UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ
INSTITUTO DE INGENIERÍA Y TECNOLOGÍA

Departamento de Ingeniería Eléctrica y Computación
Programa de Maestría en Cómputo Aplicado

"Marco de Referencia y Herramienta para La Evaluación de Seguridad de Aplicaciones en SaaS"

Tesis para obtener el grado de

Maestro en Cómputo Aplicado

Ing. Miguel Ángel Díaz de León Guillén

"Becado por el Consejo Nacional de Ciencia y Tecnología"

Bajo la Dirección del
Dr. Víctor Manuel Morales Rocha

Y la Codirección del
Mtro. Luis Felipe Fernández Martínez

Ciudad Juárez, Chihuahua, febrero del 2021

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACyT-México) le agradezco la beca recibida durante la maestría.

A la Universidad Autónoma de Ciudad Juarez por brindarme un lugar para estudiar la maestría. Al Laboratorio Nacional de Tecnologías de Información, por la infraestructura e instalaciones disponibles para este trabajo.

Expreso mi profundo agradecimiento a mi tutor de tesis Dr. Víctor Manuel Morales Rocha por su instrucción y colaboración durante el desarrollo de este proyecto. Al Mtro. Luis Felipe Fernández Martínez le agradezco sus valiosos aportes durante la maestría.

A mis compañeros de la maestría, Ing. César Javier Maldonado Flores, Ing. Samuel Martín Martínez Magdaleno e Ing. Adrián Hernández Rivas, les agradezco su apoyo y sus consejos que permitieron continuar con la maestría.

Índice

Introducción

El presente trabajo es un desarrollo tecnológico para que los desarrolladores elaboren aplicaciones más seguras para una arquitectura SaaS. Para llevarlo a cabo, se realizaron una serie de procedimientos ligados entre sí. Primeramente, se realizó una revisión sistemática para conocer los ataques más comunes que pueden ocurrir en una aplicación en SaaS, además, se obtuvo los riesgos que conlleva cada ataque y las medidas de seguridad necesarias para mitigarlos. Posteriormente, se desarrolló una estimación de riesgos para establecer las probabilidades de ocurrencia, nivel de riesgo y de impacto de cada ataque. Con los datos recaudados por la revisión sistemática y la estimación de riesgos se elaboró un marco de seguridad. El cual, funciona como guía para el desarrollador, debido a que permite la evaluación de seguridad de sus aplicaciones con base a los ataques que pueden ocurrir en SaaS y muestra las medidas que se pueden implementar para hacerlas más seguras. Por último, se desarrolló una herramienta alojada en la nube para conocer los ataques específicos que pueden ocurrir en una aplicación en SaaS.

La herramienta genera reportes de las vulnerabilidades encontradas, los ataques que estas pueden desencadenar y las medidas de seguridad que se pueden implementar para prevenirlos. El desarrollador puede buscar en el marco de seguridad los ataques mostrados en los reportes otorgados por la herramienta para conocer los riegos y el nivel de impacto en caso de que un ataque resulte exitoso, además en el marco de seguridad es posible encontrar medidas adicionales para asegurar las aplicaciones. Por lo tanto, el marco de seguridad y la herramienta en conjunto permiten la detección de vulnerabilidades en aplicaciones en SaaS, los ataques que pueden desencadenar, los riegos, el nivel de impacto y las medidas de seguridad que se pueden aplicar para mitigar los problemas de seguridad.

Primeramente, en el capítulo 1 se describe el problema que trata este trabajo y sus antecedentes, además se plantean los objetivos del proyecto, así como las metas y el impacto de este. Posteriormente, en el capítulo 2 se presenta el marco teórico. Por otra parte, en el capítulo 3 se describe el desarrollo de la revisión sistemática, estimación de riesgos y del marco de seguridad. Por otro lado, en el capítulo 4 se presentan los procedimientos elaborados para el desarrollo de la herramienta para el análisis de aplicaciones en SaaS en busca de vulnerabilidades. Por último, en el capítulo 5 se presentan las conclusiones del proyecto y trabajo futuro propuesto por el autor.

CAPITULO 1. Planteamiento del problema

1.1 Descripción del problema

El cómputo en la nube es un modelo de negocio donde un proveedor ofrece servicios de almacenamiento, cómputo y uso de recursos computacionales a través de internet [1]. Este modelo permite a las empresas ahorrar costos de administración y mantenimiento de infraestructura tecnológica. Sin embargo, el principal desafío de este modelo es la protección de los datos de las empresas [2].

Existen vulnerabilidades en las aplicaciones en la nube que afectan a la seguridad de la información, las cuales suelen ser difíciles de detectar. Una vulnerabilidad es una debilidad en la aplicación, que puede ser un error de implementación o una falla de diseño, que permite a un atacante causar daño al usuario de la aplicación y obtener privilegios adicionales [3]. Al ejecutar una aplicación en la nube con una o varias vulnerabilidades la información alojada en el servidor virtual puede ser explotada por una persona malintencionada. Por lo tanto, una vulnerabilidad puede causar pérdida de información a los usuarios de la nube.

Se han propuesto algunos trabajos que tratan, al menos en parte, de solucionar los problemas de vulnerabilidades en la nube. En el trabajo descrito en [4], se propone maximizar la satisfacción de seguridad de los servicios que utilizan los usuarios. La información de los usuarios es alojada en cierto servidor dependiendo de los requerimientos de seguridad que el mismo usuario solicita. Sin embargo, algunas de sus limitantes son la falta de medición de la calidad de servicio y el rendimiento. Por su parte, en el trabajo presentado en [5], se describe un modelo seguro y efectivo que permite monitorear los datos en la nube en tiempo real; además el sistema ajusta el ciclo de monitoreo para proteger de forma autónoma los datos. Sin embargo, carece de un procedimiento a realizar en caso de un ataque. En la investigación descrita en [6], se propone un modelo de seguridad en la nube y un marco de seguridad que identifique los desafíos de seguridad en la computación en la nube. Se presenta una serie de amenazas y ataques de seguridad en la nube, así como el método para mitigar el problema. El modelo de seguridad, enfocado en estabilidad y seguridad, es propuesto para afrontar las amenazas, en cambio, el marco de seguridad describe las tecnologías de seguridad necesarias entre cada componente de la nube. Esta investigación carece de una administración de riesgos asociados al cómputo en la nube.

Por otro lado, en el trabajo presentado en [7], se propone un marco de arquitectura de nube híbrida eficiente junto con comunicación de Li-Fi para una red Internet de las cosas (IoT por sus siglas en inglés) centrada en el ser humano, para abordar los problemas de escalabilidad, eficiencia energética, movilidad, cuellos de botella de ancho de banda y retardo de latencia. También presenta la arquitectura de la nube local para reducir el retraso de la latencia y el costo de ancho de banda y para mejorar la eficiencia, la seguridad, la confiabilidad y la disponibilidad. Por su parte, en la investigación descrita en [8], se identifican los marcos de riesgo para evaluar las aplicaciones de software como servicio (SaaS) en el contexto del usuario empresarial y ayudarlos a elegir los marcos de riesgo correctos. En contraparte, carece de una revisión automática de las vulnerabilidades existentes. Por último, en el trabajo presentado en [9], se investigan los factores de seguridad críticos que influyen en la decisión de adoptar cómputo en la nube por parte de las agencias gubernamentales sauditas, además, se propuso un marco para tres categorías, categoría de factores sociales, categoría de riesgos de seguridad en la nube y beneficios de seguridad en la nube percibidos que incluyen características de seguridad en la nube.

Delimitación del problema

Entre los modelos de servicio proporcionados por la nube, el modelo de software como servicio (SaaS) ha tenido el mayor crecimiento. Este modelo de servicio es una opción atractiva para las organizaciones, ya que pueden transferir parte o la totalidad de sus funciones de TI a un proveedor de servicios en la nube. Sin embargo, todavía existe cierta incertidumbre sobre la decisión de realizar una migración de todos los datos a la nube, principalmente por motivos de seguridad. El modelo SaaS no solo hereda los problemas de seguridad de una aplicación tradicional, sino que existen ataques y vulnerabilidades únicos para una arquitectura SaaS. Además, algunos de los ataques en este entorno son más devastadores debido a la naturaleza de los recursos compartidos en el modelo SaaS. Algunos de estos ataques y vulnerabilidades aún no son bien conocidos por los diseñadores y desarrolladores de software. Esta falta de conocimiento tiene consecuencias negativas ya que puede exponer datos sensibles de usuarios y organizaciones [10].

1.2 Justificación

La cantidad de aplicaciones que se están implementando en SaaS va en aumento. A diferencia de una aplicación tradicional en la que el software pertenece al cliente, en el modelo SaaS la aplicación es administrada por el proveedor de la nube. Además, la industria de TI puede cambiar el modelo de venta de aplicaciones con el uso de equipos en la nube con la provisión de servicios SaaS, reduciendo

los costos de marketing. Sin embargo, los clientes de la nube no tienen la gestión de la infraestructura de la nube. Por lo tanto, el usuario debe conocer las medidas de seguridad que ha implementado el proveedor de la nube [11] [12] [13].

El cómputo en la nube hereda problemas de seguridad de sistemas en sitio, redes y, además, vulnerabilidades en los servicios web, especialmente para el modelo SaaS en el que el software se concede a través de la nube [13][14]. El principal obstáculo para que una organización migre su software a SaaS es la incertidumbre de la seguridad de sus datos debido al temor a la fuga de información [14].

Actualmente no existen un marco de seguridad que permita la identificación de vulnerabilidades en aplicaciones alojadas en la nube bajo el modelo de software como servicio. Los trabajos descritos en la sección anterior no cubren los desarrollos para SaaS, por lo que esta investigación tiene como finalidad reducir brechas de seguridad al identificar vulnerabilidades en aplicaciones desarrolladas bajo el modelo de software como servicio.

Para lograr el cumplimiento del objetivo propuesto, se va a desarrollar un marco de seguridad acompañado de una herramienta dirigida a los programadores de aplicaciones en la nube. Este marco de seguridad dará a conocer una lista de recomendaciones que un desarrollador pueda tomar como referencia para diseñar y desarrollar aplicaciones para la nube más seguras. La herramienta facilitará a los desarrolladores la evaluación de seguridad de sus aplicaciones. Además, mostrará el riesgo y el nivel de impacto.

1.3 Objetivo general

Desarrollar un marco de referencia y una herramienta que permitan evaluar y mejorar la seguridad en el diseño de aplicaciones para SaaS.

1.4 Objetivos específicos

- Desarrollo de una estimación de riegos que permita conocer los peligros que corre una aplicación en SaaS.
- Desarrollo de un marco de seguridad que los desarrolladores puedan utilizar como referencia para el diseño seguro de sus aplicaciones en SaaS.
- Desarrollo de una herramienta para la evaluación de seguridad en aplicaciones implementadas en SaaS.

1.5 Metas

La presente investigación tiene las siguientes metas:

- Un nuevo marco de seguridad para los desarrolladores de aplicaciones en la nube.
- Una nueva herramienta para el análisis de seguridad en aplicaciones en la nube.

1.6 Impacto

El marco de seguridad y la herramienta impactan de la siguiente forma:

- Diseño de aplicaciones para SaaS más seguras.
- Detección de vulnerabilidades en aplicaciones en SaaS.
- El desarrollador conoce los ataques que pueden ocurrir en sus aplicaciones en SaaS.
- El desarrollador conoce las medidas de seguridad que debe implementar para hacer sus aplicaciones más seguras.

CAPITULO 2. Marco teórico

2.1 Seguridad informática

La seguridad informática es el área que se especializa en la protección de los recursos de cómputo y todo lo relacionado, sobre todo la información contenida en los dispositivos [15]. Para asegurar la información es necesario contar con tres elementos, los cuales son (1) confidencialidad, término utilizado para evitar la divulgación de información a personas no autorizadas, (2) la integridad, término utilizado para evitar la alteración de los datos de forma no autorizada, y (3) la disponibilidad, término utilizado para asegurar el acceso a los datos, (también llamada como la tríada de la CIA) [16] [17] mientras se mantiene la perspectiva en la implementación eficiente de políticas, todo ello sin complicar la productividad de la organización [18].

2.1.1 Firewall

En informática, un firewall, conocido también como corta fuegos, es un sistema de seguridad de red que monitorea y controla el tráfico de red entrante y saliente según reglas de seguridad predeterminadas [19]. Un firewall generalmente establece una barrera entre una red interna confiable y una red externa no confiable, como Internet [20]. El término firewall originalmente se refería a un muro destinado a limitar un incendio dentro de una línea de edificios adyacentes [21]. Los usos posteriores se refieren a estructuras similares, como la lámina de metal que separa el compartimento del motor de un vehículo o avión del compartimento de pasajeros. El término se aplicó a fines de la década de 1980 a la tecnología de red que surgió cuando Internet era bastante nuevo en términos de uso global y conectividad [22].

Los predecesores de los firewalls para la seguridad de la red fueron los enrutadores utilizados a fines de la década de 1980, porque separaron las redes entre sí, deteniendo así la propagación de problemas de una red a otra [20]. A continuación, se presenta las generaciones existentes en la historia de los firewalls:

- Primera generación: filtros de paquetes. El primer tipo de firewall de red informado se denomina filtro de paquetes. Los filtros de paquetes actúan inspeccionando los paquetes transferidos entre computadoras. Cuando un paquete no coincide con el conjunto de reglas de filtrado del filtro de paquetes, el filtro de paquetes descarta (descarta silenciosamente) el paquete o rechaza el paquete (lo descarta y genera una notificación de Protocolo de mensajes de control de Internet para el remitente) de lo contrario se le permite pasar [20].

- Segunda generación: filtros con estado. Los cortafuegos de segunda generación realizan el trabajo de sus predecesores de primera generación, pero también mantienen el conocimiento de conversaciones específicas entre puntos finales al recordar qué número de puerto utilizan las dos direcciones IP en la capa 4 (capa de transporte) del modelo OSI para su conversación, lo que permite el examen del intercambio general entre los nodos. Este tipo de firewall es potencialmente vulnerable a los ataques de denegación de servicio que bombardean el firewall con conexiones falsas en un intento de abrumar el firewall al llenar su memoria de estado de conexión [23].
- Tercera generación: capa de aplicación. El beneficio clave del filtrado de la capa de aplicación es que puede comprender ciertas aplicaciones y protocolos (como el Protocolo de transferencia de archivos (FTP), el Sistema de nombres de dominio (DNS) o el Protocolo de transferencia de hipertexto (HTTP)). Esto es útil, ya que es capaz de detectar si una aplicación o servicio no deseado está intentando eludir el firewall utilizando un protocolo no permitido en un puerto permitido, o detectar si se está abusando de un protocolo de alguna manera perjudicial [24].
- Cortafuegos de próxima generación: forma parte de la tercera generación de tecnología de firewall, que combina un firewall tradicional con otras funciones de filtrado de dispositivos de red, como un firewall de aplicación que utiliza inspección profunda de paquetes en línea (DPI), un sistema de prevención de intrusiones (IPS). También podrían emplearse otras técnicas, como la inspección de tráfico cifrado TLS / SSL, el filtrado de sitios web, la gestión de QoS / ancho de banda, la inspección antivirus y la integración de gestión de identidad de terceros (es decir, LDAP, RADIUS, Active Directory) [25].

Los firewalls generalmente se clasifican como basados en la red o en el host. Los firewalls basados en la red están ubicados en las compuertas de puerta de enlace de LAN, WAN e intranets. Los firewalls basados en host se colocan en el nodo de red en sí. El cortafuegos basado en host puede ser un demonio o servicio como parte del sistema operativo o una aplicación de agente como la seguridad o protección de punto final. Cada uno tiene ventajas y desventajas. Sin embargo, cada uno tiene un papel en la seguridad por capas [26]. A continuación, se enlistan los tipos de firewalls existentes:

- Capa de red. Los firewalls de la capa de red, también llamados filtros de paquetes operan a un nivel relativamente bajo de la pila de protocolos TCP / IP, no permitiendo que los paquetes pasen a través del firewall a menos que coincidan con el conjunto de reglas establecido. El

administrador del firewall puede definir las reglas; o pueden aplicarse reglas predeterminadas [26].

- Nivel de aplicación. Los firewalls de la capa de aplicación funcionan en el nivel de aplicación de la pila TCP / IP y pueden interceptar todos los paquetes que viajan o forman una aplicación. Incluyen otros paquetes que generalmente los descartan sin reconocer al remitente [26].
- Proxies. Un servidor proxy puede actuar como un cortafuegos respondiendo a los paquetes de entrada en la forma de una aplicación, mientras bloquea otros paquetes. Un servidor proxy es una puerta de enlace de una red a otra para una aplicación de red específica, en el sentido de que funciona como un proxy en nombre del usuario de la red [26].
- Traducción de direcciones de red. El cortafuegos a menudo tiene la funcionalidad de traducción de direcciones de red (NAT), y el host protegido detrás de un cortafuegos suele tener direcciones en el rango de direcciones privadas, como se define en RFC 1918. El cortafuegos a menudo tiene esa funcionalidad para ocultar la verdadera dirección del host protegido. Originalmente, la función NAT se desarrolló para abordar el número limitado de direcciones enrutables IPv4 que podrían usarse o asignarse a empresas o individuos, así como reducir tanto la cantidad como el costo de obtener suficientes direcciones públicas para cada computadora en una organización. Aunque NAT por sí mismo no se considera una característica de seguridad, ocultar las direcciones de los dispositivos protegidos se ha convertido en una defensa de uso frecuente contra el reconocimiento de la red [26].

2.1.2 Malware

Malware (software malicioso) es cualquier software diseñado intencionalmente para causar daños a una computadora, servidor, cliente o red informática [27]. El malware puede clasificarse en las siguientes categorías:

- Virus. Un virus informático es un tipo de programa informático que, cuando se ejecuta, se replica modificando otros programas informáticos e insertando su propio código [28]. Los creadores de virus utilizan engaños de ingeniería social y explotan el conocimiento detallado de las vulnerabilidades de seguridad para infectar inicialmente los sistemas y propagar el virus. La gran mayoría de los virus se dirigen a sistemas que ejecutan Microsoft Windows [29].
- Ransomware. El ransomware es un tipo de malware de criptovirología que amenaza con publicar los datos de la víctima o bloquear permanentemente el acceso a ellos a menos que

se pague un rescate. Si bien algunos ransomware simples pueden bloquear el sistema de una manera que no sea difícil de revertir para una persona conocedora, el malware más avanzado utiliza una técnica llamada extorsión criptoviral, en la que encripta los archivos de la víctima, haciéndolos inaccesibles y exige un pago de rescate para descifrarlos [30].

- Caballo de Troya. En informática, un troyano, es cualquier malware que engaña a los usuarios de su verdadera intención. El término se deriva de la historia griega antigua del engañoso caballo de Troya que condujo a la caída de la ciudad de Troya [31].
- Rootkit. Un rootkit es un "kit" que consiste en programas pequeños y útiles que permiten a un atacante mantener el acceso a la raíz, el usuario más poderoso en una computadora. En otras palabras, un rootkit es un conjunto de programas y código que permite una presencia permanente o constante e indetectable en una computadora [32].
- Puerta trasera. Una puerta trasera es una abertura creada a través del firewall y otras medidas de protección de seguridad que permiten que un atacante acceda a la computadora o la red sin ser detectado. También se puede usar para convertir una PC en una botnet, una computadora que forma parte de una gran red de PC que los piratas informáticos se hacen cargo de los ataques de denegación de servicio [33].

2.1.3 Antivirus

El software antivirus se desarrolló originalmente para detectar y eliminar virus informáticos, de ahí el nombre. Sin embargo, con la proliferación de otros tipos de malware, el software antivirus comenzó a proporcionar protección contra otras amenazas informáticas. En particular, el software antivirus moderno puede proteger a los usuarios de: objetos ayudantes de navegador maliciosos (BHO por sus siglas en inglés), secuestradores de navegador, ransomware, keyloggers, puertas traseras, rootkits, troyanos, gusanos, LSP maliciosos, marcadores, herramientas de fraude, adware y spyware [34].

En el estudio presentado por Frederick B. Cohen en 1987 se demuestra que no existe ningún algoritmo que pueda detectar perfectamente todos los virus posibles [35]. Sin embargo, actualmente existen varios métodos que un antivirus puede usar para identificar malware, los cuales son los siguientes:

- Detección basada en firma. El software antivirus tradicional depende en gran medida de las firmas para identificar malware. Sustancialmente, cuando un malware llega a las manos de una empresa antivirus, es analizado por investigadores de malware o por sistemas de análisis dinámico. Luego, una vez que se determina que es un malware, se extrae una firma adecuada del archivo y se agrega a la base de datos de firmas del software antivirus [36].

- Heurística. El análisis heurístico es un método empleado por muchos programas antivirus de computadora diseñados para detectar virus informáticos previamente desconocidos, así como nuevas variantes de virus que ya están en la "naturaleza" [37].
- Técnicas de minería de datos. Uno de los últimos enfoques aplicados en la detección de malware. Los algoritmos de minería de datos y aprendizaje automático se utilizan para tratar de clasificar el comportamiento de un archivo (ya sea malicioso o benigno) dada una serie de características del archivo, que se extraen del archivo en sí [38].

2.2 Cómputo en la nube

La computación en la nube (CC) permite el uso de recursos bajo demanda para ofrecer disponibilidad y escalabilidad a bajo costo [39][40]. En la nube, varios clientes se conectan a un servidor compartido y la ventaja de este modelo es que está bajo demanda, lo que significa que los clientes solo pagan por el uso [39]. Otros beneficios son algunos servicios gratuitos, elasticidad, fácil acceso, entre otros [41][42]. Además, la computación en la nube reduce los gastos y facilita la administración del sistema [43].

El uso del cómputo en la nube permite a los usuarios concentrarse en su negocio en lugar de enfocarse en los problemas de TI. La tecnología base para la nube es la virtualización, la cual separa un dispositivo físico en varios virtuales fácilmente administrables mediante aplicaciones. Gracias a esto, los recursos no utilizados se pueden emplear de una manera más eficiente. La virtualización permite acelerar los procesos de TI y la reducción de costos al aumentar la eficacia en el uso de la infraestructura al otorgar recursos a pedido. Por lo tanto, se aceleran los procesos, existe una reducción de costos de mano de obra y se reducen los errores humanos [44].

