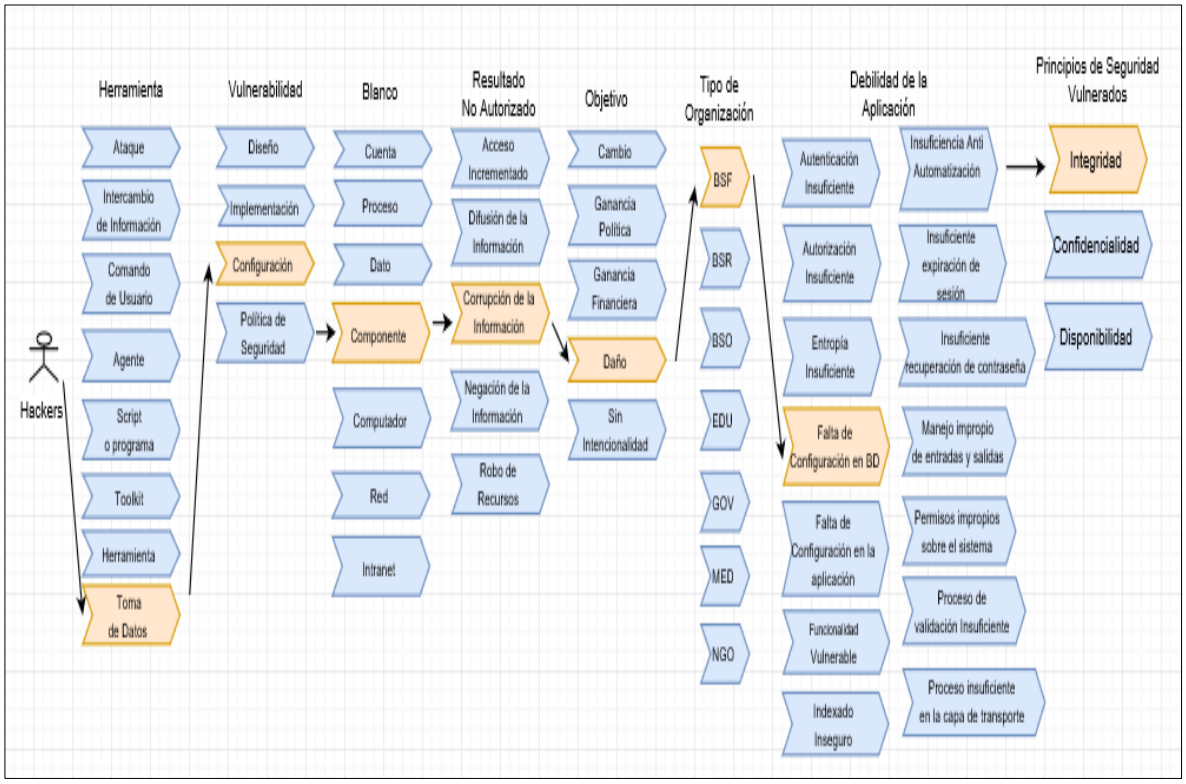
Sascha Fahl, Yasemin Acar, Henning Perl, Matthew Smith afirman que las investigaciones dan a conocer que la infraestructura SSL es un sistema frágil: la validación del certificado X.509 falla para un número no trivial de sitios Web habilitados para HTTPS, lo que resulta en mensajes de advertencia SSL presentados a los usuarios[37].

Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, Shankar Pasupathy afirman que al igual que los errores de software, los errores de configuración son también una de las principales causas de los fallos actuales del sistema. Muchos problemas de configuración se manifiestan de manera similar a errores de software como fallos, bloqueos, fallos silenciosos. Deja a los usuarios desorientados y obligados a informar a los desarrolladores de soporte técnico, desperdiciando no sólo el tiempo y esfuerzo de los usuarios, sino también de los desarrolladores[38].

Síntesis

La mayoría de las vulnerabilidades en las aplicaciones son por una configuración incorrecta, están relacionadas con errores de configuración en el servidor, ajustes inapropiados, versiones de software desactualizados y otros problemas relacionados de despliegue inseguro. El principio de seguridad vulnerado es Integridad (Véase Figura 22).

Figura 22. Clasificación de Taxonomía Configuración errónea de seguridad



Fuente. El Autor

Exposición de datos sensibles

Xiaokui Shu, Danfeng Yao y Elisa Bertino dicen que las estadísticas de las empresas de seguridad, instituciones de investigación y organizaciones gubernamentales muestran que el número de casos de fugas de datos ha crecido rápidamente en los últimos años. Entre varios casos de fugas de datos, los errores humanos son una de las principales causas de pérdida de datos. Existen soluciones para detectar fugas inadvertidas de datos sensibles causadas por errores humanos y para proporcionar alertas a organizaciones. Un enfoque común es la detección de contenido en almacenamiento y transmisión de información sensible expuesta[24].

Ashwin Kumar TK, Hong Liu, Johnson P Thomas, Goutam Mylavarapu dicen que el crecimiento reciente en los grandes datos está aumentando los problemas de seguridad y privacidad. Las organizaciones que recopilan datos de varias fuentes corren el riesgo de contraer obligaciones legales o comerciales debido a la violación de la seguridad y a la exposición de información delicada. Sólo el control de acceso a nivel de archivo es factible en la implementación actual de Hadoop y

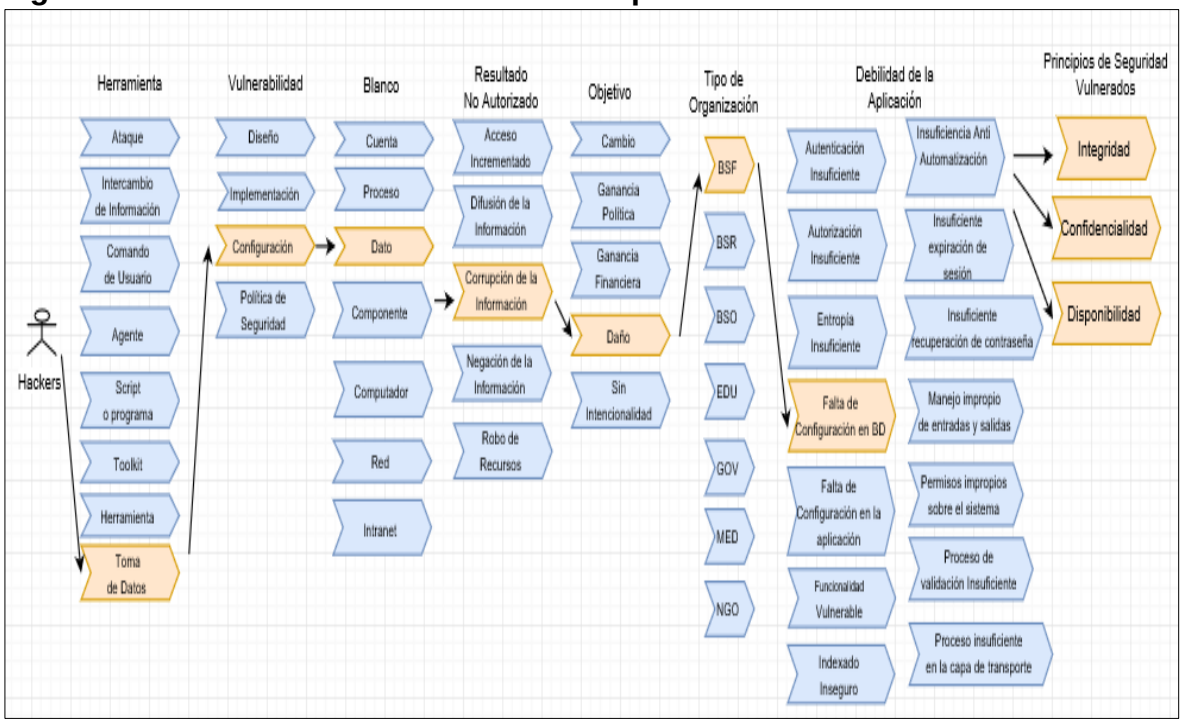
la información sensible sólo se puede identificar manualmente o de la información proporcionada por el propietario de los datos. El problema de identificar manualmente información sensible se complica debido a diferentes tipos de datos[25].

Fang Liu, Xiaokui Shu, Danfeng Yao, Ali R. Butt dicen que la exposición de datos sensibles en almacenamiento y transmisión plantea una grave amenaza para la organización y la seguridad personal. La detección de fugas de datos apunta a escanear contenido (en edad o transmisión) de los datos sensibles expuestos. Porque el volumen de datos y volúmenes de datos grandes, tal como un algoritmo de ritmo necesita ser escalable para una detección oportuna. Esta solución utiliza el marco MapReduce para detectar datos sensibles, ya que tiene la capacidad de arbitrar, escalar y utilizar recursos públicos para la tarea, tales como Amazon EC2[41].

Síntesis

En la mayoría de aplicaciones Web no se evidencia una adecuada protección de los datos sensibles como números de tarjetas o credenciales de autenticación. Los atacantes roban, modifican los datos para cometer actos de fraude, robar identidades. Los principios de seguridad vulnerados son Integridad, disponibilidad y confidencialidad (Véase Figura.23).

Figura 23. Clasificación de Taxonomía Exposición de datos sensibles



Fuente. El Autor

Falta de control de acceso de nivel de funciones

Javad Zandi, Abbas Naderi-Afooshteh presentan un modelo de control de acceso basado en roles flexibles con tecnologías sencillas y existentes que permite un control de acceso a nivel de funciones eficiente, prototipo en un sistema llamado LRBAC1, un módulo de kernel de Linux que impone el control de acceso sobre la ejecución del programa[57].

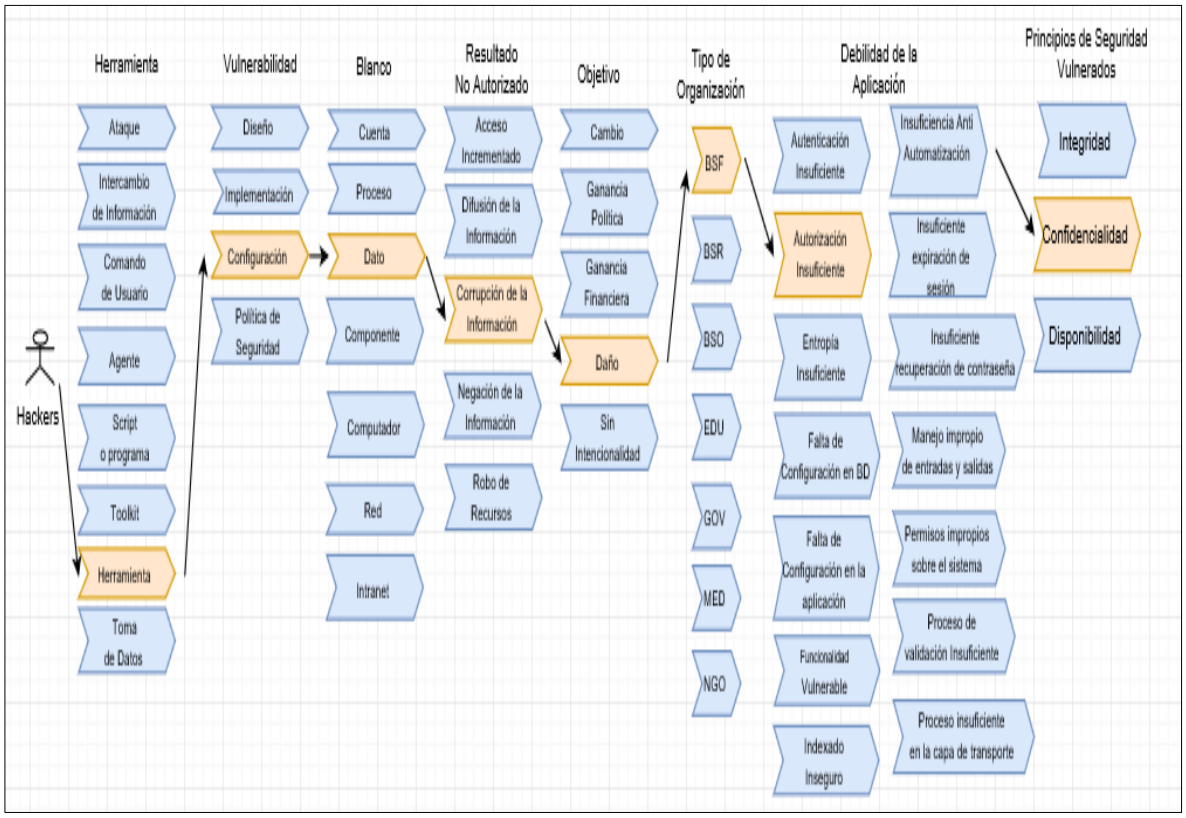
Segundo Moisés Toapanta Toapanta, Luis Enrique Mafla Gallegos, José Antonio Orizaga Trejo analizan los modelos de tendencias de gestión de identidades disponibles, autenticación, autorización y auditoría para mitigar las vulnerabilidades y riesgos de la información. Se determinó el desarrollo de este modelo para el Registro Civil de Ecuador en la siguiente fase. El método deductivo se utilizó en el análisis. Para el análisis de riesgos, la influencia del impacto económico en la gestión de la tecnología de la información y las comunicaciones, se deben considerar las metodologías resultantes, los diferentes niveles en la gestión de una TIC estratégica, táctica y operativa que debe considerar para el desarrollo del modelo y el prototipo del modelo Gestión de la identidad[42].