La nube tiene cuatro características básicas [45]:

- Escalabilidad: la nube opta por utilizar una arquitectura escalable. La escalabilidad significa que se agregan unidades de hardware para traer más recursos a la nube. Sin embargo, esta característica está en compensación con la seguridad del software. Por lo tanto, la escalabilidad puede facilitar la representación de la nube y puede aumentar los delincuentes que acceden al almacenamiento en la nube y los centros de datos de forma ilegítima.
- Disponibilidad: Los servicios, la plataforma y los datos son accesibles en cualquier momento y lugar. La nube se expone potencialmente a mayores amenazas de seguridad de software, principalmente cuando la nube se basa en Internet en lugar de la plataforma propia de una organización.

- Copia de seguridad automática: día tras día, muchos fabricantes de dispositivos electrónicos confían en el modelo de la computación en la nube y están incorporando cada vez más este paradigma en sus productos, ya que ofrece las características de comunicación y respaldo automático de la información.
- Agregar valor y servicios adicionales al usuario, como la capacidad de sincronizar entre amigos en sitios de redes sociales como Facebook.

Actualmente hay cuatro posibles modelos de implementación en la nube y cada uno de ellos tiene algunos problemas de seguridad particulares los cuales se describen a continuación [13][40] [46]:

- Nube pública: Las cuales prestan servicios a través de Internet. Por lo general se basa en pago por uso con capacidad de contrarrestar los picos de demanda. Este tipo de nubes se consideran menos seguros porque es más difícil proteger los datos contra ataques malintencionados.
- Nube privada: Son administradas por el propietario o por un tercero donde la infraestructura puede estar tanto dentro como fuera de la organización. De esta manera es posible ajustar los niveles de seguridad de acuerdo con las necesidades de la empresa. Por lo que, la administración y mantenimiento son más sencillos, la seguridad es más alta y existe un mejor control en la infraestructura.
- Nube comunitaria: La nube comunitaria es complicada para las empresas, lo que dificulta la definición de los requisitos de seguridad.
- Nube híbrida: Este modelo representa la fusión de dos o más modelos de nube. Desafortunadamente este tipo de nube hereda todos los problemas de seguridad de los demás modelos de implementación.

La arquitectura de la nube se puede dividir en 4 capas, las cuales se muestran a continuación [16].

- La capa de hardware: La funcionalidad de esta capa es la administración de los recursos físicos, como pueden ser los servidores, routers, conmutadores entre otros. Esta capa suele implementarse en centros de datos, los problemas más comunes suceden en la configuración de hardware, gestión de tráfico, y tolerancia a fallas.
- La capa de infraestructura: También llamada capa de virtualización crea recursos de almacenamiento y computo empleando tecnologías de virtualización. Está capa es fundamental para el computo en la nube debido a que ciertas características como la asignación dinámica de recursos se da gracias al empleo de la virtualización.

- La capa de plataforma: Alojada sobre la capa de infraestructura, formada de sistemas operativos y marcos de aplicación. Su función es minimizar la carga en el momento de alojar contenedores de aplicaciones en máquinas virtuales.
- La capa de aplicación: Alojada sobre la capa de plataforma, consta de aplicaciones de nube las cuales pueden utilizar la función de escalado automático para obtener mejor rendimiento, mayor disponibilidad y reducción de costos de operación.

2.2.1 Infraestructura como servicio

En modelo de infraestructura como servicio (IaaS) se le proporciona al usuario mediante un pago por uso, procesamiento, almacenamiento, redes, entre otros donde es posible la implementación e instalación de sistemas operativos y aplicaciones. El usuario no cuenta con la administración ni control de la infraestructura de la nube, aun así, es posible administrar los sistemas operativos, aplicaciones implementadas, y gozar de un control limitado de recursos de red como un cortafuegos [47] [48].

2.2.2 Plataforma como servicio

El modelo de plataforma como servicio (PaaS) consiste en la capacidad de proporcionar al usuario la capacidad de implementación de entornos de desarrollo de aplicaciones y alojamiento. El usuario no controla ni administra la red, servidores, sistemas operativos ni el almacenamiento, aun así, puede controlar las aplicaciones y los ajustes del entorno de alojamiento de sus aplicaciones [48]. Este modelo de servicio es de los más populares debido a que proporciona una pila de desarrollo al poder administrar bases de datos y aplicaciones al lado de servicios de desarrollo [47].

2.2.3 Software como servicio

El modelo de software como servicio proporciona al usuario final el uso de aplicaciones desarrolladas por algún proveedor que se ejecutan en la infraestructura de la nube. Es posible el acceso a las aplicaciones mediante navegadores web o interfaces de un programa. El usuario no tiene control ni administración sobre la infraestructura de la nube como los sistemas operativos, almacenamiento, servidores, red entre otros. Sin embargo, es posible la aplicación de algunos parámetros de configuración limitados dentro de la aplicación para el usuario [47] [48]. A diferencia de una aplicación tradicional en sitio una aplicación en SaaS corre bajo el modelo PaaS [49].

2.4.4 Proveedores de nube

- **Servicios web de Amazon (AWS).** AWS es una colección de servicios que conforman una plataforma de computación en la nube, que se basa en 11 regiones geográficas de todo el mundo. Los servicios principales y más conocidos son Amazon EC2 (Elastic Compute Cloud) y Amazon S3 (Simple Storage Service). Los productos se ofrecen a empresas grandes y pequeñas como un servicio para proporcionar una gran capacidad informática más rápida y económica que la empresa cliente que construye y mantiene una granja de servidores físicos reales. AWS maneja automáticamente los detalles, como el aprovisionamiento de recursos, el equilibrio de carga, el escalado y la supervisión. Se pueden crear aplicaciones en PHP, Java, Python, Ruby, node.js, .NET, Go o en un contenedor Docker que se ejecuta en un servidor de aplicaciones con una base de datos. Un entorno que use la configuración predeterminada ejecutará una única micro instancia de Amazon EC2 y un Elastic Load Balancer. Se agregarán instancias adicionales si es necesario, para manejar los picos en la carga de trabajo o el tráfico [50].

- **Azure.** Azure es una plataforma de computación en la nube, que permite a los desarrolladores publicar aplicaciones web que se ejecutan en diferentes marcos, escritos en diferentes lenguajes de programación como cualquiera. Lenguaje NET, node.js, php, Python y Java. Azure Web Sites admite un asistente de creación de sitios web que se puede usar para crear un sitio en blanco o usar uno de los varios sitios preconfigurados. Los desarrolladores pueden agregar o modificar el contenido del sitio web a través de múltiples métodos de implementación: TFS, FTP, CodePlex, GitHub, Dropbox, Bitbucket, Mercurial o git. Los desarrolladores pueden seleccionar el lugar donde se alojará su sitio web desde varios centros de datos de Microsoft en todo el mundo. Azure Traffic Manager enruta el tráfico de forma manual o automática entre sitios web en diferentes regiones. Los sitios web están alojados en IIS, ejecutándose en una versión personalizada de Windows Server. El componente relacionado con IoT se llama IoT Hub, que puede comunicarse con dispositivos con protocolos como MQTT, AMQP y HTTP, pero también es posible implementar otros protocolos [50].

- **Heroku.** Heroku ha estado en desarrollo desde 2007, comenzando con soporte para Ruby y agregando soporte para muchos idiomas a través de los años, como Java, Node.js, Scala, Clojure, Python, PHP y Perl. Heroku fue adquirido por Salesforce.com en 2010, como

subsidiaria. Los servicios de Herokus se ejecutan en los sistemas en la nube de Amazon. Desde el punto de vista de Developer Experience, la interfaz de Herokus está bien pulida, es intuitiva y fácil de usar. Muchas veces Heroku ha sido visto como un ejemplo por otros proveedores de PaaS, por su facilidad de uso, características y confiabilidad. Un ejemplo de ello es Deis, que utiliza un sistema de paquete de compilación inspirado en Heroku en sus implementaciones. Las unidades básicas de potencia informática en el ecosistema de Heroku son los Dynos. Un Dyno es un contenedor ligero y aislado que ejecuta una instancia de la aplicación [50].

- **Sales Force**. Sales Force es un tercer proveedor líder de servicios en la nube después de Microsoft Azure. Marc Benioff fundó la fuerza de ventas en marzo de 1999. La Sales Force es una de las principales nubes de software CRM (Customer Relationship Management) que sirve todos sus datos. Maneja más de 750 aplicaciones que son conocidas por su soporte en varias características, como generar nuevos clientes potenciales, adquirir nuevos clientes potenciales, aumentar las ventas. Fue diseñado para la gestión de los datos de la organización centrados en los detalles del cliente y las ventas. Ofrece características que personalizan sus estructuras de datos incorporadas y la GUI que se adapta a las necesidades específicas de una empresa. En los últimos días, también comenzó a ofrecer la conectividad IOT (internet de las cosas) a la plataforma CRM [51].

- **IBM Cloud**. International Business Machines, que también se llama IBM [4], ocupa el cuarto proveedor después de la fuerza de ventas en todo el mundo y ofrece una amplia gama de ofertas que contienen software, hardware y servicios. Servicios de negocios globales, servicios de tecnología global, servicios de redes, servicios de continuidad de negocios y servicios de outsourcing son los servicios incluidos. IBM Cloud entró en vigor para comprar empresas que ofrecen servicios de consultoría en nube y servicios de implementación. IBM es bien conocido por la gestión de relaciones con los clientes que contiene una colección de diferentes nubes como Sales Cloud, Health Cloud, entre otros [51].

- **Google Cloud Platform.** Google Cloud Platform proporciona servicios ofrecidos por Google, como la computación en la nube pública. Para construir y alojar aplicaciones, almacenar datos, sitios web y analizar datos en la infraestructura escalable de Google. Ofrece servicios de Iaas, PaaS y SaaS. Los desarrolladores de software, los administradores de la

nube y otros profesionales de TI pueden acceder a los servicios proporcionados por Google a través de Internet o cualquier conexión de red segura [51].

2.3 Vulnerabilidades

Una vulnerabilidad es una falla existente en un sistema debido a defectos de codificación, diseño o implementación que podría ser explotada por una fuente de amenaza [52][53][54]. Estas vulnerabilidades pueden ser activadas mediante inyección de código malintencionado. La mayoría de ataques aprovechan fallas en la implementación, validaciones de entrada, mecanismos de autenticación entre otros [54].

La gestión de vulnerabilidades es la práctica de identificar, clasificar, remediar y mitigar estos fallos de seguridad [55]. En el trabajo elaborado en [56] se describe el ciclo de vida de una vulnerabilidad en el cual se presentan todos los estados posibles, los cuales son los siguientes:

- Nacimiento. Inicia la creación del fallo, por lo general ocurre involuntariamente durante el desarrollo del sistema. En el caso de que el nacimiento se dé intencionalmente, el descubrimiento y nacimiento se dan al mismo tiempo. Se considera la existencia de una vulnerabilidad cuando la probabilidad de implementación es distinta a cero.
- Descubrimiento. Se dice que la vulnerabilidad ha sido descubierta solo cuando ha sido relevada públicamente. Además, el creador puede recibir notificaciones del fallo directamente.
- Corrección. Una vulnerabilidad es corregible cuando el desarrollador lanza una modificación de software o un parche de seguridad que corrige el defecto subyacente
- Publicidad. Sucede cuando una vulnerabilidad se hace pública de varias formas.
- Scripting. Se da cuando se divulgan códigos de explotación para la vulnerabilidad lo que permite que cualquier persona con poca o nula habilidad pueda ejecutar la misma acción para activar el fallo.
- Muerte. Ocurre cuando la cantidad de sistemas explotables por la vulnerabilidad es un numero insignificante.

2.3.1 Vulnerabilidades y exposiciones comunes

Las Vulnerabilidades y exposiciones comunes (CVE por sus siglas en inglés), es una lista de vulnerabilidades de seguridad informática conocidas. Cada registro contiene un identificador conocido como CVE-ID, utilizado para una identificación inequívoca de una vulnerabilidad, en el

cual se describe la vulnerabilidad, versiones de software implicadas, soluciones (en el caso de existir) y las referencias donde se han hecho pública la vulnerabilidad y su forma de explotación. En estas referencias también es posible encontrar enlaces directos a la base de datos del NIST donde se muestra más detalles sobre la vulnerabilidad [57] [58].

Al nombrar y estandarizar las vulnerabilidades conocidas públicamente, el diccionario CVE ha sido empleado a lo largo del tiempo para el intercambio de información de seguridad, ha facilitado el análisis de vulnerabilidades, además se utiliza como un índice importante para muchos otros recursos relacionados con la seguridad. Por ejemplo, se han desarrollado políticas de seguridad para hosts o redes basadas en las características de las lagunas de seguridad y los respectivos vectores de ataque descritos en el diccionario CVE [59].

2.3.2 Vulnerabilidades en SaaS

En la tecnología en la nube, se necesita un modelo de seguridad sólido porque las aplicaciones y los datos de diferentes inquilinos usarán los mismos recursos que pueden ser vulnerables a los ataques de seguridad [60]. Las aplicaciones en la nube heredan las mismas vulnerabilidades que las aplicaciones web y la tecnología tradicional. Sin embargo, las soluciones de seguridad tradicionales no son suficientes para el entorno de computación en la nube porque las vulnerabilidades en la aplicación web en la nube pueden ser mucho más devastadoras que las aplicaciones web tradicionales [13] .

En el trabajo realizado en [54] se realiza la clasificación de distintos tipos de vulnerabilidades, las cuales se muestran a continuación:

- **Vulnerabilidades de inyección de código.** Estas vulnerabilidades ocurren cuando una persona malintencionada manipula los valores de entrada de un usuario en una consulta lo que resulta en un flujo inseguro de información comprometiendo a la aplicación. Por lo tanto, la principal causa de este tipo de vulnerabilidad es la falta de validación de datos controlables por el usuario [54].
- **Vulnerabilidades de inyección SQL.** Este tipo de vulnerabilidades permiten a un atacante causar daños en la base de datos de una aplicación como la extracción y modificación de datos. La razón principal de esta vulnerabilidad es la validación incorrecta de la entrada del usuario, manipulación de cookies y la modificación de variables del lado del servidor [54].

- **Cross-site scripting (XSS por sus siglas en inglés).** Esta vulnerabilidad permite a una persona malintencionada la inyección de código para la ejecución de código malicioso mediante un navegador web del cliente. Cuando un usuario visita una página web explotada su navegador ejecuta el código malicioso. Algunas consecuencias son el secuestro de sesiones, fuga de datos y destrucción de contenido[54].
- **Otras vulnerabilidades de inyección de código.** La inyección XML, la inyección de comandos y la inyección Protocolo Ligero/Simplificado de Acceso a Directorios (LDAP siglas en inglés) son vulnerabilidades de inyección de código, las cuales reemplazan la entrada con formato incorrecto en lugar de las consultas XPath, los comandos del sistema operativo y las declaraciones LDAP, respectivamente, lo que resulta en comportamientos no esperados por parte de la aplicación [54].
- **Vulnerabilidades de la lógica empresarial.** Las vulnerabilidades de lógica empresarial (BLV siglas en inglés) son fallos que permiten a una persona malintencionada la manipulación de la lógica de negocios de una aplicación. Desafortunadamente, son fácilmente explotables y los ataques que explotan BLV son transacciones legítimas de aplicaciones utilizadas para llevar a cabo una operación no deseada que no es parte de la práctica comercial normal [54].
- **Vulnerabilidades de control de acceso (ACV).** Estas vulnerabilidades surgen cuando las listas de control de acceso (ACL por sus siglas en inglés) no se incorporan de forma correcta durante la implementación de la aplicación. Debido a esto, un atacante pude obtener acceso a recursos no autorizados. Los dos tipos de ataques que pueden surgir a raíz de esta vulnerabilidad son los ataques de derivación de autenticación y autorización [54].
- **Desvío del flujo de la aplicación.** Este tipo de vulnerabilidad permite a una persona malintencionada omitir el flujo de trabajo de una ampliación. Por ejemplo, el atacante puede recibir la confirmación de un pedido sin siquiera haber pagado por ello mediante una aplicación en línea vulnerable [54].
- **Vulnerabilidades de gestión de sesiones.** Las vulnerabilidades de administración de sesiones (SM por sus siglas en inglés) surgen a partir de una incorrecta administración de las variables de sesión. Lo que permite a un atacante realizar secuestros de sesión, falsificación de solicitudes entre sitio y secuestro de clic. Por un lado el secuestro de sesión va dirigido apuntan al ID de sesión del usuario mientras que, el secuestro de clics apuntan al navegador [54].
- **Desconocimiento de empleados y usuarios en la nube.** Debido al desconocimiento de los sistemas TI por parte de los empleados y usuarios de la nube un atacante puede ejecutar

ataques de ingeniería social como el ataque phishing. Si el ataque resulta exitoso el atacante puede robar las credenciales de los usuarios. Esta vulnerabilidad existe debido a la falta de estrategias de contratación y verificación de antecedentes, falta de selección de empleados y falta de aplicación de educación en seguridad informática [53].

- **Acceso fácil y no autorizado.** Sucede cuando no se implementa un método de encriptación seguro en las interfaces de gestión, las cuales son accesibles a través de internet. Debido a que los avances en criptografía provocan que un cifrado fuerte se convierta en débil [53].
- **Navegadores web y API insegura.** Es posible acceder al servicio de nube mediante un navegador y la API. Sin embargo, puede ocurrir una infección en el navegador web debido a la visita a sitios web maliciosos o APIs inseguras [53].
- **Vulnerabilidades relacionadas con el almacenamiento de datos.** En la nube es muy común que los datos de los usuarios sean almacenados en la misma ubicación. Esto se realiza con el fin de optimizar los recursos computacionales. Sin embargo, pueden surgir problemas de segregación y aislamiento de datos de los usuarios debido a diferentes jurisdicciones o lugares. Otros problemas suelen ser la eliminación de datos incompleta o insegura, la falta de transparencia y falsificación de metadatos [53].

2.4 Amenazas

Una amenaza es cualquier entidad (sistema, persona, objeto entro otros) con capacidad de actuar contra un activo provocando cualquier tipo de daño. Tanto un tornado, una inundación y un hacker son considerados amenazas. La principal consideración es que las amenazan aplican la fuerza (tornado, inundación, código de explotación entre otros) contra un activo provocando un evento de pérdida [61]. En el mundo computacional las amenazas pueden afectar la misión, funciones y reputación de una empresa mediante el acceso no autorizado, destrucción, divulgación, modificación de información o denegación [52]. Por lo tanto, una amenaza es conocida por un daño potencial a un sistema [53].

En el trabajo presentado en [62] se describen las seis categorías de amenazas que se encuentran en el modelo STRIDE presentada por Microsoft las cuales se describen a continuación:

- Falsificación, lo cual implica el robo de credenciales de otros usuarios para acceder a datos confidenciales.
- Manipulación, consiste en alterar maliciosamente los datos de otros usuarios.

- Repudio, capacidad de realizar acciones por parte de un usuario sin que otra entidad pueda comprobar dichas acciones.
- Divulgación de información, ocurre cuando se expone información confidencial de un usuario.
- Denegación de servicio, se da cuando se afecta a la disponibilidad del servicio e intenta saturar los recursos de la nube para que no respondan a las solicitudes de otros usuarios.
- Elevación de privilegios, consiste en que un usuario normal logra obtener permisos de administrador.

2.4.1 Amenazas en SaaS

En esta sección se presentan las amenazas más importantes para el modelo de servicio SaaS, las cuales se muestran a continuación:

1. **Control de pérdida sobre Recursos.** En la nube, los clientes proporcionan sus datos a un proveedor. Por lo que, puede ocurrir que el proveedor no informe como está administrando la información proporcionada por el cliente. Por lo tanto, los clientes deben administrar de forma confidencial la migración a la nube, además de ser necesarios para definir cláusulas especiales en los contratos con el proveedor [53].
2. **Uso indebido de recursos de computación en la nube.** Tener una interfaz de acceso simple hace que los usuarios o empleados malintencionados ataquen la infraestructura en la nube. Para reducir el riesgo de ataque, se debe implementar el cifrado y se deben realizar comprobaciones de antecedentes para los empleados. Además, es necesario utilizar controles de autenticación [53].
3. **Diferente modelo de prestación/recepción de servicios.** La nube puede cambiar la forma en que proporciona sus servicios. Los datos de usuario pueden cambiar los servidores, por lo que se pueden gobernar por diferentes leyes de seguridad debido a la ubicación. Como solución se propone el uso de cifrado punto a punto y estandarizar las leyes de seguridad [2].
4. **Interfaz insegura y API**. La nube concede una API para la comunicación con sus servicios. Por lo tanto, la seguridad en la nube depende en gran medida de las API. Si se ataca la API, puede afectar a la disponibilidad del servicio en la nube. Como medida de seguridad, se deben implementar mecanismos de seguridad sólidos y una interfaz segura [2].
5. **Infiltrados malintencionados.** En general, los empleados del proveedor de la nube tienen un mayor nivel de acceso, por lo que es probable que puedan acceder a información confidencial del cliente [53]. Están bien capacitados y versados en infraestructura,

herramientas y equipos para operar sus tareas. Sin embargo, pueden convertirse en adversarios cuando están insatisfechos con la toma de decisiones de la organización, sus afirmaciones no se cumplen, no son recompensados o la organización no los trata bien [63]. El proveedor debe tener herramientas que puedan rastrear a sus empleados con el fin de detectar actividad maliciosa [53]. Como medida de prevención se recomienda el uso de informes de acuerdos y notificaciones de incumplimiento, el proceso de seguridad y gestión debe ser transparente [2].