Xiong Li, Jianwei Niu, Muhammad Khurram Khan, Junguo Liao afirman que recientemente, An señaló las debilidades del esquema de autenticación remota de tres factores de Das y propuso un esquema de autenticación remota de tres factores basado en biométricos. El esquema de An mejora los problemas de seguridad de esquemas anteriores mientras que mantiene la eficacia. Sin embargo, después de un análisis detallado, se encuentra que el esquema de An tiene algunas debilidades, tales como vulnerable al ataque de negación de servicio (DoS) y ataque de falsificación, y no proporciona un acuerdo de clave de sesión. Con el fin de proporcionar alto nivel de seguridad y funciones más útiles, se diseña un nuevo esquema de autenticación robusta de tres factores basado en biométricos robusto con un acuerdo de clave utilizando criptosistema de curva elíptica[43].

Síntesis

Este método de ataque permite el acceso no autorizado a los sistemas de información, bases de datos y demás servicios, no hay seguridad de accesos de los usuarios en los procesos de autenticación y autorización, no hay control de seguridad en las conexiones de las empresas y las redes públicas o privadas. El principio de seguridad vulnerado es la confidencialidad (Véase Figura 24).

Figura 24. Clasificación de Taxonomía Falta de control de acceso de nivel de funciones



Fuente. El Autor

Cross-Site Request Forgery (CSRF)

Tanjila Farah, Moniruzzaman Shojol, Maruf Hassan, Delwar Alam afirman que XSS y CSRF son dos vulnerabilidades que tienen técnicas similares a las del asalto al Banco de Bangladesh. XSS y CSRF son la tercera y octava de las diez vulnerabilidades de aplicaciones Web de la lista OWASP desde 2013 hasta ahora. Ambos ataques violan la confianza de los usuarios en los sitios Web y los navegadores Web. Debido a la gravedad de estas vulnerabilidades, los especialistas en seguridad siempre han compartido su preocupación y advirtió a los desarrolladores Web[44].

E. Danny Álvarez; B. Daniel Correa; I. Fernando Arango afirma que el desarrollo de software y aplicaciones Web se han convertido en fundamentales en nuestras vidas. Millones de usuarios acceden a estas aplicaciones para comunicarse, obtener información y realizar transacciones. Sin embargo, estos usuarios están expuestos a muchos riesgos, comúnmente debido a la falta de experiencia del desarrollador en protocolos de seguridad. Aunque hay muchas investigaciones

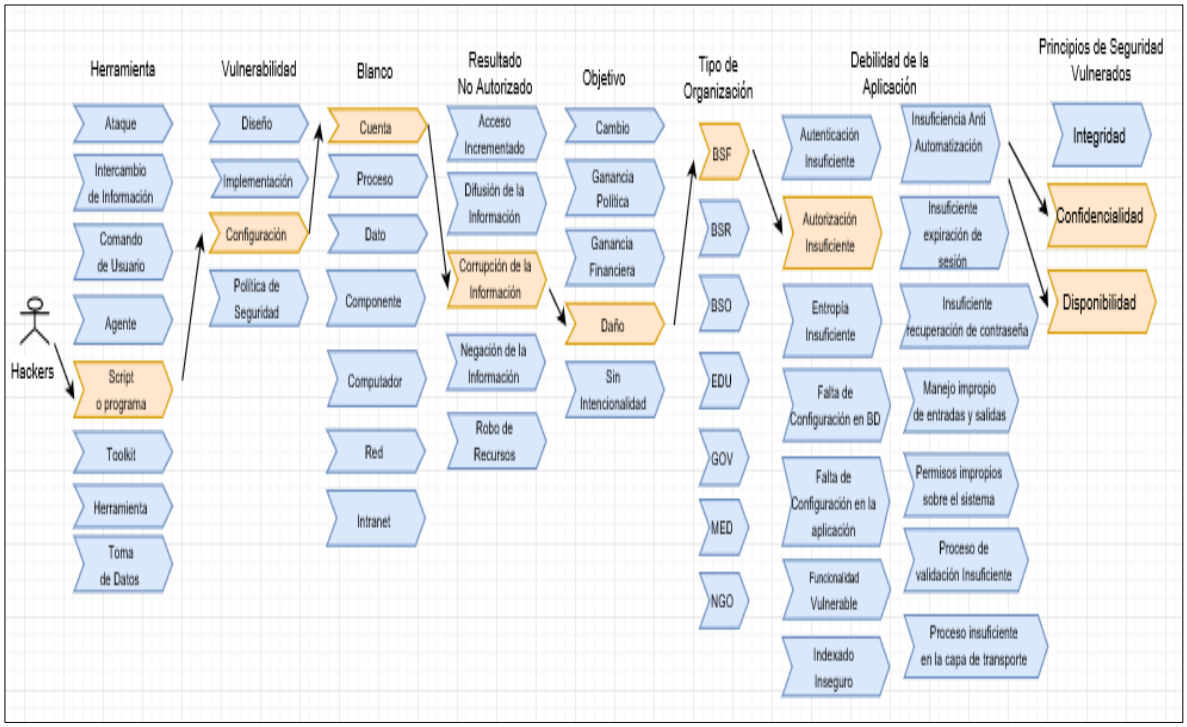
sobre la seguridad Web y la protección de hacking, hay un montón de sitios Web vulnerables. Este artículo se centra en analizar tres técnicas de hacking principales: XSS, CSRF e Inyección de SQL en un grupo representativo de sitios Web colombianos. El objetivo es obtener información sobre cómo las empresas y organizaciones colombianas dan (o no) relevancia a la seguridad y cómo podría ser afectado el usuario final[46].