6. **Scavenging de datos.** Los datos de usuario se pueden hospedar en el mismo segmento de almacenamiento. Por otro lado, se pueden hacer varias copias de seguridad de la información alojada en diferentes ubicaciones. Desafortunadamente, esto dificulta las solicitudes de eliminación completa de datos, por lo que un atacante podría robar los datos de una organización. Como medida de seguridad se recomienda que el usuario mencione una confidencialidad de sus datos [53]. Los atacantes pueden recuperar los datos eliminados, porque la información todavía puede existir en el medio de almacenamiento a menos que se destruya [64].
7. **Pérdida o fuga de datos.** La pérdida de datos se produce cuando la información se transfiere o almacena incorrectamente [64]. La pérdida de datos es causada por diferentes factores como el cifrado débil, contraseñas simples y la falta de copias de seguridad [2]. La fuga de datos es un problema considerable, por lo que se requiere un control estricto[65]. Como medidas de seguridad, se propone utilizar contraseñas seguras y métodos de cifrado, copias de seguridad periódicas y el uso de API seguras [2].
8. **Secuestro de servicios/cuentas**. Se produce cuando un atacante logra robar las credenciales de acceso de un usuario. Cuando esto se logra, un atacante puede usar la contraseña del usuario para hacer nuevos ataques [2][53][66]. Este tipo de ataque fue clasificado como el tercer mayor riesgo en la nube según un informe hecho por la alianza de seguridad en la nube [67]. Como medidas de seguridad, se propone una autenticación sólida, el uso de políticas de seguridad y el uso del cifrado en los canales de comunicación [2].
9. **Perfilado de riesgos.** Por lo general, la nube concede mantenimiento de hardware y software a un tercero. Esto puede ser beneficioso; sin embargo, la nube puede ignorar los procedimientos, lo que conduce a mayores riesgos y amenazas. Como medida de seguridad debe tener conocimiento de los registros, aspecto de los datos y la infraestructura, para garantizar la supervisión del uso de datos y alterar el sistema. Para reducir esta amenaza, la nube debe tener en cuenta los detalles de la infraestructura, los datos y los registros. Además, la nube debe tener un sistema de supervisión [2].

10. **Robo de identidad.** Se produce cuando un atacante pretende ser otro usuario utilizando sus privilegios, créditos y otros recursos, haciendo que la víctima pierda la confianza. Esto puede suceder debido a diferentes factores como keyloggers, phishing y métodos de autenticación débiles. Como medida de seguridad, se propone el uso de métodos de autenticación robusta y recuperación de contraseñas seguras. [2].

2.5 Riesgos

El riesgo es la posibilidad de que una amenaza afecte negativamente en las operaciones de una empresa como la misión, reputación, funciones entre otros mediante la explotación de una vulnerabilidad [17] [52]. En seguridad informática se dice que hay un riesgo cuando existe la probabilidad que surja un ataque derivado de una amenaza provocada por una vulnerabilidad. El riesgo de un sistema de información puede determinarse por la fórmula : Riesgo = Amenaza * Vulnerabilidad * Impacto [17]. Los riesgos más frecuentes en la nube son: abuso y uso nefasto, interfaces inseguras, empleados malintencionados, robo de tecnología, perdida de información, secuestro de cuentas y perfiles de riesgo desconocidos [68]. Determinar el valor de los riesgos permite a los analistas priorizar eventos y asignar los recursos para salvaguardas con respecto a su criticidad. [69].

Basado en los riesgos empresariales externos e internos y el proceso de promoción del sistema de información, el trabajo presentado en [70] el autor realiza un análisis de los riesgos del sistema de información a partir de ocho aspectos: riesgos de infraestructura, riesgos de proyectos, riesgos de aplicaciones, riesgos de activos de información, riesgos de continuidad comercial , riesgos estratégicos del sistema de información, proveedor de servicios de información y riesgos de subcontratación, y riesgos del sistema de información de la empresa externa [70] los cuales de detallan a continuación:

- **Riesgos de infraestructura.** La infraestructura es una designación general de una serie de computadoras concentradas o dispersas y recursos de equipos de red. El principal problema de los riesgos de infraestructura es que la confiabilidad de la infraestructura no es alta. Primero, una vez que alguno de los componentes de la infraestructura falla, además de reemplazar el hardware requerido, el impacto del tiempo de servicio en el sistema también se considerará, al mismo tiempo, debido al problema de compatibilidad del sistema, es necesario reemplazar el sistema como un todo. En segundo lugar, debido al reemplazo acelerado de la

infraestructura, el reemplazo seguro y rápido de los componentes originales también se enfrenta a una gran dificultad [70].

- **Riesgos del proyecto.** El proyecto del sistema de información es una serie de actividades que se pueden observar directamente, como el tiempo de inicio, tiempo de finalización entre otros factores. Por lo tanto, el proyecto del sistema de información tiene un estado especial en todos los riesgos del sistema de información [70].
- **Riesgos de aplicación.** El proyecto se completa de acuerdo con el plan de tiempo y el presupuesto designado. Se proporcionan todos los entregables, el siguiente trabajo es la aplicación del sistema de información [70].
- **Riesgos de activos de información**. El desarrollo y la utilización de activos de información es un trabajo estratégico y desafiante, que generalmente incluye la base de datos, el sistema de gestión de la cadena de suministro y el sistema de gestión de la relación con el cliente de la empresa de desarrollo. El desarrollo de activos de información puede aportar buenos valores económicos a la empresa, pero también va acompañado de los riesgos correspondientes [70].
- **Riesgos de continuidad del negocio.** El proceso comercial central e importante de la empresa moderna está impulsado por el servicio del sistema de información. Sin embargo, el sistema de información crea una forma exitosa para los negocios, y también puede convertirse en una fuente de fracaso empresarial [70].
- **Proveedores de servicios de información y riesgos de outsourcing de TI.** En los últimos años, el negocio de outsourcing ha logrado un gran desarrollo. Los proveedores de servicios desempeñan un papel importante en la entrega de proyectos y el funcionamiento normal del sistema de información. Ahora, sin la participación de dichos terceros, será muy difícil proporcionar un servicio de tecnología de la información. Incluso en algunas áreas donde los servicios de tecnología de la información no son muy populares, el suministro y el soporte de productos de tecnología de la información también deben depender de terceros [70].
- **Riesgos del sistema externo de información empresarial.** Con el comercio colaborativo y la integración integral de la cadena de suministro, varios riesgos del sistema de información de la empresa externa afectarán el sistema de información de la empresa. Una vez que surjan los riesgos, tendrán graves consecuencias sobre los activos de información y la aplicación del del sistema de información empresarial [70].
- **Riesgos estratégicos del sistema de información.** La estrategia del sistema de información debe adaptarse a la estrategia comercial general de la empresa, en particular, la estrategia comercial de algunas empresas está impulsada por el sistema de información. Una vez que el

sistema de información de estas empresas no pueda apoyar la promoción de la estrategia comercial, la estrategia comercial será un fracaso total [70].

2.6 Ataques

2.6.1 Ataques y sus contramedidas en SaaS

Al finalizar el desarrollo de la revisión sistemática presentada en la sección 3.1 se obtuvo una lista de los ataques más comunes en aplicaciones que corren bajo el modelo de SaaS, los cuales se describen a continuación:

ARP Spoofing. Este ataque es que una persona maliciosa envía un mensaje ARP alterado para redirigir un host malicioso. Dado que ARP no requiere comprobar el origen, el atacante puede planear un ataque ARP derivando conexiones a un host específico. Como medidas de seguridad, se deben utilizar técnicas de detección, cifrado y filtrado en tablas ARP [40][53].

Backdoor and debug options. Este tipo de ataque se produce cuando los desarrolladores dejan habilitada la depuración en sus aplicaciones. Con esto, el atacante puede realizar fácilmente cambios en la aplicación. Para contrarrestar este ataque, la opción de depuración, los exámenes periódicos y la revisión de directorios deben estar deshabilitados [71] [72].

Broken authentication. Esto sucede debido a la implementación incorrecta de la administración de autenticación. Algunas amenazas son credenciales de usuario que se pueden descubrir debido a funciones de administración de cuentas débiles, credenciales sin cifrar, datos de sesión presentes en la URL, etc. Lo que hace que una persona maliciosa robe la identidad de un usuario legítimo, por lo general un atacante busca una cuenta con permisos limitados [73]. Los principales ataques de este tipo son ataques de fuerza bruta, ataques de phishing, ataque de secuestro de cuenta, ataques internos, ataque keylogger [74]. Para contrarrestar estos ataques existen métodos de prevención como la implementación de autenticación segura, el blindaje contra Cross Site Scripting evitando errores de script, control de acceso, uso de referencias indirectas de objetos ya sea por usuario o por sesión, implementación de verificación automática, mecanismos de autenticación multinivel y el uso de firmas digitales de huellas dactilares (puede resultar en un alto costo) [74].

Buffer overflow. Esto sucede cuando una aplicación intenta almacenar más datos en un búfer de los que admite. Los búferes deben contener una cantidad estática de datos, cuando se produce un desbordamiento de búfer, el contenido está dañado o se sobrescribe [16]. Debido a esto, un atacante

puede ejecutar código malicioso y obtener privilegios de administrador [75]. Una contramedida para este ataque es la aleatorización de instrucciones [40].

Code Injection. Debido a la naturaleza de la nube de manejo de entornos compartidos, los atacantes son capaces de insertar código malicioso en las aplicaciones para obtener datos confidenciales de los usuarios. Si el usuario hace clic en una URL infectada, puede ejecutar código en su máquina, lo que puede hacer que el atacante acceda a su información. Por otro lado, la inyección SQL es una de las técnicas más comunes de este tipo de ataques que permite a un atacante insertar comandos SQL desde fórmulas web para acceder a la base de datos. El filtrado de contenido activo se utiliza para detectar este tipo de ataque y proporcionar el uso de SQL generado dinámicamente en el código[16] [71][74] [75].

Cookie Poisoning. Este ataque se produce cuando un atacante logra obtener las cookies de una víctima con el fin de obtener acceso a una aplicación. Debido a que las cookies contienen información para iniciar sesión en aplicaciones o sitios web, el atacante puede falsificarlas para autenticarse como usuario autorizado. Como medida de precaución, las cookies almacenadas en el ordenador del usuario deben eliminarse periódicamente [71][72].

Cross-Site Request Forgery (CSRF). Este tipo de ataque es hacer que el navegador de un usuario envíe una solicitud HTTP falsa con datos confidenciales como cookies y credenciales de autenticación. Mientras que el usuario legítimo está conectado a una aplicación y visita un sitio malicioso, este sitio puede inyectar código en el navegador del cliente, lo que puede conducir al robo de identidad o de datos como tarjetas de crédito[66][75]. Como medidas de prevención, se recomienda el uso de tokens secretos y encabezados de referencia y origen [40].

DNS Poisoning. Los servidores DNS relacionan las direcciones IP con un nombre de dominio. Si esta relación de nombres está dañada, el atacante puede redirigir a un usuario a un sitio web malintencionado. Para evitar este ataque, se recomienda la aplicación de cifrado y filtrado [40][53] [71].

Dumpster Diving. Esta técnica consiste en el intento de recuperar información de datos eliminados de forma insegura. El usuario malintencionado logra restaurar los datos que los usuarios eliminan. De esta manera, un atacante puede centrarse en un usuario específico para obtener información relevante. Los datos eliminados pueden contener credenciales, cookies, números de tarjetas de crédito entre otros. Para reducir la posibilidad de sufrir este ataque, se recomienda la implementación de una política de seguridad para la eliminación de documentos físicos y digitales [76][77].

Eavesdropping. Este ataque se produce cuando un atacante escucha las transmisiones de red sin ser detectado causando un error de confidencialidad. Las transmisiones pueden ser mensajes, llamadas, videoconferencias entre otros[64][78]. Un método de seguridad es la implementación de IPsec [71].

EDoS. Este ataque se centra en la facturación de clientes. Consiste en inflar los costes de los servicios concedidos a los usuarios. Los ataques DoS a los servicios de pago por uso darán lugar a un aumento en el uso del ancho de banda, la CPU y el almacenamiento [71]. Como medida de seguridad, se debe implementar un cortafuegos para la detección de EDoS [79]. Además, se pueden realizar pruebas gráficas de EDoS Shield y Alosaimi Turing para evitar este tipo de ataques [80].

Google Hacking. Un atacante puede utilizar motores de búsqueda como Google porque es una buena opción para obtener datos confidenciales de un usuario u organización. Con este método pueden descubrir brechas de seguridad en las aplicaciones y llevar a cabo otros ataques [71] [72]. Las medidas de seguridad propuestas son evitar compartir información confidencial en sitios web y el uso de herramientas para el análisis de vulnerabilidades [71].

Hash Value Manipulation. Este tipo de ataque se produce cuando una persona malintencionada altera el valor hash de un mensaje para obtener acceso a un archivo. Si el valor hash modificado se hospeda en la base de datos, el servidor vincula el archivo con el valor hash. Por otro lado, si el valor hash modificado no se encuentra en la base de datos, el servidor solicita un archivo al usuario. Esta vulnerabilidad puede ocurrir en los casos en que el servidor utiliza OpenSSL con la clase Ncrypto. Como medida de seguridad, se requiere la implementación de protocolos de comunicación sólidos que utilicen pruebas de cifrado y probabilísticas [76] [81].

Hidden field manipulation. El ataque se produce cuando el desarrollador de la aplicación utiliza campos ocultos para el usuario. Por ejemplo, se puede utilizar para almacenar precios, pero un atacante puede aprovechar para hacer compras con precios alterados. Por lo tanto, es necesario evitar la colocación de campos ocultos para contrastar este tipo de ataques[71].

Malware injection and steganography attacks. Este ataque ocurre cuando es posible inyectar código malicioso en una aplicación. Con esto, un atacante puede insertar código en los archivos que pasan a través de la red. Debido a que parece que se está enviando un archivo normal, las herramientas de seguridad pueden ignorar este ataque[82]. En [39] se propone el uso de esquemas como StegAD para la detección de tales ataques.

Man-in-the-middle Attack. Sucede cuando una persona maliciosa intercepta la comunicación entre dos usuarios. El hacker puede simplemente escuchar el mensaje y reenviarlo o modificarlo. Gracias a esto, el atacante obtiene datos sensibles de los usuarios. Este ataque puede ocurrir debido a un fallo del protocolo de Internet, administración de contraseñas débiles, uso de redes inalámbricas inseguras o autenticación débil. Como medidas de seguridad se recomienda la implementación de canales seguros a través de SSL y aplicamos medidas de autenticación y autenticación a los nodos. Algunas herramientas útiles para prevenir estos ataques son Arijack, Cain, Dsniff y Ettercap [2] [53] [71] [74] [78] .

Meta Data Spoofing Attack. Esto ocurre cuando un atacante modifica la información sobre los servicios hospedados en el lenguaje de descripción de servicios web cuando se entrega. Debido a esto, el atacante obtiene acceso a aplicaciones y datos confidenciales. Como medida de seguridad, la funcionalidad del servicio debe estar cifrada y los mecanismos de autenticación robustos necesarios para acceder a este servicio[2] [53].

Phishing Attack. El phishing es una técnica que consiste en redirigir al usuario legítimo a un sitio web falso[67]. Como resultado, el usuario cree que es un sitio web confiable, así que ingresa sus credenciales comprometiendo la cuenta de usuario[2][67]. Para mejorar la seguridad en la nube, se debe implementar el cifrado, así como el uso de TLS para las aplicaciones. Por último, el uso de certificados a través de HTTPS es esencial [2][74].

Port scanning. Un atacante utiliza el análisis de puertos para averiguar si están abiertos, cerrados o filtrados. La persona malintencionada utiliza los puertos abiertos para obtener información de la red. Si un puerto está configurado para aceptar el tráfico sin ningún filtro, este puerto se verá afectado por un análisis de puerto. Como medida de seguridad, los puertos deben filtrarse y, cuando se detecta un análisis, bloquearlo. [2][71].

Race Condition. Este ataque se produce cuando varios procesos acceden a los mismos datos simultáneamente. Una persona malintencionada puede tener permisos de administrador mientras una aplicación está en modo de administrador[75]. En la investigación en [83], se propone una técnica llamada actualización de predicado para detectar este ataque.

Replay attack. Este tipo de ataque se produce cuando un atacante reproduce un mensaje obtenido a los destinatarios [78] con el fin de obtener acceso a datos no autorizados[71]. Para su detección sin afectar al rendimiento del sistema, en [84] se propone la aplicación del esquema de codificación estocástica

Reused IP address. Si un usuario cambia las redes y a otro se le asigna la misma dirección IP, conduce a un problema de seguridad. Esto se debe a que el usuario asume que sus recursos no son accesibles al salir de la red, pero si un nuevo usuario obtiene su dirección IP, puede tener acceso a estos recursos que infringe la privacidad del usuario original. Para evitar este problema de seguridad, se propone la eliminación de la memoria caché en las tablas ARP [71].

Service Injection Attack. Esto sucede cuando un atacante logra inyectar un servicio malicioso que hace que las solicitudes de los usuarios sean redirigidas a servicios malintencionados automáticamente. Como resultado, se pierde la integridad de los datos, el robo de cuentas o servicios. Como medida de seguridad, se propone el uso de un mecanismo de asilo sólido, mecanismos para identificar máquinas virtuales y utilizar servicios de integración [53].

Shared architectures. Dado que SaaS se ejecuta en una arquitectura compartida, es posible detectar la ruta de ejecución de la aplicación. Con esto el atacante puede obtener suficiente información para el robo de cuentas de usuario. Como contramedida, el código binario de la aplicación debe revisarse [82].

Sniffing. Esta es técnica para capturar paquetes que viajan a través de la red usando una herramienta de software [67]. Un atacante podría robar datos confidenciales como credenciales y tarjetas de crédito [85]. Como medida de precaución, se recomienda el uso de protocolos criptográficos como SSL y TLS, la implementación de IPsec y el cifrado de cada paquete IP [85].

Social Engineering. Estos ataques se producen cuando una persona maliciosa logra que usuarios legítimos revelen sus datos sensibles tales como credenciales, correos electrónicos, números de tarjetas de crédito entre otros a través del engaño. Esto se logra a través del uso de páginas web, contacto telefónico, correos electrónicos falsos, entre otros [74][75]. Los principales enfoques para reducir este tipo de ataques son implementar políticas de seguridad y capacitación de los usuarios [86].

Sybil attack. El atacante roba la identidad de un usuario para crear una relación con un usuario legítimo que puede llevar al atacante a elevar sus privilegios dentro del sistema [71]. Como solución, se recomienda la implementación del algoritmo de cifrado simétrico [87].

User to Root Attack El atacante obtiene los privilegios de un usuario administrador, esto se logra mediante el desbordamiento de datos en una aplicación. Para reducir el riesgo de este ataque, que puede tener serias implicaciones para la confidencialidad y la integridad, recomendamos usar una contraseña segura y un mejor mecanismo de autenticación [2]. Además, es necesario adoptar un mecanismo de separación de privilegios para gestionar el control de acceso entre diferentes plataformas [75].

XML signature wrapping attack. Este ataque se produce debido a una vulnerabilidad que existe en los mensajes SOAP. Cuando el usuario envía una solicitud a través del explorador, el servidor genera un mensaje SOAP. El mensaje puede encontrar información utilizada para establecer la comunicación entre el cliente y el servidor. Si el atacante logra poner en peligro el mensaje, puede autenticarse como un usuario legítimo. Como medida de prevención para este ataque, se recomienda el uso de herramientas para el análisis de vulnerabilidades y la verificación manual. [74].

Zombie Attack. También conocido como un ataque DoS distribuido, este tipo de ataque es más difícil de detectar que un ataque DoS. El atacante tiene varias máquinas infectadas llamadas zombies para ejecutar ataques de forma remota [2]. Como medidas de seguridad, se recomienda la implementación de sistemas IDS / IPS, controles de carga, limitación de paquetes ICMP y SYN, filtrado de direcciones

IP, análisis de detección detallados para la detección de intrusos y mejores controles de autenticación y autorización [53][71][74][85].

2.7 Estimación de riesgos

La estimación de riesgos forma parte de la estructura de la gestión de riesgos, la cual otorga la identificación de objetivos que pueden verse involucrados, realiza un análisis de riesgo, proporciona consecuencias y probabilidades de ocurrencia para poder tomar una decisión con el fin de descubrir si se requiere un tratamiento adicional [88].

El motivo principal para la elaboración de una estimación de riesgos es ofrecer información calificada sobre la probabilidad de que ocurra un evento o se repita durante un tiempo determinado [89]. Un análisis de riesgo representa el proceso de analizar un entorno objetivo y las relaciones de los atributos relacionados con el riesgo. El analista de riesgos debe identificar amenazas, fuentes de amenazas, vulnerabilidades, asociar vulnerabilidades con los activos afectados, identificar y evaluar el riesgo, reduciendo las contramedidas [90]. La estimación de riesgos responde preguntas como [88]:

- ¿Qué puede pasar y por qué?
- ¿Cuáles son las consecuencias?
- ¿Cuál es la probabilidad de que vuelva a suceder en el futuro?
- ¿Existen factores que mitiguen las consecuencias del riesgo o que reduzcan la probabilidad del riesgo?
- ¿Es el nivel de riesgo?

En la industria actual hay tantas herramientas de estimación de riesgos disponibles que requieren que los profesionales descubran posibles fallas de un producto. Aunque el análisis dirigido por expertos, aunque relevante, no es perfecto debido a muchos sesgos personales, restricciones de costo y tiempo. Muchas organizaciones usan el software sin ninguna evaluación o consideración. Por lo tanto, es muy esencial determinar el impacto del software o evaluar el software de una organización cuando se utiliza con fines institucionales. En este proceso, el software evaluado se verifica si cumple con todos los requisitos del usuario o no [91].

Estimación de riesgos (ER), que facilita la estimación y el cálculo del riesgo que enfrenta la organización. El riesgo se representa principalmente como una función del grado de daño y la posibilidad de que ocurra un daño. La gestión de riesgos tiene como objetivo identificar, controlar y mitigar los riesgos para los sistemas de información. Por lo tanto, la evaluación de riesgos es una

piedra angular de la gestión de riesgos, que incluye pasos que pueden agruparse en las siguientes cuatro fases, a saber: (i) determinación del riesgo enfrentado, (ii) evaluación del riesgo, (iii) acciones receptivas para mitigar el riesgo y (iv) monitoreo del riesgo [92].

A continuación se presentan varios métodos propuestos por diversos autores para elaborar estimaciones de riesgo [92]:

- En el trabajo[93] se propuso el método CERTS para evaluar de manera efectiva y objetiva las herramientas para administrar el riesgo al que está expuesto un sistema de información y para crear criterios de comparación para las herramientas. Los CERTS consisten en siete criterios: consistencia, usabilidad, adaptabilidad, factibilidad, integridad, validez, credibilidad. Cada criterio incluye de dos a cuatro atributos que describen y definen el criterio específico.
- En el trabajo [94] se propusieron 17 criterios para la selección de un método de análisis de riesgo apropiado. Este trabajo se centra principalmente en los sistemas en desarrollo en lugar de los existentes y se basa en CERTS propuesto por [93]. Utiliza cinco de siete criterios CERTS. Los criterios propuestos se agruparon en (a) características del método (incluyendo costo, acuerdo de analistas y administración, flexibilidad, complejidad, integridad, consistencia, facilidad de uso, utilidad, validez, confiabilidad y soporte de software) y (b) características de la organización (incluyendo nivel de riesgo, tamaño, conciencia de seguridad, requisitos externos y estructura organizativa). Los trabajos [94] [93] son la base para varios marcos de criterios propuestos.
- En otro enfoque, el trabajo presentado en [95] compara los pasos de los métodos de ER. Las dimensiones estructurales del marco incluyen criterios de alcance y evaluación que apoyan su profundidad y amplitud de contexto. Las dimensiones de procedimiento del marco incluyen: "proceso" y "herramientas de evaluación" que se utilizan para mejorar su funcionalidad.
- En el trabajo [96] se presenta un marco de comparación centrado en MAGERIT, MEHARI, Instituto Nacional de Estándares y Tecnología (NIST por sus siglas en inglés) y la Guía de seguridad de Microsoft. El marco se basa en los pasos de cada método y su documentación. Este trabajo sugiere que los métodos incluyen tres pasos generales de análisis de riesgos, a saber, “identificación de amenazas”, “identificación de vulnerabilidades” y “determinación de riesgos”. Todos los métodos recomiendan contramedidas como parte del proceso de gestión de riesgos.

- En la investigación presentada en [97] se creó un marco para la comparación y el análisis de métodos de evaluación de riesgos. El marco se basa en las fases del análisis, como la verificación, la medición, la evaluación y sus resultados, y en los indicadores clave de rendimiento relacionados con la integridad y eficacia de cada uno. El objetivo del marco era proporcionar una manera fácil para que las organizaciones comparen y seleccionen el método apropiado. Sin embargo, como el autor también admite, el marco aún no se ha utilizado en la práctica.
- En el trabajo presentado en [98] se utilizó el proceso de jerarquía analítica (AHP por sus siglas en inglés) para desarrollar un marco de criterios para la evaluación de métodos y herramientas de análisis de riesgos. Uno de los criterios propuestos es el soporte del método o proceso. El soporte puede ser metódico (métrica, objetividad, precisión, flexibilidad, integridad) o software (interfaz de usuario, equipo apropiado, entre otros).
- En el trabajo presentado en [99] se propuso un marco para la comparación de métodos basado en seis criterios que van del 0 al 3, a saber: recursos necesarios, recopilación de datos para activos, amenazas y vulnerabilidades, costo, tiempo, precisión y simplicidad.
- En el trabajo presentado en [100] se utilizó el Proceso de Jerarquía Analítica (AHP) para proponer un modelo que permita la comparación transparente y objetiva de los diferentes métodos de AR. El modelo tiene como objetivo ayudar a la selección del método que sea más apropiado para las necesidades de una organización. Incluye cinco criterios clave (a saber, el alcance del método, la facilidad de uso, la madurez del método y el público objetivo) analizados en 17 subcriterios. El análisis de los 17 criterios es particularmente extenso y ayuda a crear un marco de evaluación integrado e identificar las características de cada método.
- En el trabajo desarrollado en [101] se propuso cinco criterios de comparación para los métodos de AR, a saber: complejidad, enfoque metodológico, herramienta de soporte, cobertura geográfica y el origen. El objetivo era proporcionar asistencia en la selección de métodos a través de un proceso de ejecución hipotecaria basado en una serie de criterios simples.
- El autor [102] propuso siete criterios de comparación, a saber: cuantificación, integración de características de seguridad, integración de amenazas y vulnerabilidades, perspectiva de fase de requisitos, nivel de precisión / validación, cumplimiento de estándares y herramientas de soporte. Se utiliza una tabla de comparación con valores binarios (SÍ / NO) para cada criterio.

- En el trabajo presentado en [103] se proponer un marco compuesto por 10 criterios de comparación para los métodos de AR. Algunos de los criterios y el enfoque seguido se parecen al trabajo de [101], pero se agregaron más detalles y criterios
- En el trabajo presentado en [104] se lleva a cabo una revisión de la literatura de ER de 2004 a 2014. Clasifican los artículos académicos de acuerdo con siete categorías relacionadas con la evaluación de riesgos, a saber: (a) que identifican el riesgo, (b) comparan el análisis de riesgo, (c) mejoran el análisis de riesgo, (d) comparar marcos, (e) mejorar marcos, (f) proporcionar estudios de casos y (g) realizar una evaluación de riesgos mediante la comparación de los resultados del análisis de riesgos. Finalmente, propuso una taxonomía de evaluación de riesgos, que se centra únicamente en el análisis de riesgos, utilizando los siguientes criterios: evaluación, perspectiva, valoración de recursos y medición de riesgos.
- En la investigación desarrollada en [105] el autor compara 11 métodos de evaluación de riesgos de acuerdo con las tareas identificadas, la aplicación y los resultados del análisis utilizando el Marco de riesgo unificado básico (CURF).Su perspectiva es que los marcos previos para la comparación de métodos son restrictivos ya que los parámetros predeterminados limitan el análisis cuando los elementos de los métodos analizados no encajan en sus definiciones. Usando su método de abajo hacia arriba que primero identifica las tareas de los métodos y los define, el autor identifica numerosos elementos de identificación, estimación y evaluación de riesgos, clasificándolos según los puntajes de finalización (no abordados, parcialmente abordados, totalmente abordados). Después de aplicar el método a los estudios de caso, el autor puede ilustrar las ventajas y desventajas de un conjunto de tres métodos que demuestran su nivel de integridad.

2.8 Marco técnico de seguridad

Un marco de seguridad otorga una guía a los propietario y operadores de los sistemas informáticos. Esta guía identifica e implementa una estrategia de gestión de riesgos de seguridad para proteger los sistemas de la información. Para esto, el marco proporciona información sobre las vulnerabilidades, amenazas e impactos potenciales de un ataque informático. Proteger los sistemas de amenazas emergentes es un proceso. Este proceso requiere incorporar las mejores políticas, prácticas y procedimientos de seguridad. El marco proporciona estudios de casos que destacan las amenazas de seguridad informática comunes, las vulnerabilidades y las recomendaciones de mitigación para reducir estas vulnerabilidades, garantizando la confiabilidad del servicio [106].Por lo tanto, un marco técnico de seguridad reduce las amenazas de seguridad de los datos en el entorno de nube al otorgar

transparencia tanto para el proveedor como el usuario de la nube [107] y proporciona un conjunto de mecanismos de seguridad que ayudan a los desarrolladores a proteger sus aplicaciones [108].

2.8.1 Ejemplos de marcos de seguridad

- Un marco de seguridad para entornos de computación en la nube seguros [109]. En ese trabajo los autores proponen un enfoque que utiliza el análisis cuantitativo para asegurar que los resultados sean más lógicos y eficientes. Se presenta un marco que identifica los desafíos de seguridad y privacidad en la computación en la nube. Destaca los riesgos y los ataques específicos de la nube e ilustra claramente sus mitigaciones y contramedidas. Esto ayuda a administrar el sistema en la nube de manera más efectiva y proporciona a los analistas de seguridad la solución específica para contrarrestar la amenaza.
- Un marco de seguridad mejorado para electrodomésticos en hogares inteligentes [110]. En este trabajo se propone un marco de seguridad mejorado para dispositivos inteligentes en un entorno de hogar inteligente. El marco de seguridad proporciona el sistema de integridad utilizando las técnicas de autofirmación y control de acceso para prevenir las amenazas de seguridad como la modificación de datos, la filtración y la fabricación de códigos.
- Un marco conjunto de seguridad de datos médicos y consciente de los recursos para los sistemas de salud portátiles [111]. En esta investigación los autores desarrollaron un marco de seguridad basado en biometría para sistemas de monitoreo de salud portátiles con recursos limitados extrayendo los latidos del corazón de las señales de ECG. Se analiza que las características biométricas basadas en el dominio del tiempo juegan un papel importante en la optimización de la seguridad en las aplicaciones médicas basadas en IoMT.

2.9 SALSA Framework

Esta es una metodología para llevar a cabo una revisión sistemática de un tema específico [112]. A continuación se presentan sus fases [112]:

- **Protocolo.** En esta fase se define el alcance de esta investigación mediante el planteamiento de preguntas
- **Buscar.** En este paso, se seleccionan las bases de datos y se crea una cadena para llevar a cabo la búsqueda en distintas bases de datos con el fin de obtener documentos relevantes para la investigación.

- **Valoración.** En este paso se lleva a cabo una evaluación para seleccionar los documentos que serán útiles para la investigación. Para ello, se realizó una selección de estudios y una evaluación de la calidad.
- **Síntesis.** En esta fase, se lleva a cabo una clasificación de los documentos para crear una base de información para su análisis. Este método se divide en dos fases: extracción de información y categorización. Durante la fase de extracción de datos, implica la obtención de datos importantes en los documentos seleccionados en el paso anterior de SALSA FRAMEWORK. Por otro lado, la categorización implica la clasificación y el procesamiento de datos para su análisis.
- **Análisis.** Durante la fase de análisis, los datos obtenidos de la síntesis se utilizaron para proporcionar elementos suficientes para responder a las preguntas de investigación. El objetivo es mapear la relación entre los temas de cada documento.
- **Informe.** Por último, el informe se realiza en forma de documento en el que se muestran los resultados obtenidos de la revisión sistemática.

2.10 Problemas de seguridad en la nube

Los ataques informáticos más comunes que pueden ocurrir en la nube son los siguientes:

- *Denegación de servicio (DoS)*: Sucede cuando se afecta la disponibilidad de una aplicación o servicio. Un ataque de este tipo suele ser más devastador en un ambiente de nube debido a la naturaleza compartida de esta tecnología [113].
- *Ataques de autentificación*: Los servicios en la nube han provocado que ataques como ataque de hombre en medio o de fuerza bruta sean más propensos que nunca. Por lo que es necesaria la implementación de mecanismos fuertes de autenticación y administración de credenciales. [114].
- *Aislamiento de máquinas virtuales:* Las máquinas virtuales de la nube se encuentran lógicamente aisladas. Sin embargo, puede ocurrir fuga de datos y ataques a través de máquinas virtuales vecinas mediante la explotación de vulnerabilidades [67].
- *Ataques de canal lateral*: Un atacante puede alquilar ciertas máquinas virtuales con el fin de lanzar ataques de canal lateral, los cuales se realizan mediante el monitoreo de los circuitos síncronos del servidor. Si un ataque de este tipo resulta exitoso la persona malintencionada puede robar datos sensibles de los usuarios de la nube. Como contramedida se sugiera la aplicación lógica insensible a la demora dual-spacer (D3L por sus siglas en inglés) [115].

- *Ataques internos*: Son efectuados por los empleados de la nube al estar debidamente entrenados en el uso de los sistemas y mecanismos de seguridad pueden. Para evitar este problema se recomienda el control y administración de los accesos por parte de los empleados [116].
- *Ataques de memoria compartida (SMA):* Debido al concepto multicliente de la nube, los usuarios comparten la memoria física lo que puede provocar que un atacante logre obtener información útil [117].
- *Elevación de privilegios*: El atacante adquiere permisos de administrador dentro del sistema lo que le permite obtener acceso a las máquinas virtuales o al host [79].
- *Escaneo de puertos.* El escaneo de puertos proporciona una lista de puertos abiertos, puertos cerrados y puertos filtrados lo que permite a una persona malintencionada planear el ataque a los servicios que se ejecutan en estos puertos [79].
- *Ataques de canal de puerta trasera.* P a una persona malintencionada obtener acceso remoto al nodo infectado para comprometer la confidencialidad del usuario [79].

2.11 Herramientas para detección de vulnerabilidades

En esta sección se presentan diversas herramientas de seguridad para la detección de vulnerabilidades en redes y códigos fuentes.

2.11.1 Nmap

Nmap (Network Mapper) es una herramienta gratuita y de código abierto para el descubrimiento de redes y la auditoría de seguridad. Muchos administradores de sistemas y redes también lo encuentran útil para tareas como el inventario de red, la administración de los horarios de actualización de servicios y la supervisión del tiempo de actividad del host o del servicio. Nmap utiliza paquetes IP sin procesar de formas novedosas para determinar que host está disponible en la red, que servicios ofrece ese host, que sistemas operativos están ejecutando, que tipo de filtros de paquetes / firewall están en uso entre otras características. Fue diseñado para escanear rápidamente redes grandes, pero funciona bien contra un solo host. Nmap se ejecuta en todos los principales sistemas operativos de la computadora, y los paquetes binarios oficiales están disponibles para Linux, Windows y Mac Os X [118].

2.11.2 OWASP's Dependency Check

OWASP Dependency Check es un escáner gratuito elaborado por por el Proyecto de seguridad de aplicaciones web abiertas (OWASP por sus siglas en inglés) que cataloga todos los componentes de

código abierto utilizados en una aplicación y resalta las vulnerabilidades en estas dependencias. Funciona para Java, .NET, Ruby, PHP, Node.js y Python, así como para algunos proyectos C / C ++. Dependency Check se integra con herramientas de compilación comunes, incluidos Ant, Maven y Gradle, y servidores CI como Jenkis. Esta herramienta informa sobre cualquier componente con vulnerabilidades conocidas reportadas en la Base de Datos Nacional de Vulnerabilidad del NIST y recibe actualizaciones de los datos de NVD [119].

2.11.3 Flan Scan

Flan Scan es un escáner de vulnerabilidades de red ligero. Con Flan Scan es posible el hallazgo de puertos abiertos en una red, identificar servicios y su versión, y obtener una lista de vulnerabilidades CVE relevantes. Flan Scan convierte a Nmap, una herramienta de seguridad, en un escáner de vulnerabilidades de red completo. Además, envía los resultados a la nube e implementa el escáner en Kubernetes [120]. Las principales características de Flan Scan son las siguientes [120]:

- Fácil implementación y configuración. Flan Scan se ejecuta dentro de un contenedor Docker. Como resultado, Flan Scan se puede construir y enviar a un registro Docker y mantiene la flexibilidad para configurarse en tiempo de ejecución.
- Resultados en la nube. Flan Scan agrega soporte para enviar resultados a un Google Cloud Storage Bucket o un S3 bucket.
- Informes procesables. Flan Scan genera informes procesables a partir de la salida de Nmap para que sea posible identificar rápidamente los servicios vulnerables en su red, las CVE aplicables y las direcciones IP y los puertos donde se encontraron estos servicios.

2.12 Medidas de seguridad

Esta sección presenta medidas de seguridad adicionales, esto será útil para mejorar la seguridad de las aplicaciones alojadas en un entorno SaaS. Se describen los métodos para garantizar entornos *Multi-arrendatario*, el uso de un antivirus y el uso de un IDS. Por último, se muestran las medidas de control para la autenticación y autorización de los usuarios.

1. **Contramedidas por parte del proveedor**. Por motivos de seguridad, el proveedor de nube debe proporcionar al usuario credenciales cifradas y garantizar la integridad y autorización de los servicios. Además, llevar a cabo evaluaciones de riesgos constantes, capacitación de empleados en seguridad informática, cumplir con el privilegio mínimo en los usuarios, implementación de políticas de seguridad en la administración de credenciales, monitorear

los movimientos en línea de los empleados, implementar defensas contra código malicioso, implementar defensa en capas contra ataques remotos, monitorear comportamientos sospechosos y actuar según sea necesario, desactivar el acceso una vez que la sesión haya terminado, mantener registros para facilitar la investigación, implementar mecanismos de copia de seguridad y recuperación y documentar amenazas internas [121]. Para mitigar las vulnerabilidades que existen en la nube, tanto el proveedor puede implementar las siguientes contramedidas: cifrado de extremo a extremo, análisis de actividad malintencionado, implementación de API seguras, planes de continuidad del negocio, evaluación de empleados y contratistas para evitar ataques internos y validar el consumo de nube para evitar atacar EDoS [47].

2. **Asegurar el entorno *Multi-arrendatario* SaaS.** Para mejorar un *entorno Multi-arrendatario* en SaaS es posible implementar tres medidas de seguridad. En primer lugar, la segmentación basada en base de datos para que solo determinadas columnas sean accesibles para cada inquilino. Cifrado, el cifrado de la información suele ser útil en caso de que se vea comprometida o robada por un usuario malintencionado de esta manera no podrá leer la información. Por último, el inquilino debe conocer las protecciones existentes para garantizar el aislamiento de datos entre los inquilinos de la nube [122].
3. **Autenticación segura basada en la nube (CSA por sus siglas en inglés).** Este protocolo es un conjunto de tres protocolos. El primero se utiliza para el registro. El segundo es un protocolo de identificación basado en la adaptación que es muy útil para contrarrestar los ataques DoS. Finalmente, el tercer protocolo se utiliza para la autenticación. Las ventajas de este protocolo es que la nube puede confirmar la identidad del cliente para la autenticación segura y también, puede detectar y prevenir ataques DoS [123].
4. **Antivirus e IDS.** La implementación de un antivirus en la nube puede mejorar la seguridad porque supervisa y bloquea cualquier código malicioso que afecte al sistema en la nube. El antivirus puede analizar los archivos que se transfieren en la nube de esta manera es posible detectar amenazas y detenerlas [82]. Por otro lado, la implementación de un sistema de detección de intrusiones en la nube aumenta el nivel de seguridad porque detecta amaneceres dentro de la red [124].
5. **Métodos para mitigar las amenazas de autenticación y control de acceso.** Algunas medidas importantes para mejorar la seguridad en la autenticación y el control de acceso son: aplicar políticas de inicio de sesión único, implementar la autenticación multifactor, la implementación de la autenticación biométrica, la implementación del cifrado RSA, la implementación de sistema de detección de intrusión y cortafuegos, implementación de

estándares abiertos para la autenticación de Exchange y datos de autorización entre dominios de seguridad que permiten a los usuarios compartir recursos con el uso de token en lugar de contraseñas [47].

Según [125] se proponen siete medidas de seguridad, las cuales son las siguientes:

- Acceso de usuarios privilegiados: Los proveedores de nube deben ser cuidadosos en la contratación de nuevos empleados y en la asignación de privilegios, además, deben contar con fuertes mecanismos de seguridad para controlar y monitorear sus actividades.
- Cumplimiento normativo: Los proveedores de nube deben contar con certificaciones de seguridad para ser una mejor opción para los clientes
- Localización de información: Cada país cuenta con leyes diferentes por lo que los clientes de la nube deben saber bajo que leyes se esta está rigiendo su información, así como su ubicación.
- Segregación de datos: Los usuarios de la nube deben estar enterados en como su información es segregada. Una buena opción es la encriptación de datos sensibles.
- Recuperación: Debido a que siempre es probable que ocurrir una falla el proveedor de la nube debe contar con una infraestructura de recuperación en caso de desastres para la restauración de la información.
- Soporte a investigación: En el caso de que exista alguna sospecha de un ataque, el proveedor de la nube debe estar dispuesto a cooperar en las investigaciones. Por lo tanto, los registros recolectados por el proveedor deben estar disponibles para los usuarios.
- Disponibilidad a largo plazo: Debido a que toda empresa pude caer en bancarrota, los usuarios de la nube deben tener la seguridad de que su información siempre estará disponible y, además, que sea fácilmente exportada a otros proveedores siempre que sea necesario.

CAPÍTULO 3. Marco de seguridad para aplicaciones en SaaS

Para la elaboración del marco de seguridad para aplicaciones en SaaS se optó por realizar una revisión sistemática y posteriormente una estimación de riesgos. En la sección 3.1 se describe detalladamente los pasos realizados para elaborar la revisión sistemática con el empleo de *SALSA framework*. La revisión sistemática fue elaborada con el fin de obtener información relevante y de calidad sobre problemas de seguridad y sus respectivas contramedidas. En la sección 3.3 se describe el desarrollo de la estimación de riesgos en base a la guía publicada por el NIST con el nombre de "Guía para realizar evaluaciones de riesgo" [126]. Esta revisión se realizó con la finalidad de establecer probabilidades de ocurrencia de incidentes y los niveles de amenaza de los ataques que comúnmente ocurren en SaaS. Por último, con la utilización de la información clasificada durante la revisión sistemática y la estimación de riesgos se diseñó el marco de seguridad en el cual los desarrolladores podrán realizar una evaluación de seguridad en sus aplicaciones bajo el modelo de servicio SaaS.

3.1 Revisión sistemática

Esta sección presenta el proceso que se siguió para llevar a cabo la revisión sistemática utilizando *SALSA Framework*, realizando los pasos descritos en la sección 2. Los resultados obtenidos por la revisión sistemática fueron presentados en forma de artículo, el cual fue publicado en la revista *Journal of Computer Security*[1]. En las siguientes secciones se describen los pasos realizados para la elaboración de la revisión sistemática, los cuales son los siguientes:

1. Protocolo
2. Búsqueda
3. Evaluación
4. Síntesis
5. Análisis
6. Resultados

[1] https://content.iospress.com/articles/journal-of-computer-security/jcs200002

3.1.1 Protocolo

El objetivo de investigación para la elaboración de la revisión sistemática se define en forma de preguntas, las cuales son los siguientes:

1. ¿Cuáles son las amenazas que afectan a SaaS?
2. ¿Cuáles son los ataques más comunes en SaaS?
3. ¿Se aplican las mismas amenazas a las aplicaciones en la nube y a las aplicaciones en sitio?
4. ¿Será posible clasificar las amenazas?
5. ¿Qué medidas de seguridad se pueden aplicar en SaaS?