Alexei Czeskis, Alexander Moshchuk, Tadayoshi Kohno, Helen J. Wang afirman que los ataques CSRF son una de las principales amenazas en la Web de hoy. Estos ataques explotan la autoridad ambiental en los navegadores (por ejemplo, cookies, estado de autenticación HTTP), torneado confundidos y causando efectos secundarios no deseados sobre los sitios Web vulnerables. Las defensas existentes contra las CSRF y su cobertura y/o facilidad de despliegue. En este artículo se presenta una solución de navegador/servidor, listas de referencias permitidas (ARL), que las direcciones raíz de los CSRFs y elimina la autoridad ambiental para participantes que quieren ser resistentes a los ataques de la CSRF[47].

Síntesis

La explotación CSRF son actividades ilícitas, acceso a cuentas privadas de usuarios. Los cibercriminales roban información, comprometen las cuentas sociales de usuarios o celebridades y publican los contenidos privados. Los principios de seguridad vulnerados son Disponibilidad y Confidencialidad (Véase Figura 25).

Figura 25. Clasificación de Taxonomía Cross-Site Request Forgery (CSRF)



Fuente. El Autor

Uso de componentes con vulnerabilidades conocidas

Sultan S. Alqahtani, Ellis E. Eghan, Juergen Rilling dicen que la globalización de la industria del software ha introducido un uso generalizado de los componentes del sistema a través de los límites tradicionales del sistema. Debido a esta reutilización global, las vulnerabilidades y preocupaciones de seguridad ya no están limitadas en su alcance a los sistemas individuales, sino que ahora pueden afectar a los ecosistemas de software global. Aunque las vulnerabilidades y las preocupaciones de seguridad conocidas se informan en bases de datos de vulnerabilidades especializadas, estos repositorios a menudo siguen siendo silos de información. En esta investigación, se introduce un enfoque de modelado, que elimina estos silos vinculando el conocimiento de seguridad con otros artefactos de software para mejorar la trazabilidad y la confianza en los productos de software[45].

Chen Chen, Lin Bai, Yehua Yang, Jinho Choi afirman que mantener las especificaciones de los requisitos actualizados cuando los sistemas evolucionan es una tarea manual y costosa. Los ingenieros de software tienen que pasar por todo el documento de requisitos y buscar los requisitos que se ven afectados por

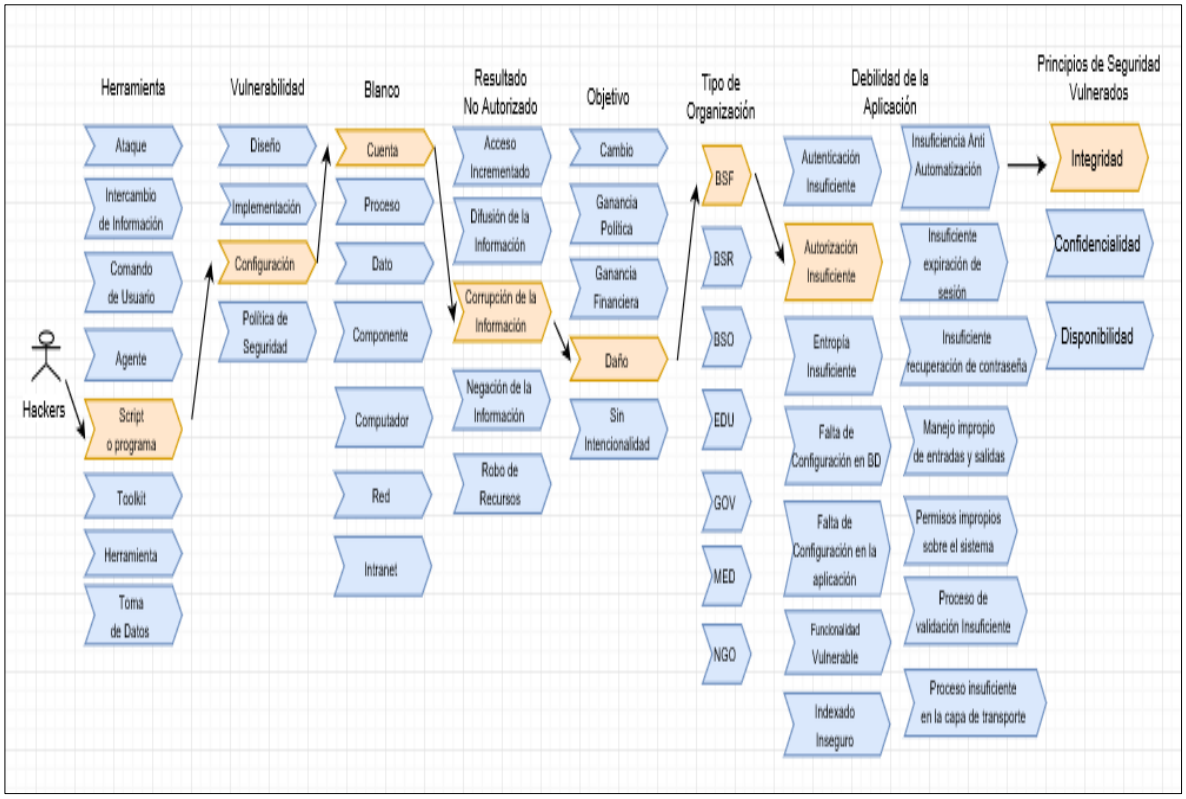
un cambio. En consecuencia, los ingenieros suelen aplicar cambios a la implementación directamente y dejar los requisitos sin cambios[50].