3.1.2 Búsqueda

Se utilizaron bases de datos relacionadas con el área de investigación, la tabla 1 muestra las cadenas de búsqueda y las bases de datos, así como el número de documentos que cada resultado dio lugar.

Tabla 1: Resultados de búsqueda clasificados por base de datos

Cadena de búsqueda	Base de datos	Fecha	Resultados
(SaaS OR "Software as a Service") AND (Vulnerabilities OR Security OR “Security Issues" OR "Security Challenges" OR "Security measures"))	ScienceDirect	03/05/2019	4771
(SaaS OR "Software as a Service") AND (vulnerabilities OR Security OR “Security Issues" OR "Security Challenges" OR "Security Measures")	IEEE	03/05/2019	672
(SaaS OR "Software as a Service") AND (Vulnerabilities OR Security OR “Security Issues" OR "Security Challenges" OR "Security Measures"))	ACM	03/05/2019	91
(SaaS OR "Software as a Service") AND (Vulnerabilities OR Security OR “Security Issues" OR "Security Challenges" OR "Security Measures")	SCOPUS	03/05/2019	1043
(SaaS OR "software as a service") AND (vulnerabilities OR security OR “security issues" OR "security challenges" OR "security measures")	WOS	03/05/2019	146
		Total	6723

3.1.3 Evaluación

La tabla 2 muestra los criterios que se utilizaron para incluir y excluir documentos de los resultados obtenidos durante la fase de búsqueda. Por otra parte, la tabla 3 muestra los filtros aplicados a un total de 6,723 artículos. Al final del proceso, había 47 artículos para la revisión sistemática.

Tabla 2**:** Criterios de inclusión y exclusión

ID	Criterios
I1	Documentos sobre amenazas en SaaS
I2	Documentos relacionados con la seguridad en SaaS
I3	Documentos sobre ataques en SaaS
E1	Investigaciones que no están en inglés
E2	Investigaciones publicadas antes del 2013
E3	Documentos publicados
E4	Investigaciones que no concuerdan con los objetivos
E5	Investigaciones que no tratan sobre seguridad en SaaS
E6	Investigaciones en los cuales no se tiene acceso

Tabla 3 : Resultados de la selección del estudio

Fase	Criterios	Total de artículos	Incluidos	Excluidos
Resultados de la búsqueda	Search string	6723	6723	0
Documentos no escritos en inglés	E1	6723	6705	18
Documentos antiguos	E2	6705	4329	2376
Documentos duplicados	E3	4329	3651	678
Lectura del titulo	E1 a E6	3651	437	3214
Lectura de resumen	E1 a E6	437	97	340
Lectura de introducción conclusión	E1 a E6	97	59	38
Lectura completa	I1 a I3 y E1 a E8	59	47	11

3.1.4 Síntesis

Para llevar a cabo este paso en los documentos seleccionados, se identificaron tres temas principales para cubrir el objetivo de la investigación, que son amenazas, ataques y medidas de seguridad. En la tabla 4 se muestra un ejemplo de extracción de datos de los cuatro primeros documentos.

Tabla 4: Ejemplo de extracción de datos en diferentes documentos.

Tema	Trabajo 1	Trabajo 2	Trabajo 3	Trabajo 4
Amenazas	SysAdmin malicioso, Pérdida / manipulación de datos, Ataques Dos, Ataque EDoS	La implementación insegura del protocolo OAuth puede dar lugar a la posibilidad de un ataque DoS	N/A	Diferente prestación de servicios, interfaz y API inseguras, información privilegiada maliciosa, pérdida y fuga de datos, secuestro de cuentas, generación de perfiles de riesgo, identificación

Medidas de seguridad	Autenticación multifactor, Auditoría y registro, IDS/IPS, mitigación de DDos, Firewall	Protocolo de resistencia DoS basado en la nube	Cifrado, escaneo, validación, interfaces seguras, ataques de información privilegiada seguros, recursos apalancados, planes de continuidad del negocio	N/A
Ataques	N/A	N/A	N/A	Zombie attack, user root attacks, escaneo de puertos, middle attack, metadata spoofing attack, phishing attack

3.1.5 Análisis

En este paso, cada uno de los trabajos previamente seleccionados se analiza en función de lo que se capturó durante la fase de síntesis. En la figura 1 se muestra el número de trabajos que tratan un ataque en específico, siendo el zombie attack y el man-in-the-middle attack los más populares. El objetivo de la fase de análisis es conseguir responder a las preguntas de investigación planteadas en la sección 3.1.1 En primer lugar, se examinarán los documentos relacionados con la cuestión de las amenazas que se producen en SaaS presentados en la sección 2.4.1. Posteriormente, se analizan los documentos relacionados con los ataques que pueden ocurrir en SaaS en la sección 2.6.1. Por último, la sección 2.12 muestra las medidas de seguridad que se pueden aplicar para evitar amenazas y ataques.

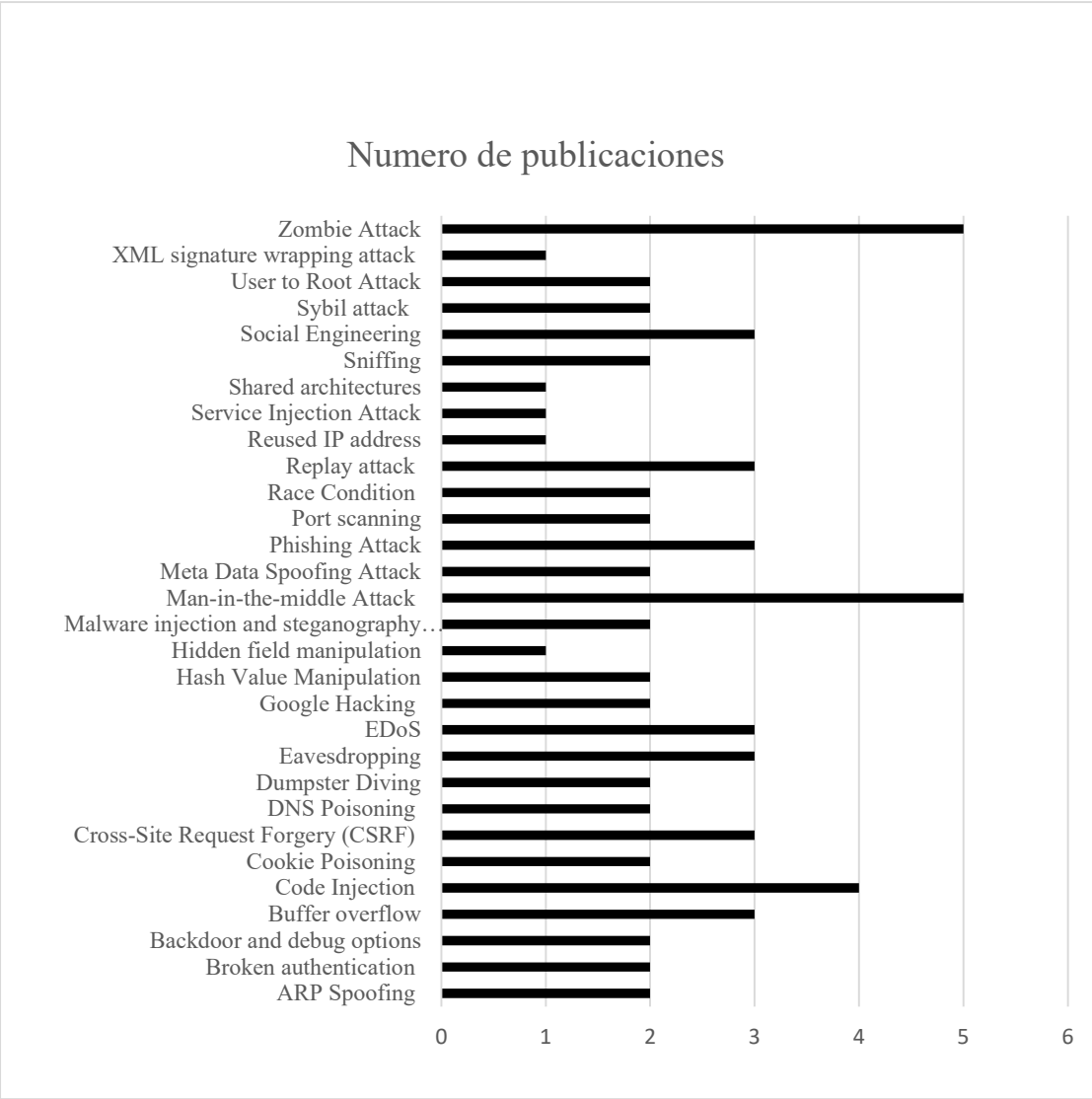


Figura 1: Número de publicaciones por ataque

La tabla 5 muestra un resumen de la lista de ataques y sus posibles contramedidas. A continuación, se muestra la lista de ataques y sus posibles contramedidas de acuerdo con la información recopilada de los diferentes documentos que se analizaron.

Tabla 5: Un resumen de la lista de ataques y sus posibles contramedidas

Ataque	Medidas de seguridad
ARP Spoofing	Filtrado de tabla ARP fiable, detección, cifrado.
Backdoor and debug options	El desarrollador debe deshabilitar la opción de depuración

Broken authentication	Controles sólidos de autenticación y administración de sesiones Aplicar automatización para verificar Evite errores de scripting entre sitios (XSS)
Buffer overflow	Aleatorización del conjunto de instrucciones
Code Injection	Filtrado de contenido activo Detección de vulnerabilidades de aplicaciones web
Cookie Poisoning	Limpieza regular de cookies
CSRF	Token secreto, encabezado de referencia y encabezado de origen
DNS Poisoning	Cifrado y filtrado
Dumpster Diving	Definir e implementar una política sobre la eliminación de documentos confidenciales Hacer un mayor uso de trituradoras de documentos
Eavesdropping	Implementar Internet Protocol Security (IP sec) Implementar políticas de seguridad y software antivirus
EdoS	Escudo anti EDoS y pruebas graficas de turing Alosaimi
Google Hacking	No compartir la información confidencial Uso de herramientas para escanear vulnerabilidades
Hash Value Manipulation	Se necesita un protocolo sólido para la posesión probable de datos
Hidden field manipulation	Evite colocar parámetros en campo oculto
Malware injection and steganography	Esquemas como StegAD
Man-in-the-middle Attack	Aplicación de las medidas de seguridad convencionales Herramientas como Dsniff, Cain, Ettercap, Wsniff, Airjack pueden ayudar
Meta Data Spoofing Attack	Mantenga la funcionalidad del servicio y otros detalles cifrados Autenticación fuerte
Phishing Attack	Algoritmos de cifrado y descifrado autoadaptables y autoajustables Implementar Transport Layer Security (TLS) Utilización de certificados y HTTPS
Port scanning	Bloqueo de puertos
Race Condition	Técnica de actualización de predicados
Replay attack	Esquema de codificación estocástica se propone en [84]
Reused IP address	Eliminación de caché
Service Injection Attack	Fuerte aislamiento Mecanismo de identificación sólido Implementación de la integridad del servicio
Shared architectures	Analizar el código binario de la aplicación
Sniffing	Implementación de SSL y TLS Implementación de Ipsec
Social Engineering	Centrarse en las políticas de seguridad y la formación del personal
Sybil attack	Una solución basada en criptografía de clave simétrica

User to Root Attack	Contraseña segura Mecanismo de autenticación fuerte
XML signature wrapping attack	Escáneres automáticos y verificación manual
Zombie Attack	Autenticación robusta Uso de IDS/IPS Un análisis periódico y exhaustivo del sistema Filtrar paquetes ICMP y SYN Filtrar direcciones IP privadas

3.1.6 Resultados

En esta sección se muestran los resultados recaudados con la elaboración de la estimación de riesgos. El objetivo de esta fase es responder las preguntas de investigación. Las preguntas que se plantearon anteriormente con sus respectivas respuestas son las siguientes:

¿Cuáles son las amenazas que afectan a SaaS?

La Sección 2.4.1 presenta en detalle las amenazas que pueden ocurrir en SaaS. En resumen, las amenazas pueden originarse a partir de software, un atacante y un empleado malintencionado. Los autores consideran que las amenazas que representan el mayor riesgo son el robo de identidad, el robo de cuentas y los empleados malintencionados porque pueden afectar a la confidencialidad, integridad y disponibilidad de la información.

¿Cuáles son los ataques más comunes en SaaS?

En la sección 2.6.1 se presentaron un total de 30 ataques que pueden considerarse como los más comunes que pueden ocurrir en SaaS. La figura 2 muestra los ataques más mencionados en los artículos investigados.

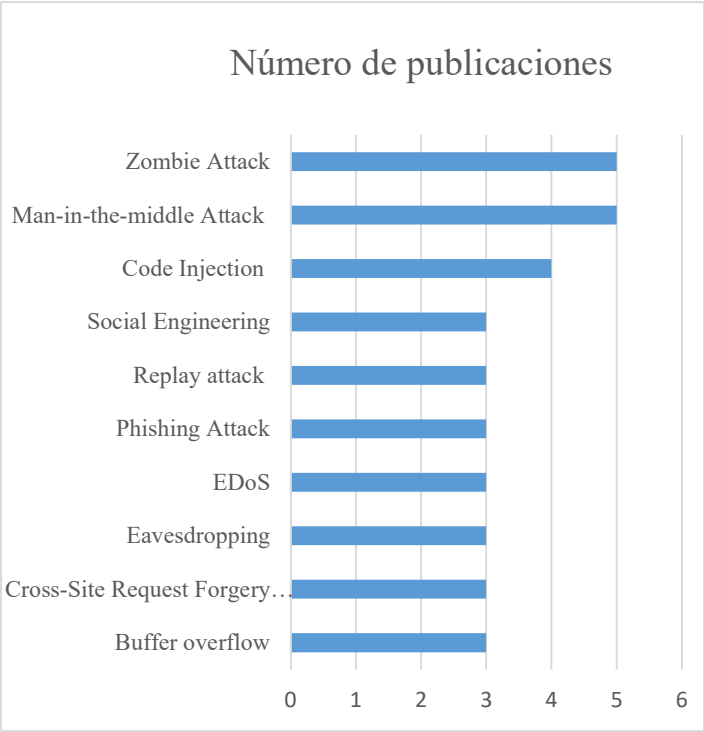


Figura 2: Numero de publicaciones por ataque

¿Se aplican las mismas amenazas a las aplicaciones en la nube y a las aplicaciones en sitio?

La Sección 2.3.2 menciona que el modelo de servicio SaaS hereda los problemas de seguridad de las aplicaciones tradicionales. A continuación, se muestra una serie de problemas de seguridad que se producen SaaS.

- **Problemas de seguridad**. Ataques como la denegación de servicios son más devastadores en entornos en la nube debido a la naturaleza del uso compartido de la infraestructura. Los firewalls tradicionales y el sistema de detección y prevención de intrusiones de red (IDP) no son eficaces para contrastar los ataques DDoS, XML-DoS y http-DoS. Las aplicaciones alojadas bajo el esquema SaaS requieren otro nivel de defensa en el nivel de aplicación para reducir la posibilidad de que estos ataques ocurran [14]. SaaS tiene impuestos específicos, por lo que para evitar riesgos, es necesario proteger los repositorios de información y datos almacenados [64]. Para mejorar la seguridad en la nube, la autenticación, la autorización y el control de acceso deben concederse a los usuarios [127].
- **Seguridad de datos.** Para garantizar la seguridad en una aplicación en la nube, es necesario tener confidencialidad, integridad y disponibilidad, lo que se conoce como CIA [16][17]. Algunos métodos para mejorar la seguridad de los datos que involucran a la CIA son los

siguientes: cifrado aplicado a los datos en reposo y tránsito, implementación de funciones hash, para validar la integración es posible utilizar el servicio de auditoría de terceros (TPA), no Almacenar en el mismo colocar las credenciales y las claves de cifrado, aplicación de autenticación robusta, para la disponibilidad se recomienda hacer copias de seguridad periódicas, redundancia y duplicación [73].

- **Software security.** La seguridad del software se utiliza para mejorar la seguridad de las aplicaciones. Esto para evitar vulnerabilidades tales como un desbordamiento de buffer [128]. La creación de una aplicación vulnerable puede causar su explotación por los usuarios malintencionados. Actualmente hay muchas amenazas de seguridad que afectan incluso a las aplicaciones en la nube, algunas de las cuales no son detectadas por las herramientas de seguridad. Por lo tanto, se requiere un buen control en el proceso de desarrollo [121]. La seguridad del software es el principal problema al que pueden enfrentarse los sistemas en la nube y los profesionales de aplicaciones. Los propietarios de datos pueden estar preocupados de que los datos y el software no están bajo su control, sino que son propiedad de la nube. Además, es posible que el titular de los datos no sepa dónde están los datos geográficamente en cualquier momento [45].
- **Seguridad *Multi-arrendatario*.** Una característica de SaaS es Multi-arrendatario [67]. El uso de multiinquilino permite a varios clientes conectarse a la misma lógica de la aplicación, asegurando que cada uno tenga su aplicación personalizada y no tenga acceso a los datos de otro inquilino [2]. Además, permite ahorrar recursos porque la eficiencia se mejora cuando se utiliza una infraestructura compartida [16]. El uso de esta tecnología aumenta el miedo al usuario del servicio porque sus datos pueden estar en la misma base de datos que su competidor o un usuario malintencionado [108]. Debido a que los datos de los diferentes inquilinos están en la misma infraestructura, puede suceder que si un inquilino es atacado, puede afectar a otros [2]. Además, el análisis forense es difícil [129].

¿Será posible clasificar los ataques?

Para garantizar la seguridad en una aplicación en la nube, es necesario tener confidencialidad, integridad y disponibilidad, lo que se conoce como CIA [17]. En este trabajo se utiliza la metodología STRIDE para clasificar el tipo de amenaza causada por los ataques, como se muestra en la tabla 6.

Tabla 6: Clasificación de los ataques en el modelo STRIDE y los objetivos de seguridad afectados.

Ataque	Tipo STRIDE	Objetivos de seguridad
ARP Spoofing	Falsificación	Autenticación
Broken authentication	Falsificación	Autenticación
Backdoor and debug options	Manipulación	Integridad
Buffer overflow	Elevación de privilegios	Autorización
Code Injection	Falsificación, Manipulación	Autenticación, Integridad
Cookie Poisoning	Falsificación	Autenticación
Cross-Site Request Forgery (CSRF)	Manipulación	Integridad
DNS Poisoning	Falsificación	Autenticación
Dumpster Diving	Divulgación de información	Confidencialidad
Eavesdropping	Divulgación de información	Confidencialidad
EdoS	Denegación de servicio	Disponibilidad
Google Hacking	Divulgación de información	Confidencialidad
Hash Value Manipulation	Manipulación	Integridad
Hidden field manipulation	Manipulación	Integridad
Malware injection and steganography attacks	Manipulación	Integridad
Man-in-the-middle Attack	Divulgación de información	Confidencialidad
Meta Data Spoofing Attack	Falsificación	Autenticación
Phishing Attack	Falsificación	Autenticación
Port scanning	Divulgación de información	Confidencialidad

Race Condition	Elevación de privilegios	Autorización
Replay attack	Manipulación	Integridad
Reused IP address	Divulgación de información	Confidencialidad
Service Injection Attack	Falsificación, Manipulación	Autenticación, Integridad
Shared architectures	Falsificación	Autenticación
Sniffing	Divulgación de información	Confidencialidad
Social Engineering	Divulgación de información	Autenticación
Sybil attack	Elevación de privilegios	Autorización
User to Root Attack	Elevación de privilegios	Autorización
XML signature wrapping attack	Elevación de privilegios	Autorización
Zombie Attack	Denegación de servicio	Disponibilidad

¿Qué medidas de seguridad se pueden aplicar en SaaS?

En la sección 2.6.1, se presentaron algunas formas de mitigación al final de cada ataque. Mientras que, en la Tabla 1 se muestra un resumen de todos los ataques y sus contramedidas. Finalmente, la sección 2.12 describe una serie de recomendaciones generales que deben aplicar tanto el proveedor como el cliente SaaS.

3.2 Estimación de riesgos

Para el desarrollo del marco de seguridad que se propone en este proyecto fue necesaria la elaboración de una estimación de riesgos con el fin de conocer el nivel de impacto y riesgo que un desarrollador corre al ser vulnerable a un ataque. Para esto, se utilizó como base la guía proporcionada por el NIST "*Guide for Conducting Risk Assessments"* [126]. Los pasos para la elaboración de la estimación de riesgos se muestran a continuación.

3.2.1 Preparación para la estimación de riesgos

La estimación de riesgos será desarrollada con el fin de resguardar la confidencialidad, integridad y disponibilidad de los datos en la nube. La cual contendrá la identificación de amenazas y ataques que pueden ocurrir en una aplicación en SaaS. Por lo tanto, los desarrolladores podrán decidir qué medidas de seguridad implementar en sus aplicaciones para desarrollar software en la nube más seguro.

3.2.2 Identificación del alcance

El alcance de la estimación de riesgos será para todas las aplicaciones desarrolladas para un entorno en nube bajo el modelo de SaaS. Estará dirigida a los desarrolladores de software nuevo e implementado. El tiempo de efectividad de esta estimación de riesgos dependerá del descubrimiento de nuevas fuentes de amenazas o ataques.

3.2.3 Identificación de supuestos y restricciones

Durante esta fase se identifican las fuentes de amenazas, eventos de amenaza, y vulnerabilidades con el fin de poder desarrollar una estimación de riesgos de calidad. Las fuentes de amenazas se describen en la sección 2.4.1. Mientras que, los eventos de amenaza (ataques informáticos) se muestran en la sección 2.6.1. Por ultimo las vulnerabilidades se presentan en la sección 2.3.2

3.2.4 Definiciones

Mediante esta fase se presentan las definiciones de palabras clave que se utilizan durante la estimación de riesgos, las cuales son las siguientes:

Probabilidad de ocurrencia. Según el NIST, la probabilidad de ocurrencia se basa en la probabilidad de que una amenaza particular sea capaz de explotar una vulnerabilidad particular, con posibles calificaciones de Baja, Media o Alta [126].

- Alto: el adversario potencial está altamente calificado y motivado y las medidas que se han implementado para proteger contra la vulnerabilidad son insuficientes.
- Medio: el adversario potencial está motivado y capacitado, pero las medidas implementadas para proteger contra la vulnerabilidad pueden impedir su éxito.
- Bajo: el adversario potencial no es calificado o carece de motivación y existen medidas para proteger contra la vulnerabilidad que son parcial o completamente efectivas.

Impacto. El nivel de impacto se determina evaluando la cantidad de daño que podría ocurrir si la vulnerabilidad en cuestión se explotara o se aprovechara de otra manera [126].

- Alto: aprovechar la vulnerabilidad podría ocasionar pérdidas financieras muy significativas, daños graves a la misión o reputación de la organización, o incluso lesiones graves, incluida la pérdida de vidas
- Medio: aprovecharse de la vulnerabilidad podría provocar pérdidas financieras, daños a la misión o reputación de la organización o lesiones humanas.
- Bajo: aprovechar la vulnerabilidad podría ocasionar cierto grado de pérdida financiera o impacto en la misión y reputación de la organización.

Nivel de Riesgo. Una vez que se ha determinado la probabilidad de ocurrencia y el impacto, se puede determinar la calificación de riesgo general, que se define como una función de las dos anteriores. El riesgo general se puede calificar como Bajo, Medio o Alto, lo que proporciona orientación a los responsables de asegurar y mantener los sistemas en cuestión [126].

- Alto: existe un fuerte requisito de implementar medidas adicionales para proteger contra la vulnerabilidad. En algunos casos, se puede permitir que el sistema continúe funcionando, pero se debe diseñar e implementar un plan lo antes posible.
- Medio: se requiere la implementación de medidas adicionales para proteger contra la vulnerabilidad. Un plan para implementar las medidas requeridas debe hacerse de manera oportuna.
- Bajo: el propietario del sistema determinará si implementa medidas adicionales para protegerse contra la vulnerabilidad o si puede optar por aceptar el riesgo y dejar el sistema sin cambios.

3.2.5 Desarrollo

Esta estimación de riesgos está dirigida a los desarrolladores de aplicaciones en SaaS, por lo que se da por hecho que el proveedor de la nube controla la seguridad de las máquinas virtuales y de la red, por lo que los ataques que ocurren en este nivel se consideran con una probabilidad de ocurrencia media debido a las medidas de seguridad implementadas por el proveedor. Además, se asume que el atacante siempre está motivado y bien entrenado para realizar acciones maliciosas.

El desarrollador es responsable de la seguridad de sus aplicaciones, por lo que en ataques que pueden ocurrir en la capa de aplicación se consideran con una probabilidad de ocurrencia alta. A continuación, se muestran los ataques y los niveles de ocurrencia, impacto y riesgo que representa cada uno.

ARP Spoofing. En el caso de que un ataque de este tipo resulte exitoso, una persona malintencionada puede redireccionar una conexión a otro host. La nube debe proporcionar medidas de seguridad para el ataque por lo que la probabilidad de ocurrencia es media, teniendo en cuenta que el atacante estará altamente motivado y capacitado. Por lo tanto, el nivel de riesgo se considera bajo por el tipo de daños que puede ocasionar este incidente.

Backdoor and debug options. Cuando un desarrollador deja habilitada la opción de depuración en su aplicación SaaS un atacante puede obtener permisos de administrador. En el contexto que abarca esta estimación de riesgos el proveedor de nube no controla las aplicaciones SaaS por lo que la probabilidad de ocurrencia es alta. El impacto se considera alto debido a que un atacante al contar con permisos de administrador es capaz de afectar la confidencialidad, integridad y disponibilidad de los datos. Por lo que, el nivel se riesgo se considera alto.

Broken authentication. Este ataque sucede cuando una persona malintencionada logra robar las credenciales de algún usuario. El proveedor de nube implementa medidas de seguridad para prevenir este tipo de escenarios, sin embargo, el ataque puede ocurrir también en el lado del usuario por lo que el nivel de ocurrencia se considera alto. El nivel de impacto es alto ya que una cuenta robada puede tener permisos de administrador perjudicando todo el sistema SaaS. Así que, el nivel de riesgo se considera alto debido a que la aplicación en la nube puede quedar totalmente fuera de servicio.

Buffer overflow. Similar al caso anterior, el atacante puede obtener permisos de administrador al lograr ejecutar código en una aplicación. La probabilidad de ocurrencia se considera alta porque depende de los desarrolladores implementar contramedidas para el ataque. El nivel de impacto se considera alto al afectarse la confidencialidad, integridad y disponibilidad de los datos. Por lo tanto, el nivel de riesgo es alto.

Code Injection. Al ocurrir este tipo de ataques los datos sensibles de los usuarios pueden ser expuestos a una persona malintencionada. La probabilidad de ocurrencia se establece en alta debido a que la responsabilidad de la implementación de medidas de seguridad contra este ataque es del de desarrollador. El impacto se considera bajo debido al impacto en la reputación de la empresa ya que este ataque afecta a los usuarios del SaaS. Por lo tanto, el riesgo se considera medio por la alta probabilidad de ocurrencia.

Cookie Poisoning. En el caso de suceder este ataque una persona malintencionada puede robar datos de autenticación para iniciar sesión como un usuario autorizado en una aplicación en la nube. Al ser un ataque dirigido a los usuarios del SaaS la probabilidad de ocurrencia es alta. El impacto se establece en alto debido a que las credenciales robadas pueden ser de un usuario administrador del sistema por lo que puede afectar en la confidencialidad, integridad y disponibilidad del SaaS. Por lo que, el nivel de riesgo es alto.

Cross-Site Request Forgery (CSRF). Este ataque consiste en el cambio de estados, si la víctima es un usuario normal una persona malintencionada puede transferir fondos o modificar un correo electrónico. En el caso de ser un administrador es posible que toda la aplicación se vea comprometida. Se considera como alta la probabilidad de ocurrencia debido a que esta fuera del alcance del proveedor de la nube. El nivel de impacto se considera alto porque si el usuario atacado es un administrador de la aplicación se comprometerá la confidencialidad, integridad y disponibilidad del SaaS. Por lo tanto, el nivel de riesgo es alto.

DNS Poisoning. Ocurre cuando un usuario es redireccionado a otra dirección IP mediante el envenenamiento de nombres de dominio. La probabilidad de ocurrencia es considerada alta al estar dirigido al usuario de la aplicación en la nube. El impacto se establece en bajo debido a que no afectara de forma negativa al SaaS. Así que, el nivel de riesgo es bajo.

Dumpster Diving. Sucede al recuperar archivos eliminados, el manejo del almacenamiento es responsabilidad del proveedor por lo que la probabilidad de ocurrencia es medio porque existen medidas de prevención. El impacto se establece como bajo ya que la información recuperada por parte de un ataque suele ser datos sensibles de los usuarios. Por lo que, el nivel de riesgo se considera bajo.

Eavesdropping. Este ataque se da cuando un atacante escucha las transmisiones dentro de la red afectando la confidencialidad de los datos. La probabilidad de ocurrencia es media debido a que la administración de la red está bajo la supervisión del proveedor de la nube y este cuenta con medidas de seguridad. El impacto se considera bajo porque no tendría gran impacto en las finanzas, reputación o misión de la empresa. Por lo que, el nivel de riesgo es bajo.

EDoS. Consiste en la inflación de costos por el uso excesivo de recursos de la infraestructura de la nube por parte de una persona malintencionada. La probabilidad de ocurrencia es alta debido a que si un atacante consigue permisos de administrador en la aplicación puede realizar modificaciones con el fin de aumentar la utilización de recursos. El nivel de impacto se establece en alto debido a que puede afectar gravemente las finanzas de una empresa. Por lo tanto, el nivel de riesgo es alto.

Google Hacking. Ocurre cuando un atacante obtiene datos relevantes de la empresa objetivo en motores de búsqueda como Google para un ataque malicioso. La probabilidad de ocurrencia es alta debido a que es posible que los desarrolladores permitan a un motor de búsqueda la indexación de

datos sensibles. El impacto se considera bajo debido a que no afectara considerablemente a las finanzas o misión de la empresa. Por lo tanto, el nivel de riesgo es bajo.

Hash Value Manipulation. Este tipo de ataque ocurre cuando una persona malintencionada modifica el valor hash de un mensaje con el fin de obtener acceso a recursos no autorizados. La probabilidad de ocurrencia se establece en alta debido a que puede ocurrir a nivel aplicación donde el proveedor no maneja la seguridad. El impacto se considera bajo ya que no afecta en gran medida a la reputación, misión y finanzas de la empresa. Por lo tanto, el riesgo es bajo.

Hidden field manipulation. Este ataque sucede cuando el desarrollador de una aplicación web maneja campos ocultos. Un atacante puede aprovechar de forma malintencionada estos campos, por ejemplo, editar un campo en el cual se guarda el total de una compra y reducir el monto a pagar. La probabilidad de ocurrencia se considera alta debido a que es responsabilidad de los desarrolladores evitar esta amenaza. El nivel de impacto se considera alto debido a que puede afectar gravemente las finanzas de una compañía. Por lo que, el nivel de riesgo es alto.

Malware injection and steganography. Ocurre cuando se logra inyectar código malicioso en una aplicación y distribuir código dentro de la red. La probabilidad de ocurrencia es alta debido a que este ataque sucede en la capa de aplicación donde es responsabilidad del desarrollador emplear medidas de seguridad. El nivel de impacto se establece en medio debido a que puede afectar significativamente la misión y reputación de la empresa. Por lo tanto, el riesgo es medio.

Man-in-the-middle Attack. Este ataque se da cuando un atacante logra interceptar un mensaje entre dos nodos con el fin de obtener datos sensibles. La probabilidad de ocurrencia es alta debido a que es responsabilidad del desarrollador implementar canales encriptados entre usuarios y la aplicación. El nivel de impacto se considera alto porque es posible que una persona malintencionada logre robar las credenciales de un usuario administrador. Así que, el riesgo es alto.

Meta Data Spoofing Attack. Sucede cuando un atacante logra alterar los documentos con metadatos proporcionados por un servicio web de esta forma, la persona malintencionada obtiene datos sensibles. La probabilidad de ocurrencia es alta debido a que es responsabilidad del desarrollador el manejo de seguridad en los servicios web. El impacto se considera bajo debido a que no afecta de forma significativa a la empresa. Por lo tanto, el nivel de riesgo es bajo.

Phishing Attack. Sucede cuando un usuario legitimo es redireccionado a un sitio web falso lo que puede provocar el robo de credenciales de un usuario normal o de administrador. La probabilidad de ocurrencia se fija en alta al estar fuera del alcance de la administración del proveedor de la nube. El nivel de impacto se estable en alto debido a que pueden ser las credenciales de un administrador las que se vean comprometidas lo que puede desencadenar en la perdida de confidencialidad, integridad y disponibilidad de los datos. Por lo tanto, el riesgo es alto.

Port scanning. Se da cuando una persona malintencionada realiza un escaneo de puerto con el fin de obtener información relevante para futuros ataques. La probabilidad de ocurrencia se establece en medio debido a que es responsabilidad del proveedor de la nube el manejo de la seguridad dentro de la red de las máquinas virtuales. El impacto se considera bajo debido a que no afecta a la misión de la empresa que maneja el SaaS. Por lo tanto, el riesgo se establece en bajo.

Race Condition. Sucede cuando varios procesos acceden a los mismos datos al mismo tiempo lo que puede provocar que un atacante logre permisos de administrador mientras una aplicación este en modo administrador. La probabilidad de ocurrencia es alta al ser un ataque en el nivel de aplicación donde no es responsabilidad del proveedor de nube el manejo de la seguridad. El impacto se considera alto debido a que es posible afectar de forma significativa las finanzas, reputación y misión de la empresa. Por lo tanto, el riesgo es alto.

Replay attack. Ocurre cuando un atacante reenvía un mensaje con el fin de obtener acceso a recursos no autorizados. La probabilidad de ocurrencia se establece en media debido a que el proveedor de nube maneja la seguridad de la red. El nivel de impacto es bajo debido a que no afecta significativamente a la empresa. Por lo tanto, el nivel de riesgo es bajo.

Reused IP address. Sucede cuando un atacante utiliza una IP reciclada con la cual pueda obtener acceso a recursos no autorizados. La probabilidad de ocurrencia es media debido a que el proveedor de la nube implementa medidas de seguridad en la red. El impacto es bajo debido a que no afecta de forma significativa a la empresa. Por lo tanto, el nivel de riesgo es bajo.

Service Injection Attack. Ocurre cuando un atacante logra redireccionar a los usuarios a un servicio malicioso lo que provoca la pérdida de integridad, robo de cuentas y perdida de disponibilidad. La probabilidad de ocurrencia es media porque es responsabilidad del proveedor el manejo de las máquinas virtuales. El nivel de impacto es alto ya que al verse comprometida una cuenta de

administrador puede afectar gravemente las finanzas, misión y reputación de una empresa. Por lo tanto, el nivel de riesgo es alto.

Shared architectures. Debido a la naturaleza de la nube en la cual las máquinas virtuales comparten recursos es posible que una persona malintencionada pueda robar cuentas al detectar el directorio de ejecución. La probabilidad de ocurrencia es media debido a que el proveedor de nube es el responsable de administrar la seguridad en las máquinas virtuales. El impacto se establece en alto debido a que es posible el robo de una cuenta con privilegios de administrador lo que perjudica gravemente a las finanzas de la empresa, misión y reputación. Por lo tanto, el nivel de riesgo es alto.

Sniffing. Este ataque se da cuando una persona malintencionada captura paquetes dentro de la red. Al ser responsabilidad del proveedor de la nube el manejo de la seguridad en la red la probabilidad de ocurrencia se considera medio. El Impacto se considera bajo debido a que no afecta de forma significativa a la empresa. Por lo tanto, el riesgo se establece en bajo.

Social Engineering. Sucede cuando un empleado es engañado por una persona malintencionada con el fin de obtener datos sensibles de un sistema. La probabilidad de ocurrencia es alta debido a que un administrador puede ser víctima de un engaño comprometiendo toda la aplicación en SaaS lo que puede percudir en la financiación, misión y reputación de la empresa. Por lo que, el nivel de riesgo es alto.

Sybil attack. Se da cuando puede contaminar un sistema creando un gran número de identidades, de esta forma ciertos nodos legítimos pueden sufrir una usurpación de identidad. La probabilidad de ocurrencia se establece en media porque es responsabilidad de la nube el manejo de la seguridad en la red de las máquinas virtuales. El impacto se establece en alto debido a que es posible robar cuentas con privilegios de administrador lo que afecta gravemente a la misión, reputación y finanzas de la empresa. Por lo que, el riesgo es medio.

User to Root Attack. Al ocurrir este ataque una persona malintencionada puede obtener permisos de administrador mediante un desbordamiento de datos en una aplicación. La probabilidad de ocurrencia es alta debido a que el proveedor de la nube no administra la seguridad de la aplicación. El nivel de impacto se considera alto porque afectaría gravemente a la reputación, misión y financiación de una empresa. Por lo tanto, el nivel de riesgo es alto.

XML signature wrapping attack. Este ataque se ejecuta gracias a una vulnerabilidad existente en los mensajes Simple Object Access Protocol (SOAP), si se obtiene al llevar a cabo el ataque una persona malintencionada puede autenticarse como un usuario legítimo. La probabilidad de ocurrencia se estable en baja al ser un servicio web, fuera del alcance del manejo del proveedor de la nube. El impacto se considera alto al ser posible afectar gravemente a la reputación, misión y finanzas de una empresa en el caso de que una cuenta con privilegios de administrador sea objetivo de un ataque exitoso. Por lo tanto, el riesgo se fija en alto.

Zombie Attack. Este ataque ocurre cuando una persona malintencionada lanza un ataque desde varias máquinas llamadas zombis con el fin de denegar un servicio. La probabilidad de ocurrencia se establece en alto debido a que es posible saturar una aplicación SaaS mediante múltiples solicitudes de conexión. El nivel de impacto es alto porque una aplicación no disponible puede afectar gravemente la financiación, reputación y misión de una empresa. Por lo tanto, el riesgo se fija en alto.

En la tabla 7 se muestra un resumen de la estimación de riesgos, destacando los parámetros de probabilidad de ocurrencia, impacto y nivel de riesgo.

Tabla 7: Resumen de la estimación de riesgos

Ataque	Probabilidad de ocurrencia	Impacto	Nivel de riesgo
ARP Spoofing	Media	Bajo	Bajo
Backdoor and debug options	Alta	Alto	Alto
Broken authentication	Alta	Alto	Alto
Buffer overflow	Alta	Alto	Alto
Code Injection	Alta	Bajo	Medio
Cookie Poisoning	Alta	Alto	Alto
CSRF	Alta	Alto	Alto
DNS Poisoning	Alta	Baja	Bajo
Dumpster Diving	Media	Bajo	Bajo
Eavesdropping	Media	Bajo	Bajo
EDoS	Media	Alto	Medio
Google Hacking	Media	Bajo	Bajo
Hash Value Manipulation	Alta	Bajo	Bajo

Hidden field manipulation	Alta	Alto	Alto
Malware injection and steganography	Alta	Medio	Medio
Man-in-the-middle Attack	Alta	Alto	Alto
Meta Data Spoofing Attack	Alta	Bajo	Bajo
Phishing Attack	Alta	Alto	Alto
Port scanning	Media	Bajo	Bajo
Race Condition	Alta	Alto	Alto
Replay attack	Media	Bajo	Bajo
Reused IP address	Media	Bajo	Bajo
Service Injection Attack	Alta	Alto	Alto
Shared architectures	Media	Alto	Alto
Sniffing	Alta	Bajo	Bajo
Social Engineering	Alta	Alto	Alto
Sybil attack	Media	Alto	Medio
User to Root Attack	Media	Alto	Medio
XML signature wrapping attack	Alta	Alto	Alto
Zombie Attack	Alta	Alto	Alto

Como se observa en la tabla anterior, la probabilidad de ocurrencia de la mayoría de los ataques se sitúa en alta por lo que se concluye que es necesario implementar medidas adicionales de seguridad por parte del desarrollador para mitigar los ataques que pueden que ocurrir en una aplicación alojada en el modelo de servicio SaaS.

3.3 Marco de seguridad

El marco de seguridad fue elaborado con la información recaudada durante la revisión sistemática y la estimación de riesgos. Con este marco de referencia, los desarrolladores podrán realizar una evaluación de seguridad en sus aplicaciones bajo el modelo de servicio *SaaS*. La tabla 8 muestra el marco de seguridad. A continuación, se describe cada columna que lo conforma:

1. Ataque: En esta columna se presentan los distintos ataques que pueden ocurrir en un modelo de servicio SaaS.

2. R: Esta columna muestra el responsable de aplicar las medidas de seguridad para el ataque que se está tratando. Los valores pueden ser proveedor de nube (PN) o desarrollador (D)
3. PO: En esta columna se representa la probabilidad de ocurrencia de que suceda un ataque. Los posibles son baja (B), media (M) y alta (A).
4. I: En la columna se muestra el nivel de impacto que tendrá un ataque al ejecutarse correctamente. Los valores son los mismos que en el punto 3.
5. NR: En esta columna se describe el nivel de riesgo que representa un ataque. Al igual que en el caso anterior, los valores se pueden representar con el punto 3.
6. S: Esta columna representa el tipo de amenaza según la clasificación STRIDE de un ataque. Los valores pueden ser falsificación (F), manipulación (M), elevación de privilegios (E), divulgación de información (DI) o denegación de servicio (DE).
7. Riesgo: En esta columna se describe el riesgo que se corre al estar vulnerable al ataque.
8. Medidas: En la columna se muestran las medidas se seguridad que se pueden aplicar a cada ataque con el fin de reducir la probabilidad de ocurrencia.
9. A: Esta columna es para especificar si han sido aplicadas las medidas de seguridad para mitigar los riesgos. Los posibles valores son “si” y “no”, los cuales deben ser ingresados por los desarrolladores usuarios del marco de seguridad.
10. POc: Esta columna representa la nueva probabilidad de ocurrencia, la cual solo aplicará si y solo si todas las contramedidas han sido aplicadas correctamente.
11. NRc: Esta columna es el nuevo nivel de riesgo, el cual solo aplicará si y solo si todas las contramedidas han sido aplicadas correctamente.
12. Estatus: En esta columna se ingresan dos posibles valores, OK o “acción requerida” los cuales son ingresados por el desarrollador. Se dice que se requiere una acción cuando no se ha aplicado alguna contramedida.