Wilhelm Kirchmayr, Michael Moser, Ludwig Nocke, Josef Pichler, Rudolf Tober afirman que en desarrollo de software se enfrentan con el problema de comprender y tomar el código fuente de otros desarrolladores. El desafío clave es entender la especificación subyacente implementada por el sistema de software. Recuperar esta comprensión es más difícil cuando el código fuente es la única fuente confiable de información, la documentación está obsoleta o sólo está presente en fragmentos, y los desarrolladores originales ya no están disponibles. Desafortunadamente, se encuentra frecuentemente situaciones de este tipo en sistemas de software científico y de ingeniería, desarrollados en la industria[48].

Síntesis

Esta vulnerabilidad es utilizada por el atacante para identificar los componentes débiles de una aplicación Web. Los componentes pueden ser frameworks, librerías y otros módulos del software. El principio de seguridad vulnerado es la Integridad (Véase Figura 26).

Figura 26. Clasificación de Taxonomía Uso de componentes con vulnerabilidades conocidas



Fuente. El Autor

Redireccionamiento y reenvío no validado

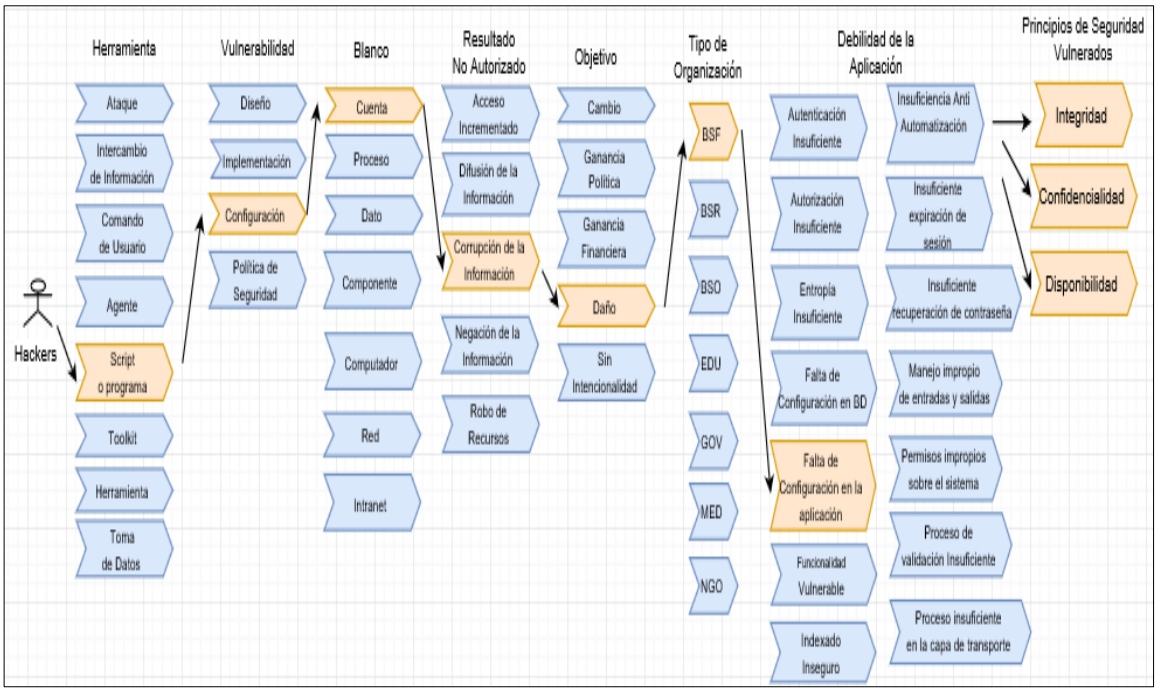
Rozita Kadar, Sharifah Mashita Syed-Mohamad, Nur'Aini Abdul Rashid dicen que la comprensión del programa es un proceso vital que implica mucho esfuerzo en el mantenimiento del software. Un desafío clave para los desarrolladores es comprender un sistema de software para ser mantenido ya que es difícil y consume mucho tiempo. Hoy en día, los sistemas de software han crecido en tamaño causando el aumento de tareas de los desarrolladores en la exploración y comprensión del código fuente[49].

Jing Wang, Hongjun Wu dicen que el re direccionamiento de URL es necesario en las aplicaciones web. Sin embargo, si se utiliza incorrectamente, podría dar lugar a ataques como el phishing. Estos re direccionamientos utilizados incorrectamente se denominan re direccionamientos y reenvíos no válidos (URF). Este documento prescribe un mecanismo para descubrir sistemáticamente URF vulnerabilidades en las aplicaciones web. La implementación del prototipo, que se llama Sistema de Detección de Reenvíos y Forwards no Validados (URFDS), utiliza una técnica de escaneo en caja negra para modificar URL y analizar la salida generada para identificar URF[51].

Síntesis

Esta vulnerabilidad es la redirección del usuario de la aplicación a otra página que no es segura. Frecuentemente en las aplicaciones se encuentra redirecciones hacia otras páginas de destino que los atacantes han seleccionado. Los principios de seguridad vulnerados son integridad, disponibilidad y confidencialidad (Véase Figura 27).

Figura 27. Clasificación de Taxonomía Redireccionamiento y reenvío no validado



Fuente. El Autor

6. **CONCLUSIONES**

Conclusiones relacionadas con el proceso de la revisión sistemática

El objetivo general del presente trabajo consistió en desarrollar una revisión sistemática basada en tipos de vulnerabilidad de Servicios Web REST la cual se realizó mediante la aplicación de una metodología de revisión sistemática que permitió identificar los principales tipos de vulnerabilidades que afectan la seguridad de los servicios Web REST.