Tabla 8: Marco de seguridad

Ataque	**R**	**PO**	**I**	**NR**	**S**	**Riesgo**	**Medidas**	**A**	**POc**	**NRc**	**Estatus**
ARP Spoofing	PN	M	B	B	F	Redirección a host malicioso	Filtrado de tabla ARP fiable, detección, cifrado		M	B	
Backdoor and debug options	D	A	A	A	F	Cambios en la aplicación	El desarrollador debe deshabilitar la opción de depuración		M	M	

Broken authentication	D	A	A	A	M	Robo de identidad	Controles sólidos de autenticación y administración de sesiones		M	M	
							Aplicar automatización para verificar				
							Evitar errores de scripting entre sitios (XSS)				
Buffer overflow	D	A	A	A	E	Ejecución de código malicioso	Aleatorización del conjunto de instrucciones		M	M	
Code Injection	D	A	B	M	F	Pérdida de confidencialidad	Filtrado de contenido activo		M	B	
							Detección de vulnerabilidades de aplicaciones web				
Cookie Poisoning	D	A	A	A	F	Acceso no autorizado	Limpieza regular de cookies		M	M	
CSRF	D	A	A	A	M	Robo de identidad	Token secreto, encabezado de referencia y encabezado de origen		M	M	
DNS Poisoning	D	A	B	B	F	Redirección a app maliciosa	Cifrado y filtrado		M	B	
Dumpster Diving	PN	M	B	B	DI	Robo de información	Definir e implementar una política sobre la eliminación de documentos confidenciales		M	B	
							Hacer un mayor uso de trituradoras de documentos				
Eavesdropping	PN	M	B	B	DI	Pérdida de confidencialidad	Implementar Internet Protocol Security (IP sec)		M	B	
							Implementar políticas de seguridad y software antivirus				
EDoS	PN	M	A	M	DE	Inflación de costos de servicios	Escudo anti EDoS y pruebas gráficas de turing Alosaimi		M	M	
Google Hacking	PN	M	B	B	DI	Pérdida de confidencialidad	No compartir la información confidencial		M	B	
							Uso de herramientas para escanear vulnerabilidades				
Hash Value Manipulation	D	A	B	B	M	Pérdida de confidencialidad	Protocolo sólido para la posesión probable de datos		M	B	
Hidden field manipulation	D	A	A	A	M	Pérdida financiera	Evitar colocar parámetros en campos ocultos		M	A	

Malware injection and steganography	D	A	M	M	M	Inyección de código malicioso	Esquema StegAD		M	M	
Man-in-the-middle Attack	D	A	A	A	DI	Robo de datos sensibles	Aplicación de las medidas de seguridad convencionales		M	A	
							Uso de Dsniff, Cain, Ettercap, Wsniff, Airjack				
Meta Data Spoofing Attack	D	A	B	B	F	Pérdida de confidencialidad	Mantenga la funcionalidad del servicio y otros detalles cifrados		M	B	
							Autenticación fuerte				
Phishing Attack	D	A	A	A	F	Redirección de usuario a sitio malicioso	Algoritmos de cifrado y descifrado autoadaptables y autoajustables		M	M	
							Implementar Transport Layer Security (TLS)				
							Utilización de certificados y HTTPS				
Port scanning	PN	M	B	B	DI	Exposición de puertos abiertos	Bloqueo de puertos		M	B	
Race Condition	D	A	A	A	E	Obtención de permisos de administrador	Técnica de actualización de predicados		M	M	
Replay attack	PN	M	B	B	M	Acceso a datos no autorizado	Esquema de codificación estocástica		M	B	
Reused IP address	PN	M	B	B	DI	Acceso a recursos no autorizado	Eliminación de caché		M	B	
Service Injection Attack	D	A	A	A	F, M	Solicitudes de usuarios redireccionados a servicio malicioso	Fuerte aislamiento		M	M	
							Mecanismo de identificación sólidos				
							Implementación de la integridad del servicio				
Shared architectures	PN	M	A	A	F	Robo de cuentas	Analizar el código binario de la aplicación		M	M	
Sniffing	D	A	B	B	DI	Datos sensibles expuestos	Implementación de SSL y TLS		M	B	
							Implementación de IPsec				
Social Engineering	D	A	A	A	DI	Robo de datos sensibles	Centrarse principalmente en las políticas de seguridad y la formación del personal		M	M	

Sybil attack	PN	M	A	M	E	Obtención de permisos de administrador	Una solución basada en criptografía de clave simétrica		M	M	
User to Root Attack	PN	M	A	M	E	Obtención de permisos de administrador	Contraseña segura		M	M	
							Mecanismo de autenticación fuerte				
XML signature wrapping attack	D	A	A	A	E	Atacante puede autenticarse como usuario legítimo	Escáneres automáticos y verificación manual		M	M	
Zombie Attack / Denail of Service	PN / D	A	A	A	DE	Pérdida de disponibilidad	Autenticación robusta		M	M	
							Uso de IDS/IPS				
							Un análisis periódico y exhaustivo del sistema				
							Filtrar paquetes ICMP y SYN				
							Filtrar direcciones IP privadas				

3.4 Conclusiones del marco de seguridad

Detrás del desarrollo del marco de seguridad se encuentra una revisión sistemática para conocer los distintos tipos de ataques y amenazas a los que una aplicación en SaaS está sujeta y una estimación de riesgos para conocer las probabilidades de ocurrencia, el impacto y el nivel de riesgo. Es importante resaltar que la revisión sistemática fue presentada en forma de un artículo en inglés en el *Journal of Computer Security* con el nombre de *"A systematic review of security threats and countermeasures in SaaS"* [10]. Por otro lado, el marco de seguridad facilita al desarrollador la evaluación de seguridad de sus aplicaciones en SaaS. Esto se debe a que, con el uso del marco de seguridad, el desarrollador tiene conocimiento sobre los distintos ataques que pueden ocurrir en sus implementaciones. Además, conoce las probabilidades de ocurrencia de dichos ataques, su impacto, sus contramedidas y quién es el responsable de aplicar las medidas de seguridad descritas en el marco. Por último, para una evaluación de seguridad más eficaz se recomienda la utilización de la herramienta *SaaS Neer* propuesta en este trabajo.

CAPÍTULO 4. Desarrollo de herramienta para la detección de vulnerabilidades en SaaS

En este capítulo se describe el procedimiento del desarrollo de una herramienta para la detección de vulnerabilidades en SaaS, la cual tiene como nombre *Saas Neer*. Esta herramienta está basada en el proyecto WebMap [130] disponible en Github. A la fecha de clonación del proyecto de Github [2], aún

[2] https://github.com/SabyasachiRana/WebMap clonado el día 01 de febrero del 2020

no estaba listo para producción debido a la falta del manejo de usuarios y de una forma eficaz de proteger el servidor en el que corre la aplicación debido a la necesidad indispensable de ser ejecutada con permisos de administrador. Es importante resaltar que el proyecto *WebMap* se rige bajo la licencia *GNU General Public License v3.0*, la cual permite:

- Uso comercial
- Modificación
- Distribución
- Uso de patentes
- Uso privado

Al momento de su clonación, *WebMap* contaba con las siguientes características:

- Importar y analizar archivos *Nmap XML* en busca de vulnerabilidades.
- Estadísticas y gráficos sobre servicios descubiertos, puertos, sistema operativo, entre otros.
- Crear un informe *PDF* con las vulnerabilidades detectadas.
- Posibilidad de ejecutar en servidores propios.
- Corre bajo el esquema de una aplicación tradicional.

Por otra parte, WebMap presentaba los siguientes problemas:

- No es posible ejecutarlo en un ambiente de producción debido a fuertes problemas de seguridad.
- Se requiere generar un token para utilizarlo, lo cual no es posible debido a un problema con un comando de instalación.
- La aplicación no genera de forma correcta un archivo *XML* para su posterior análisis.
- Si se selecciona un reporte y se elimina, la aplicación se detiene por error.
- Dependiente de Docker.
- El proyecto tiene un año sin ninguna actualización o mejora.

Con base a lo descrito anteriormente sobre *WebMap*, se desarrolló *SaaS Neer*, una herramienta para la detección de vulnerabilidades, en la cual se agregaron las siguientes características:

- Solución de problemas de *WebMap*.
- Implementación de un software como servicio con nivel 2 de madurez.
- Capacidad para el manejo de usuarios mediante la implementación de *AWS Cognito*.
- Los usuarios no pueden ver los datos de otros usuarios.

- Se agregó la capacidad de analizar dependencias de un proyecto en busca de vulnerabilidades con el empleo de OWASP.
- Descarga de *PDF* con el informe de vulnerabilidades en dependencias de un proyecto.
- Todos los usuarios utilizan la misma instancia de la aplicación.

La aplicación se desarrolló bajo una arquitectura SaaS con un nivel 2 de madurez, lo cual se describe detalladamente en la sección 4.2, con el fin de obtener las ventajas que proporciona el uso de la nube. Como proveedor se eligió a Amazon AWS debido a la posibilidad de contar con cuentas *Free Tier*, lo cual resultaba ideal para el desarrollo de la herramienta debido a que permite llevar a cabo pruebas sin costo. Incluso es posible utilizarlo en producción sin costo, siempre y cuando no se sobrepase las políticas del uso gratis como la cantidad de memoria, ancho de banda, procesamiento, entre otros.

4.1 Metodología de desarrollo

Para poder tomar una decisión respecto al procedimiento de desarrollo de la herramienta fue necesario establecer los requisitos funcionales y no funcionales con los que debería contar la aplicación. Se establecieron como requisitos funcionales los siguientes:

1. Capacidad de analizar puertos en busca de servicios en un servidor.
2. Encontrar vulnerabilidades en servicios.
3. Hallar vulnerabilidades en dependencias de código abierto en proyectos de desarrollo.
4. Proveer un reporte de las vulnerabilidades detectadas descargable con formato amigable para el usuario.

Por otro lado, los requisitos no funcionales de la aplicación son los siguientes:

1. Realizar los análisis de seguridad en pocos minutos.
2. Tener tiempo de respuesta de segundos a las solicitudes de los usuarios.
3. La aplicación debe ser fiable por lo que no debe interrumpirse un análisis por algún error.
4. La herramienta no debe guardar los análisis de los usuarios.
5. Implementación de la herramienta en una arquitectura SaaS.

Para el desarrollo de la herramienta se utilizó el modelo de desarrollo por prototipos debido a que se conocían los objetivos del sistema, pero se carecía del conocimiento de los valores de entrada y de salida. La construcción de prototipos es un proceso que facilita al ingeniero de software el desarrollo de la aplicación. El prototipo suele tomar una de las tres formas siguientes [131]:

- Un modelo en papel o en computadora que describe la interacción hombre máquina, de tal forma que al usuario le resulte más simple la comprensión del funcionamiento. Por ejemplo, en el caso de que sea necesario desarrollar un cajero automático, se puede crear un sistema que simule el uso del cajero sin la necesidad de conectarse a una base de datos y sin entregar dinero. De esta forma, es posible que el usuario final tenga una idea de cómo será el funcionamiento del sistema sin tener que crearlo y así poder discutir el procedimiento de desarrollo. Por lo general, en un prototipo no se simulan todos los módulos de un sistema, pero de ser necesario se irán construyendo más prototipos según el avance del desarrollo lo requiera.
- Un modelo que requiera la implementación de una función altamente importante. En este caso, utilizando el ejemplo del cajero automático, el prototipo podría simular todo el procedimiento a seguir para obtener dinero.
- Un programa adecuado en cierta parte a la ampliación que se plantea desarrollar. Por lo que, se puede disponer de un software parecido a un cajero automático que, al presentarlo al usuario final, el analista sea capaz de identificar las necesidades del cliente y de esta forma obtener los requisitos de la aplicación a desarrollar.

La función principal del prototipo es identificar los requisitos de la aplicación. A continuación, se muestras las etapas necesarias para la construcción de un prototipo [131]:

1. Obtención de requisitos. En esta etapa se definen los objetivos generales y específicos que sea desean lograr con el prototipo.
2. Diseño. En esta etapa se realiza un diseño rápido del prototipo.
3. Construcción. Durante este proceso se elabora el prototipo.
4. Evaluación. El usuario realiza una evaluación para refinar los requisitos del software.
5. Refinamiento. Durante esta sección se crea una iteración en la que el prototipo satisface las necesidades del usuario mientras le otorga más conocimiento de la aplicación al ingeniero de software.
6. Producto final.

Durante el desarrollo de los prototipos se recomienda seguir las siguientes indicaciones [131]:

- Es preferible ir evolucionando el prototipo para evitar realizarlo y sustituirlo por uno nuevo.
- Cualquier aplicación nueva en la que se sospeche que en su funcionalidad se presente el riesgo de no ser aceptada por el usuario final, significa que esa aplicación requiere un prototipo rápido.

- Por lo general, en un proyecto de prototipos, el 50% del desarrollo recae en el usuario. Los equipos de desarrollo suelen ser 50% usuarios y la otra mitad desarrolladores.
- Se debe estimar que el primer prototipo del sistema sea imperfecto con la finalidad de reducir costos a la hora de proponer el prototipo final, debido a que responderá de mejor forma a las necesidades del usuario.

Para la gestión de versión del desarrollo se eligió a GitHub debido a que cuenta con la posibilidad de tener proyectos privados con colaboradores infinitos.

4.2 Arquitectura de la herramienta

La función principal de *SaaS Neer* es elaborar un análisis de otras aplicaciones que estén corriendo en SaaS. Sin embargo, también será posible escanear otro tipo de aplicaciones como servicios web, APIs, entre otros. Esto se logra de dos formas:

1. La aplicación busca los servicios que corren en un servidor e identifica sus respectivas versiones. Posteriormente, se realiza una búsqueda de las vulnerabilidades existentes con la API proporcionada por circl.lu. Al final, la aplicación otorgará al cliente un *PDF* descargable con el reporte de los problemas de seguridad encontrados.
2. El cliente, mediante un formulario, sube los archivos que contienen las dependencias de sus proyectos. Al finalizar la carga de archivos, la aplicación identifica las versiones de estas dependencias y con la información de una base de datos proporcionada por NIST, descarga una lista de vulnerabilidades aplicadas a las dependencias del proyecto. Por último, la herramienta otorga al cliente un reporte en *PDF* en el cual se puede visualizar la lista de problemas de seguridad.

Para el diseño del SaaS se empleó el nivel 2 de madurez, donde todos los *arrendatarios* que representan a cada cliente van a compartir una sola instancia de la aplicación y sus distintas necesidades se manejarán mediante opciones de configuración. Este nivel fue elegido debido a que el sistema no va a almacenar los resultados de los análisis, por lo que no será necesario la utilización de una base de datos para el almacenamiento de datos sensibles. En la figura 3 se muestra la representación gráfica de esta arquitectura.

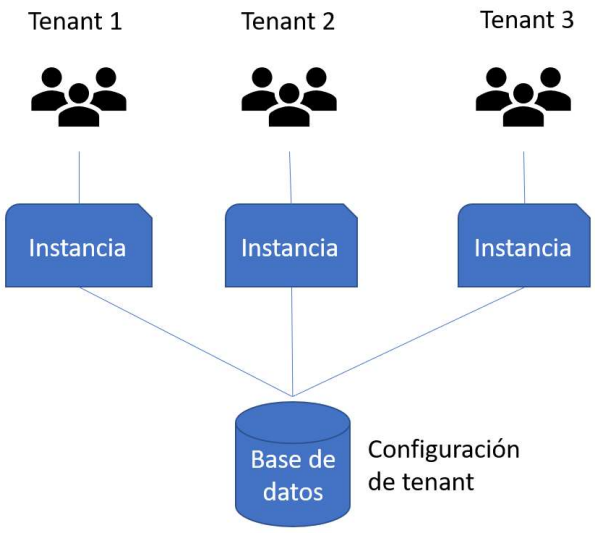


Figura 3: Nivel 2 de maduración de *SaaS Neer*

Para la administración de usuarios se utiliza el servicio de *Cognito* proporcionado por Amazon AWS. La decisión de emplear este servicio fue debido a que AWS es el que administra la autenticación de la aplicación. Además, permite el uso de 50,000 cuentas de usuarios totalmente gratis. Una de las principales ventajas que implica contar con *Cognito* es el ahorro de tiempo para el desarrollo de un módulo de autenticación, mejor seguridad ya que está constantemente actualizado y la facilidad de su implementación en cualquier ambiente de desarrollo. En la figura 4 se muestra el diagrama de la arquitectura de la aplicación *SaaS Neer,* así como los servicios utilizados y la interconexión entre estos.

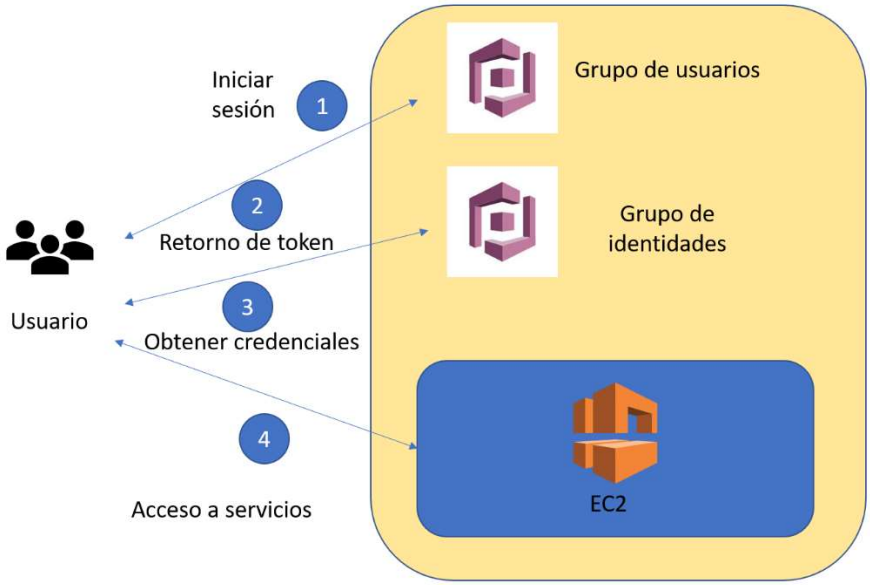


Figura 4: Arquitectura de *SaaS Neer*

4.3 Desarrollo

El objetivo era desarrollar una herramienta para la detección de vulnerabilidades en SaaS con base al marco de seguridad presentado en la sección 3.3. Así que, era necesario que la herramienta detectara los ataques que pueden ocurrir en una aplicación en SaaS. Una vez que la herramienta muestre los resultados, el usuario de la herramienta puede dirigirse al marco de seguridad para localizar el ataque, su impacto, sus contramedidas, entre otros. Como fase previa al desarrollo de la herramienta, se realizó la búsqueda de otras aplicaciones, de esta forma se obtendría una serie de características y carencias de las aplicaciones existentes, con el fin de tenerlas en cuenta y ofrecer una herramienta superior a las encontradas. Durante esta fase de búsqueda se encontró con un proyecto que era capaz de encontrar vulnerabilidades en dependencias de un desarrollo. Así que, se optó por implementar esta función al producto final de este trabajo. Lo que resultó en la herramienta desarrollada, conocida como *SaaS Neer*. Por lo tanto, se estableció que *SaaS Neer* debe estar alojada en la nube, ser capaz de encontrar vulnerabilidades en aplicaciones en SaaS y en dependencias de desarrollos, tener la capacidad de generar reportes de los escaneos y tener una interfaz amigable al usuario. En las siguientes secciones se muestra el procedimiento llevado a cabo para el desarrollo de *SaaS Neer*.

4.3.1 Búsqueda de aplicaciones de escaneo de vulnerabilidades en aplicaciones

Se ejecutó una búsqueda en el motor de Google con la siguiente cadena “herramientas para la detección de vulnerabilidades en SaaS” la cual no arrojó ningún resultado favorable. Por lo que, se

optó por buscar alternativas utilizadas en aplicaciones en sitio y buscar la forma de implementarlas en SaaS. Se sabe que la aplicación gratuita *Nmap* descrita en la sección 2.11.1 es capaz de encontrar los servicios que se están ejecutando en los puertos de un host. Con esta información se decidió enfocar la investigación en la búsqueda de herramientas que implementen *Nmap* en sus escaneos. Al realizar una investigación de herramientas implementadas con *Nmap*, con el fin de obtener vulnerabilidades a partir de puertos abiertos, fue posible el hallazgo de *Flan Scan*, un proyecto elaborado por *Cloudflare,* descrito en la sección 2.11.3.

Flan Scan detecta los servicios de una red y posteriormente busca esos servicios en una base de datos CVE para la identificación de vulnerabilidades [120]. Sin embargo, el proyecto de *Flan Scan* no cuenta con una interfaz amigable para los usuarios y es necesario instalarlo en un dispositivo local para su ejecución.

Al estudiar el proyecto *Flan Scan* fue posible conocer que la herramienta *Nmap* puede detectar vulnerabilidades CVE en los servicios mediante scripts y con la implementación de una API otorgada por vulners.com de forma gratuita. Así que, se decidió por buscar un proyecto en Github, el cual tuviera algún desarrollo de *Nmap* en un servicio web. De esta manera, se realizaría la migración del proyecto a un ambiente SaaS para posteriormente utilizar los scripts de código abierto que fueron implementados en *Flan Scan*. Sin embargo, se obtuvo información de un proyecto en desarrollo conocido como *Webmap* el cual se describe en la sección 4.3.4 Así que, la decisión final fue corregir una serie de errores de programación y diseño de *Webmap*, migrarlo a una arquitectura de nivel 2 de madurez SaaS, agregar servicios de autenticación como *AWS Cognito* y agregar la función de búsqueda de vulnerabilidades en dependencias.

4.3.2 Búsqueda de aplicaciones de escaneo de vulnerabilidades en dependencias

Para el módulo de escaneo de vulnerabilidades en dependencias se realizó una búsqueda similar a la sección anterior de otras aplicaciones con la misma función. De esta forma se encontró el proyecto *OWASP Dependency Check,* elaborado por OWASP y ya descrito en la sección 2.12.2. Este proyecto es totalmente compatible con los módulos de *SaaS Neer* debido a que es posible instalarlo en un ambiente *Linux* y ser ejecutado mediante comandos, lo cual resultaba ideal para ser anexada como una función extra.

Para su uso es necesario instalar *brew*, un administrador de paquetes, para llevar a cabo este procedimiento requiere el siguiente comando:

```
sudo apt install linuxbrew-wrapper
```

Posteriormente, se instala OWASP con el siguiente comando:

```
brew install dependency-check
```

Ahora es posible analizar las dependencias de un proyecto desde la terminal con la siguiente sintaxis:

```
dependency-check.bat --project "My App Name" --scan "c:\java\application\lib"
```

4.3.3 Implementación de Django

Durante la fase de desarrollo se utilizó *Django*, un marco escrito en *Python* compacto para el desarrollo rápido en entornos de ritmo rápido con buena compatibilidad con bases de datos relacionales [132]. En la página web del proyecto de *Django* [133] es posible visualizar las siguientes ventajas:

- Rápido. *Django* fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible.
- Seguro. *Django* toma en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes.
- Escalable. Algunos de los sitios más activos de la Web aprovechan la capacidad de *Django* para escalar de manera rápida y flexible.