La seguridad en los servicios web es un tema de gran actualidad e importancia con una creciente cantidad de publicaciones e investigaciones por lo que la aplicación de una metodología de revisión sistemática permitió seleccionar y sintetizar los artículos estrictamente relacionados con los estudios sobre vulnerabilidades de los servicios Web REST.

Adoptar una taxonomía para analizar las vulnerabilidades de los servicios Web REST permitió clasificar las amenazas por tipos de vulnerabilidades lo cual orientó a estudiar su estructura y los principios de seguridad vulnerados.

Conclusiones relacionadas con las vulnerabilidades de los servicios Web REST

La taxonomía propuesta identificó que los principios de seguridad vulnerados en los sistemas Web REST están de acuerdo a Cohen. Estos son:

Integridad.
Disponibilidad.
Confidencialidad.

La taxonomía propuesta identificó los siguientes tipos de vulnerabilidad en los servicios WebREST:

Inyección SQL,
Pérdida de autenticación y administración de sesión.
XSS.
Referencia directa a objetos inseguros
Configuración errónea de seguridad.
Exposición de datos sensibles.
Falta de control de acceso de nivel de funciones.
Cross-Site Request Forgery (CSRF).
Uso de componentes con vulnerabilidades conocidas.
Redireccionamiento y reenvío no validado.

Los tipos de ataque seleccionados permitieron realizar una clasificación a partir de una taxonomía establecida, lo que dio lugar a desarrollar la estructura de las debilidades identificando los aspectos más importantes y relevantes como el atacante, la herramienta, el tipo de vulnerabilidad, el blanco, el resultado no autorizado, el objetivo, el tipo de organización y el tipo de debilidad de la aplicación.

El uso de componentes con vulnerabilidades conocidas y el redireccionamiento y reenvío no validado son vulnerabilidades con escasa información.

La taxonomía que se utilizó permitió hacer la clasificación de los tipos de vulnerabilidad seleccionados que se encuentran en los servicios Web REST y permitió trazar la ruta que sigue cada tipo de vulnerabilidad cuando intenta conseguir su objetivo.

Las vulnerabilidades que se encontraron al utilizar la taxonomía son las mismas que expone OWASP, es decir, no se encontraron vulnerabilidades nuevas o desconocidas.

La taxonomía basada en procesos – orientación al evento es fácil de entender y utilizar, lo que permitió realizar la trazabilidad de cada tipo de vulnerabilidad.

Al identificar cada tipo de vulnerabilidad en los servicios Web REST se encontró que las categorías que se necesitan para clasificarla están contempladas en la taxonomía basada en procesos – orientación al evento.

Al identificar cada tipo de vulnerabilidad en los servicios Web REST se encontró que los ítems de la categoría que se necesitan para clasificarla están contemplados en la taxonomía basada en procesos –orientación al evento.

7. RECOMENDACIONES

Se recomienda para futuros trabajos buscar vulnerabilidades como uso de componentes con vulnerabilidades conocidas y redireccionamiento y reenvío no validado, que carecen de información relevante pero que son de cuidado.

Utilizar la taxonomía basada en procesos – orientación al evento en búsquedas futuras de vulnerabilidades.

REFERENCIAS

[1] K. HAUPT, Florian. LEYMANN, Frank. SCHERER, Anton . VUKOJEVIC-HAUPT, “A Framework for the Structural Analysis of REST APIs,” 2017 IEEE International Conference on Software Architecture (ICSA), pp. 55–58, 2017.

[2] B. Mehta, “Arquitectura REST,” RESTful Java Patterns and Best Practices. [Online]. Available: https://www.packtpub.com/mapt/book/Application+Development/9781783287963/3/ch03lvl1sec26/REST+architecture+components. [Accessed: 06-May-2017].

[3] B. S. M. Arezoo, MIRTALEBI, “A Cryptography Approach on Security Layer of Web Service,” 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT), p. 1.5, 2016.

[4] J. Y. C. YOUNG Su Jang, “Detecting SQL injection attacks using query result size,” En: Computers & Security, vol. 44, pp. 104–118, Apr-2014.

[5] S. S. KAR Debabrata, PANIGRAHI Suvasini, “SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM,” En: Computers & Security, vol. 60, pp. 206–225, 2016.

[6] S. A. ZAMANI Mazdak, MANAF Azizah Abd., “A Taxonomy of SQL Injection Detection and Prevention Techniques,” En: IEEE 2013 International Conference on Informatics and Creative Multimedia, pp. 53–56, 2013.

[7] R. S. R. S. K. NANHAY SINGH Mohit Dayal, “SQL Injection: Types, Methodology, Attack Queries and Prevention,” En: IEEE 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 2872–2876, 2016.

[8] B. L. APPELT Dennis, NGUYEN Cu D., “Behind an Application Firewall, Are we safe from SQL Injection Attacks?,” En: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), pp. 1–10, 2015.

[9] N. P. HANMANTHU B., RAM B. Raghu, “SQL Injection Attacks Prevention Based on Decision Tree Classification,” En: IEEE Sponsored 9th International Conference on Intelligent Systems and Control (ISCO) 2015, pp. 1–5, 2015.

[10] SRIVASTAVA Mahima, “Algorithm to Prevent Back End Database against SQL Injection Attacks,” En: IEEE 2014 International Conference on Computing for Sustainable Global Development (INDIACom), pp. 754–757, 2014.

[11] P. O. HULUKA Daniel, “Root Cause Analysis of Session Management and Broken Authentication Vulnerabilities,” En: IEEE World Congress on Internet Security (WorldCIS-2012), 2012.

[12] T. P. DACOSTA Italo, CHAKRADEO Saurabh, AHAMAD Mustaque, “One time cookies: Preventing session hijacking attacks with stateless authentication tokens,” En: ACM Transactions on Internet Technology (TOIT), vol. 12, 2012.

[13] S. K. R. NAGARAJA Arun, “A Session Key Utilization Based Approach For Memory Management in Wireless Networks,” En: ACM ICEMIS ’15 Proceedings of the The International Conference on Engineering & MIS 2015, 2014.

[14] A. K. G. RAHUL Kumar, INDRAVENI K, “Automated Session Fixation Vulnerability Detection in Web Applications using the Set-Cookie HTTP response header in cookies,” En: ACM SIN ’14 Proceedings of the 7th International Conference on Security of Information and Networks, 2014.

[15] G. S. CHAUDHARY Pooja, GUPTA B.B., “Cross-Site Scripting (XSS) Worms in Online Social Network (OSN): Taxonomy and Defensive Mechanisms,” En: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 2131–2136, 2016.

[16] T. H. BAOJIANG Cui, BAOLIAN Long, “Reverse Analysis Method of Static XSS Defect Detection Technique Based on Database Query Language,” En: IEEE 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computin, pp. 487–491, 2014.

[17] G. B. B. GUPTA Shashank, “Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment,” En: ScienceDirect Procedia Technology, pp. 1595–1602, 2016.

[18] S. S. V. Sharath Chandra, “BIXSAN: Browser Independent XSS Sanitizer for prevention of XSS attacks,” En: ACM SIGSOFT Software Engineering Notes, pp. 1–7, Sep-2011.

[19] G. R. DUCHÈNE Fabien, RAWAT Sanjay, RICHIER Jean-Luc, “LigRE: Reverse-Engineering of Control and Data Flow Models for Black-Box XSS Detection,” En: IEEE 2013 20th Working Conference on Reverse Engineering (WCRE), pp. 252–261, 2013.

[20] S. L. YU You, YANG Yuanyuan, GU Jian, “Analysis and suggestions for the Security of Web Applications,” IEn: EEE 2011 International Conference on Computer Science and Network Technology, 2011.

[21] W. K. ESHETE Birhanu, VILLAFIORITA Adolfo, “Early Detection of Security Misconfiguration Vulnerabilities in Web Applications,” En: IEEE 2011 Sixth International Conference on Availability, Reliability and Security, pp. 169–174, 2011.

[22] F. E. B. SULATYCKI Rohini, “Two threat patterns that exploit ‘Security misconfiguration’ and ‘Sensitive data exposure’ vulnerabilities,” En: ACM EuroPLoP ’15 Proceedings of the 20th European Conference on Pattern Languages of Programs, 2015.

[23] R. M. K. BAUER Lujo, GARRISS Scott, “Detecting and Resolving Policy Misconfigurations in Access-Control Systems,” En: ACM Transactions on Information and System Security (TISSEC), 2011.

[24] B. E. SHU Xiaokui, YAO Danfeng, “Privacy-Preserving Detection of Sensitive Data Exposure,” En: Privacy-Preserving Detection of Sensitive Data Exposure, vol. 10, no. 5, pp. 1092–1103, 2015.

[25] M. G. TK Ashwin Kumar, LIU Hong, THOMAS Johnson P, “Identifying Sensitive Data Items within Hadoop,” En: 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS), pp. 1308–1313, 2015.

[26] R. I. HABEEB Omotunde, “Mitigating SQL Injection Attacks Via Hybrid Threat Modelling,” En: 2015 2nd International Conference on Information Science and Security (ICISS), pp. 1–4, 2015.

[27] M. A. SADEGHIAN Amirmohammad, ZAMANI Mazdak, “A taxonomy of SQL Injection Attacks,” En IEEE 2013 Int. Conf. Informatics Creat. Multimed., pp. 269–273, 2013.

[28] S. T. P. DEEPA G., “Securing Web Applications from injection and Logic Vulnerabilities Approaches and Challenges,” En: ScienceDirect - Information and Software Technology, vol. 74, pp. 160–180, 2016.

[29] B. D. BURSZTEIN Elie, SOMAN Chinmay, “SessionJuggler: Secure Web Login From an Untrusted Terminal Using Session Hijacking,” En: ACM WWW’12 Proceedings of the 21st international conference on World Wide Web, pp. 321–330, 2012.

[30] N. U. T. FAGHANI Mohammad Reza, “A Study of XSS Worm Propagation and Detection Mechanisms in Online Social Networks,” En: IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, pp. 1815–1826, 2013.

[31] G. A. MALVIYA Vikas K., SAURAV Saket, “On Security Issues in Web Applications through Cross Site Scripting (XSS),” En: IEEE 2013 20th Asia-Pacific Software Engineering Conference, pp. 583–588, 2013.

[32] Y. P. YING Zhao, “Modeling the Propagation of XSS Worm on Social Networks,” En: 2013 IEEE Globecom Workshops (GC Wkshps), pp. 207–210, 2013.

[33] P. S. RAHUL Johari, “A Survey On Web Application Vulnerabilities (SQLIA,XSS) Exploitation and Security Engine for SQL Injection,” En: IEEE 2012 International Conference on Communication Systems and Network Technologies, 2012.

[34] S. M. KOIZUMI Daiki, MATSUDA Takeshi, “On the Automatic Detection Algorithm of Cross Site Scripting (XSS) with the Non-Stationary Bernoulli Distribution,” En: IEEE The 5th International Conference on Communications, Computers and Applications (MIC-CCA2012), pp. 131–135, 2012.

[35] “Ataque referencia insegura a objetos.” [Online]. Available: http://slideplayer.es/slide/5966797/. [Accessed: 06-May-2017].

[36] S. F. SALGADO Lenin, RON Mario, “ANÁLISIS DE RIESGOS DE LAS APLICACIONES WEB DE LA SUPERINTENDENCIA DE BANCOS Y SEGUROS, UTILIZANDO LAS RECOMENDACIONES TOP TEN DE OWASP,” En: Repositorio Institucional de la Universidad de las Fuerzas Armadas ESPE, 2014.

[37] S. M. FAHL Sascha, ACAR Yasemin, PERL Henning, “Why eve and mallory (also) love webmasters: a study on the root causes of SSL misconfigurations,” En: ACM ASIA CCS ’14 Proceedings of the 9th ACM symposium on Information, computer and communications security, 2014.

[38] P. S. XU Tianyin, ZHANG Jiaqi, HUANG Peng, ZHENG Jing, SHENG Tianwei, YUAN Ding, ZHOU Yuanyuan, “Do not blame users for misconfigurations,” En: SOSP ’13 Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pp. 244–259, 2013.

[39] T. R. CASALINO Matteo Maria, “Refactoring Multi-Layered Access Control Policies Through (De)Composition,” En: IEEE Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), pp. 243–250, 2013.

[40] S. C. BAUER Lujo, LIANG Yuan, REITER Michael K., “Discovering Access-Control Misconfigurations: New Approaches and Evaluation Methodologies,” En: CODASPY ’12 Proceedings of the second ACM conference on Data and Application Security and Privacy, pp. 95–104, 2012.

[41] B. A. R. LIU Fang, SHU Xiaokui, YAO Danfeng, “Privacy-Preserving Scanning of Big Content for Sensitive Data Exposure with MapReduce,” En: CODASPY ’15 Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, pp. 195–206, 2015.

[42] O. T. J. A. TOAPANTA TOAPANTA Segundo Moisés, MAFLA GALLEGOS Luis Enrique, “Analysis to define management of identities access control of security processes for the registration civil from Ecuador,” En: 2016 IEEE International Smart Cities Conference (ISC2), pp. 1–4, 2016.

[43] L. J. LI Xiong, NIU Jianwei, KHAN Muhammad Khurram, “Robust biometrics based three-factor remote user authentication scheme with key agreement,” En: Robust biometrics based three-factor remote user authentication scheme with key agreement, pp. 105–110, 2013.

[44] A. D. FARAH Tanjila, SHOJOL Moniruzzaman, HASSAN Maruf, “Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS & CSRF,” En: 2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), pp. 74–78, 2016.

[45] R. J. ALQAHTANI Sultan S., EGHAN Ellis E., “SV-AF – A Security Vulnerability Analysis Framework,” En: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 219–229, 2016.

[46] A. I. F. ALVAREZ E. Danny, CORREA B. Daniel, “An Analysis of XSS, CSRF and SQL Injection In Colombian Software And Web Site Development,” En: 2016 8th Euro American Conference on Telematics and Information Systems (EATIS), pp. 1–5, 2016.

[47] W. H. J. CZESKIS Alexei, MOSHCHUK Alexander, KOHNO Tadayoshi, “Lightweight Server Support for Browser-Based CSRF Protection,” En: ACM WWW ’13 Proceedings of the 22nd international conference on World Wide Web, pp. 273–284, 2013.

[48] T. R. KIRCHMAYR Wilhelm, MOSER Michael, NOCKE Ludwig, PICHLER Josef, “Integration of Static and Dynamic Code Analysis for Understanding Legacy Source Code,” En: Integration of Static and Dynamic Code Analysis for Understanding Legacy Source Code, pp. 543–552, 2016.

[49] R. N. A. KADAR Rozita, SYED MOHAMAD Sharifah Mashita, “Semantic-Based Extraction Approach for Generating Source Code Summary Towards Program Comprehension,” En: IEEE 2015 9th Malaysian Software Engineering Conference, no. 129–134, 2015.

[50] C. J. CHEN Chen, BAI Lin, YANG Yehua, “Identifying Outdated Requirements Based on Source Code Changes,” En: Requirements

Engineering Conference (RE), 2012 20th IEEE International, pp. 61–70, 2012.

[51] W. H. WANG Jing, “URFDS: Systematic Discovery of Unvalidated Redirects and Forwards in Web Application,” En: 2015 IEEE Conference on Communications and Network Security (CNS), pp. 697–698, 2015.

[52] B. D. CARVAJAL Carlos, “Extensión de taxonomía y tratamiento de valores faltantes sobre un repositorio de incidentes de seguridad informática,” En: Revista Ingeniería, vol. 18, no. 1, Bogotá, pp. 24–49, May-2013.

[53] “Capítulo1. Seguridad Informática: Conceptos Básicos.” [Online]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/jerez_l_ca/capitulo1.pdf.

[54] “Certsi.” [Online]. Available: https://www.certsi.es/respuesta-incidentes/rediris/taxonomia.

[55] A. D. Vicente, “OWASP Top 10 2013: actualización de los riesgos más extendidos asociados a las aplicaciones Web,” En: Revista SIC: ciberseguridad, seguridad de la información y privacidad, pp. 92–94, Sep-2013.

[56] J. E. M. Daza, “Revisión Sistemática.” [Online]. Available: http://download.docslide.net/documents/proceso-de-revision-sistematica.html. [Accessed: 19-May-2017].

[57] N.-A. A. ZANDI Javad, “LRBAC: Flexible Function-Level Hierarchical Role Based Access Control for Linux,” En: IEEE 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), pp. 29–35, 2015.

[58] Federick B. Cohen, Protection and Security on the Information Superhighway, John Wiley & Sons, New York, Estados Unidos, 1995

[59] Bayona Zulima Ortiz y Galindo Pulido Francisco Hacia una Taxonomía de Incidentes de Seguridad en Internet. [Online]. Disponible http://revistas.udistrital.edu.co/ojs/index.php/reving/article/view/2308/3126

[60] William Stallings, Network and Internetwork Security Principles and Practice, Prentice Hall, Englewood Cliffs, NJ, USA, 1995.

[61] Howard, John D and Longstaff, Thomas A. A Common Language for Computer Security Incidents. SANDIA REPORT SAND98-8667 Unlimited Release Printed October 1998

[62] Fielding, Roy Thomas, Architectural Styles and the Design of Network-based Software Architectures DISSERTATION submitted in partial satisfaction of the requirements for the degree of DOCTOR OF PHILOSOPHY in

Information and Computer Science. 2000.

[63] W3C Consortium. Web Services Architecture. [En línea] 11 de Febrero de 2004. [Citado el: 25 de septiembre de 2017.] https://www.w3.org/TR/ws-arch/#whatis