El ambiente de desarrollo de *SaaS Neer* fue alojado en un sistema operativo *Linux* utilizando *Visual Studio Code* debido a las licencias gratuitas, compatibilidad, soporte y rendimiento. El sistema empleado para la gestión de versión fue *GitHub* debido a las siguientes ventajas:

- Es posible personalizar cualquier servicio host en la nube.
- Seguimiento de errores.
- Búsqueda rápida.
- La comunidad brinda buen soporte.
- Permite descargar como archivo el código fuente.
- Posibilita la importación en Git, SVN o TFS.

4.3.3 Descripción de *WebMap*

El proyecto de *WebMap* constaba de una serie de módulos los cuales se pueden observar de forma general en la figura 5. A continuación, se describe la utilidad de cada sección de la aplicación:

- urls.py. En este archivo se encuentran las reglas de direccionamiento. Es al primer módulo al que se conecta el usuario debido a que aquí se establecen las rutas de las conexiones.
- views.py. En esta sección se establecen las funciones de vista. Una función de vista es una función de *Python* que toma una solicitud web y devuelve una respuesta web. Esta respuesta puede ser el contenido de una página web, una redirección, un error 404, entre otros.
- functions.py. En este módulo se escribieron funciones utlizadas como la obtención de la lista de puertos, cantidad de host activos, definiciones de colores de etiquetas, entre otros.
- api.py. Aquí fue escrito una *RESTful API* en la cual es posible solicitar información sobre archivos escaneados previamente mediante métodos GET.
- pdf.py. Este módulo fue creado con la finalidad de generar los reportes de los análisis en formato *PDF* y realizar la descarga del archivo generado.
- functions_nmap.py. En este archivo se colocaron las funciones que invocan a la herramienta *nmap* para llevar a cabo los análisis de seguridad.
- token.py. Módulo encargado del inicio de sesión de un usuario. El usuario tenía la tarea de ejecutar otro archivo con permisos de administrador en ambiente local para que *WebMap* generara un token y así poder utilizar la aplicación.
- async.js. En este archivo escrito en javascript se encuentran las funciones que se ejecutan al realizar ciertos eventos como puede ser un clic en un icono.

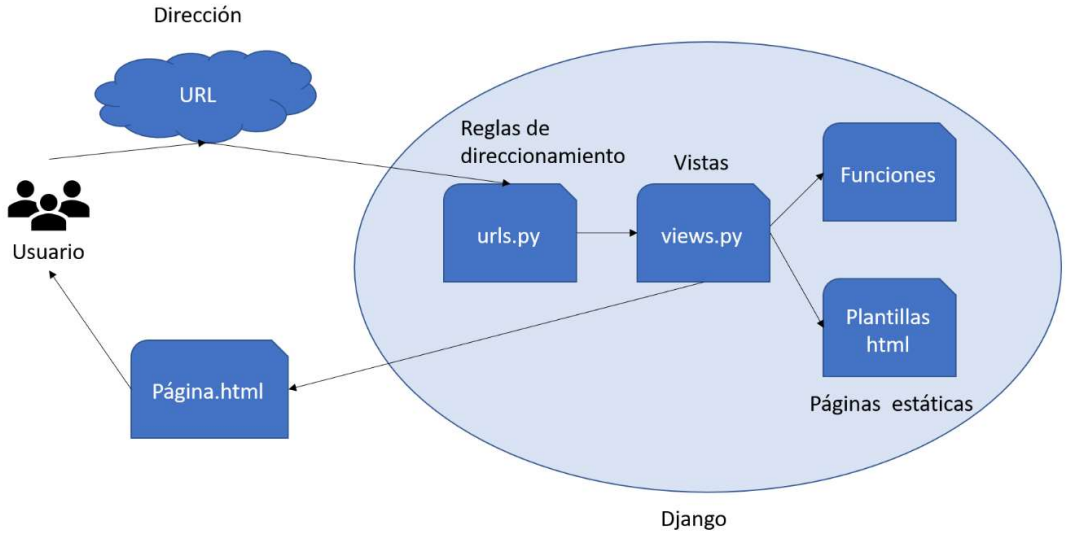


Figura 5: Diseño de WebMap.

4.3.4 Mejoras en SaaS Neer

Durante el desarrollo de *SaaS Neer* se realizaron varias mejoras, correcciones de errores y modificaciones a los módulos mencionados anteriormente. Además, se agregaron distintas funciones para la implementación de *AWS Cognito* con la finalidad de administrar a los usuarios que van a iniciar sesión en la aplicación. De esta forma se sustituye los métodos de autenticación de *Django* por las de *AWS Cognito*. Con esta implementación se obtuvieron las siguientes características dentro de la aplicación *Saas Neer*:

- Alta de nuevos usuarios.
- Capacidad de restaurar contraseñas olvidadas.
- Inicio de sesión para la utilización de la aplicación.
- Bloqueo de accesos no autorizados a la aplicación.

Los pasos para implementación de *Cognito* fueron los siguientes:

1. Creación de cuenta en *Amazon Web Services.*
2. Navegación a *Cognito* mediante el enlace https://console.aws.amazon.com/cognito/home .
3. Creación de grupo de usuarios con nombre *Saas Neer*.
4. Permitir el acceso al grupo de usuarios mediante la aplicación *SaaS Neer*. Al realizar este proceso se crea un identificador único, el cual será utilizado posteriormente para la conexión al servicio de autenticación.
5. Se realizó una búsqueda de librerías que permitieran la ejecución de *Cognito* dentro de *Django*. De esta forma, fue posible encontrar el proyecto *django_cognito* elaborado por el usuario de *GitHub Olorin92* en la liga https://github.com/Olorin92/django_cognito. Esta librería facilita la implementación de *Cognito* dentro de *Django*.
6. Descarga del proyecto *django_cognito* en el mismo directorio de *Saas Neer.*
7. Se agregaron las llaves secretas propias de *AWS* y de la aplicación, así como el *ID* del grupo de usuarios de *Cognito* en el archivo *Settings.py* del proyecto *django_cognito.*
8. Creación de páginas *HTML* de *login* y *register*.
9. Desarrollo de funciones para iniciar sesión, alta de usuario y recuperación de contraseña en *views.py* de *Saas Neer* para el manejo de la librería *django_cognito.*

En el siguiente código se muestra la función para dar de alta un usuario desde *Saas Neer*.

```
1.  @require_http_methods(['POST'])
2.  def sign_up(request, param_mapping=None):
3.      try:
4.          data = json.loads(request.body.decode('utf-8'))
```

```
        username = parse_parameter(data, param_mapping, 'username')
        result = helpers.sign_up(data)
        return JsonResponse(result)
    except CognitoException as ex:
        return JsonResponse(ex.args[0], status=ex.status)
    except ValueError as ex:
        return JsonResponse({"error": ex.args[0]}, status=400)
    pass
```

A continuación, se muestran las modificaciones elaboradas a los archivos existentes:

- functions.py. Eliminación de inicio de sesión mediante un token. La función eliminada es la siguiente:

```
tokenhash = open('/root/token.sha256').read().strip()
    if tokenhash == hashlib.sha256(token.encode('utf-
8')).hexdigest():     if tokenhash == hashlib.sha256(token.encode('utf-8')).hexdigest():
        return True        return True
    return False       #return True
    return True
```

- views.py. Se agregaron las distintas funciones para la implantación de *AWS Cognito*. Además, fue corregido un error al generar archivos *XML* a los análisis, ahora el escáner puede leer archivos con extensión .xml lo que permite a *SaaS Neer* leer de forma correcta los nuevos análisis agregados. El código modificado para la lectura de archivos *XML* resultó de la siguiente forma:

```
for i in xmlfiles:
            if re.search('\.xml$', i) is None:
            if re.search('\$', i) is None:
                continue
            portstats = {}
            xmlfilescount = (xmlfilescount + 1)
```

- pdf.py. Modificación de *pdf.py* para generar reporte con leyendas y logos de *SaaS Neer.*
- urls.py. En el archivo *urls.py* se agregó el nuevo direccionamiento, el cual contiene la ruta de la función para realiza un nuevo análisis de seguridad en dependencias de código abierto. Los direccionamientos resultaron de la siguiente forma:

```
urlpatterns = [
```

```
    path('', views.index, name='index'),
    path('setscanfile/<scanfile>', views.setscanfile, name='setscanfile'),
    path('<address>/', views.details, name='details'),
    path('port/<port>/', views.port, name='port'),
    path('service/<filterservice>/', views.index, name='service'),
    path('portid/<filterportid>/', views.index, name='portid'),
    path('api/v1/scan/<scanfile>/<faddress>', api.apiv1_hostdetails, name='apiv1_hostdetails'),
    path('api/v1/scan/<scanfile>', api.apiv1_hostdetails, name='apiv1_hostdetails'),
    path('api/v1/scan', api.apiv1_scan, name='apiv1_scan'),
    path('api/v1/nmap/scan/active', functions_nmap.nmap_scaninfo, name='apiv1_scan_active'),
    path('api/v1/nmap/scan/new', functions_nmap.nmap_newscan, name='apiv1_scan_new'),
    path('api/v1/nmap/scan/new_dependency', functions_nmap.dependency_newscan, name='apiv1_scan_new_dependency'),
    path('api/v1/nmap/ndiff/<f1>/<f2>', ndiff.ndiff, name='ndiff'),
    path('api/setlabel/<objtype>/<label>/<hashstr>/', api.label, name='api_label'),
    path('api/rmlabel/<objtype>/<hashstr>/', api.rmlabel, name='api_rmlabel'),
    path('api/pdf/', api.genPDF, name='genPDF'),
    path('api/getcve/', api.getCVE, name='getCVE'),
    path('api/savenotes/', api.saveNotes, name='genPDF'),
    path('api/rmnotes/<hashstr>/', api.rmNotes, name='api_rmnotes'),
    path('api/<address>/<portid>/', api.port_details, name='api_port'),
    path('view/login/', views.login, name='login'),
    path('view/pdf/', pdf.reportPDFView, name='reportPDFView'),
    path('view/network/', network.visjs, name='network_view'),
    path('view/ndiff/<f1>/<f2>', views.scan_diff, name='ndiffview')
]
```

- main.html. Se agregó un nuevo icono y evento correspondientes al análisis de dependencias en *main.html*. La línea agregada es la siguiente:

```
<li><i class="material-icons">add_box</i> <a href="#!" onclick="javascript:newscanDependency();">New Dependency scan</a></li>
```

- async.js. Fue agregada una nueva función para responder al evento de realizar un nuevo escáner de dependencias, el código es el siguiente:

```
$('#modalfooter').html('<button onclick="javascript:startscandepency();" class="btn green">Start</button>');
```

- functions_security.py. El archivo *functions_nmap.py* fue renombrado a *functions_security.py*. Para corregir el problema de *WebMap* que no creaba los reportes con

extensión *XML* se realizó un par de modificaciones en la función de escaneo con *Nmap*, resultando de la siguiente forma:

```
1. def nmap_newscan(request):
2. if request.method == "POST":
3. if(re.search('^[a-zA-Z0-9\_\-\.]+$', request.POST['filename']) and re.search('^[a-zA-Z0-9\-
   \.\:\=\s,]+$', request.POST['params']) and re.search('^[a-zA-Z0-9\-
   \.\:\/\s]+$', request.POST['target'])):
4. res = {'p':request.POST}
5. a = 'testr'
6. filename = 'testrar'
7. os.popen('nmap '+request.POST['params']+' --
   script='+settings.BASE_DIR+'/nmapreport/nmap/nse/ -
   oX /tmp/'+request.POST['filename']+'.active '+request.POST['target']+' > /dev/null 2>&1 &
   & '+
8. 'sleep 10 && mv /tmp/'+request.POST['filename']+'.active /opt/xml/'+request.POST['filena
   me']+'.xml')
9. return HttpResponse(json.dumps(res, indent=4), content_type="application/json")
10. else:
11. res = {'error':'invalid syntax'}
12. return HttpResponse(json.dumps(res, indent=4), content_type="application/json")
```

Se agregó una nueva función para ejecutar el análisis de dependencias. El fragmento de código agregado es el siguiente:

```
1. os.popen('cd /home/linuxbrew/.linuxbrew/bin &&
2. ./dependency-check --project "segunda prueba" --scan "/home/mike/Downloads/" &&
3. mv dependency-check-report.html /home/mike').
```

Referente al logo que se muestra al ejecutar *SaaS Neer*, el cual se puede apreciar en la figura 6, fue diseñado sin costo. Se aplicaron algunas limitantes, con el servicio proporcionado por la página [134]. En el logo es posible visualizar el nombre del proyecto, así como su eslogan “A SAAS SECURITY SCANNER”.

Figura 6: logo de la herramienta.

4.3.4 Obstáculos en el desarrollo de *SaaS Neer*

SaaS Neer emplea una *API* proporcionada por El Centro de Respuesta a Incidentes Informáticos de Luxemburgo (CIRCL por sus siglas en inglés). *SaaS Neer* enviaba mediante un *GET* el código *CPE* del servicio y la *API* regresaba la lista de vulnerabilidades. Desafortunadamente, el CIRCL deshabilitó esta función debido a un mal uso, como se muestra en la figura 7. Sin embargo, facilitaron el enlace a su proyecto de *GitHub* llamado *CVE Search*. Por lo tanto, era posible la implementación de la *API* para la búsqueda de vulnerabilidades en un ambiente local junto a *SaaS Neer*.

Figura 7: Circl deshabilita la búsqueda de vulnerabilidades en servicios por mal uso.

4.3.5 Implementación de *CVE Search* en el servidor local

Primeramente, para la implementación de *CVE Search* se descarga el código de *GitHub* con el siguiente comando:

Git clone https://github.com/cve-search/cve-search.git

Posteriormente, se lleva a cabo la instalación de las dependencias del proyecto con el siguiente comando:

```
sudo pip3 install -r requirements.txt
```

CVE Search require la instalación de *MongoDB*, un sistema de base de datos *NoSQL* [135]. Este se instala con el siguiente comando:

```
sudo apt-get install -y mongodb-org
```

Para iniciar el servicio de *MongoDB* se emplea el siguiente comando:

```
sudo systemctl start mongod
```

Ahora, para llenar la base de datos de *CVE Search* se insertan los siguientes comandos:

```
./sbin/db_mgmt_cpe_dictionary.py -p
./sbin/db_mgmt_json.py -p
./sbin/db_updater.py -c
```

El procedimiento anterior puede tardar aproximadamente una hora en un equipo con dos núcleos y 8GB de RAM. El siguiente comando se utiliza para actualizar la base de datos:

```
./sbin/db_updater.py -v
```

Es posible buscar vulnerabilidades directamente desde la terminal con los siguientes comandos:

```
./bin/search.py -p cisco:ios:12.4
./bin/search.py -p cisco:ios:12.4 -o json
./bin/search.py -f nagios -n
./bin/search.py -p microsoft:windows_7 -o html
```

Para realizar solicitudes a la *API* de *CVE Search* desde *SaaS Neer* es necesario iniciar la interfaz web mediante los siguientes comandos:

```
cd ./web
./index.py
```

Es posible acceder a la lista de vulnerabilidades desde el navegador web como se muestra en la figura 8.

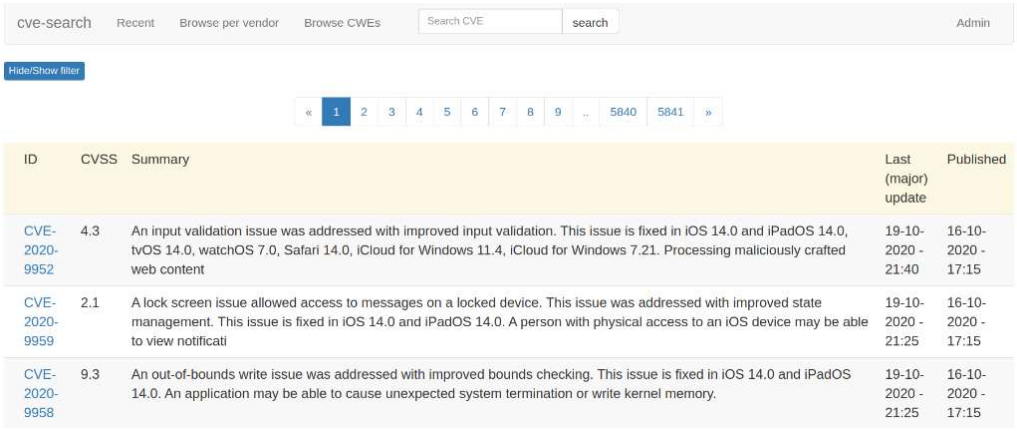


Figura 8: Interfaz web de la API de CVE Search.

Se realizaron un par de modificaciones en el código de *SaaS Neer* para apuntar el *GET* al servidor local. Anteriormente, el código redireccionaba a la API de CIRCL.LU. A continuación, se muestra la modificación realizada:

```
1.  for cpestr in cpecve['cpe'][i]:
2.              print(cpestr)
3.                  r = requests.get('http://127.0.0.1:5000/api/cvefor/'+cpestr)
```

Durante las pruebas de conexión entre *SaaS Neer* y la *API* de *CVE Search* se encontró un problema. *SaaS Neer* realizaba el *GET* a la *API* para la búsqueda de vulnerabilidades con *CPE* versión 2.2 cuando la *API* requería *CPE* versión 2.3. Por lo que, la *API* no regresó ningún resultado en todas las búsquedas realizadas. Por lo tanto, era necesario convertir el código enviado a *CPE* 2.3.

Para convertir el CPE 2.2 a CPE 2.3 se utilizó el siguiente código:

```
1.  for cpestr in cpecve['cpe'][i]:
2.              print(cpestr)
3.              if re.search('^cpe:[^:]+:[^:]+:[^:]+:.+$', cpestr):
4.                  #r = requests.get('http://cve.circl.lu/api/cvefor/'+cpestr)
5.                  cpestr = cpestr.replace("cpe:/", "cpe:2.3:")
6.                  cpestr = cpestr.replace("::", ":-:")
7.                  cpestr = cpestr.replace("~-", "~")
8.                  cpestr = cpestr.replace("~", ":-:")
9.                  cpestr = cpestr.replace("::", ":")
```

```
10.                 cpestr = cpestr.strip(":-")
11.                 cpestr = unquote(cpestr)
12.             r = requests.get('http://127.0.0.1:5000/api/cvefor/'+cpestr)
```

Donde en la tercera línea se comprueba la versión del *CPE*. Si es versión 2.2 procede a convertirlo a 2.3 en base a la nomenclatura de ambas versiones. Por último, en la línea 12 se realiza el *GET* a la *API* de *CVE Search* con el código *CPE* 2.3 de forma correcta como se observa en la figura 9.

```
16:32:37] "GET /api/cvefor/cpe:2.3:a:vsftpd:vsftpd:2.3.4 HTTP/1.1" 200 -
16:32:37] "GET /api/cvefor/cpe:2.3:a:openbsd:openssh:4.7p1 HTTP/1.1" 200 -
16:32:37] "GET /api/cvefor/cpe:2.3:a:isc:bind:9.4.2 HTTP/1.1" 200 -
16:32:37] "GET /api/cvefor/cpe:2.3:a:apache:http_server:2.2.8 HTTP/1.1" 200 -
16:32:38] "GET /api/cvefor/cpe:2.3:a:proftpd:proftpd:1.3.1 HTTP/1.1" 200 -
16:32:38] "GET /api/cvefor/cpe:2.3:a:mysql:mysql:5.0.51a-3ubuntu5 HTTP/1.1" 200 -
16:32:38] "GET /api/cvefor/cpe:2.3:a:postgresql:postgresql:8.3 HTTP/1.1" 200 -
16:32:38] "GET /api/cvefor/cpe:2.3:a:apache:coyote_http_connector:1.1 HTTP/1.1" 200 -
```

Figura 9: Interfaz web de la API de CVE Search.

4.3.6 Interfaz de *SaaS Neer*

A continuación, se muestran una serie de capturas y la descripción de las distintas funciones de S*aaS Neer*. Las funciones con las que cuenta la herramienta *SaaS Neer* son: registro de usuarios, inicio de sesión, análisis de aplicaciones, reporte del escaneo de aplicaciones, análisis de dependencias, reporte del escaneo de dependencias. A continuación, se describe cada función.

Registro de usuarios e inicio de sesión

Al ingresar a la aplicación mediante un navegador web se podrá apreciar la pantalla de inicio de sesión como se muestra en la figura 10. En el caso de contar con un usuario y una contraseña, se procede a ingresarlos, de lo contrario, se hace clic en “Don’t have an account” para crear una nueva cuenta.

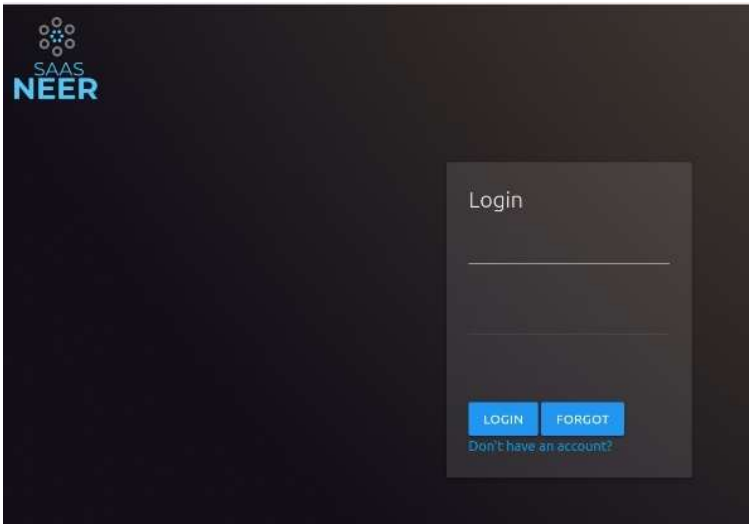


Figura 10: Vista inicio de sesión de *SaaS Neer*.

Para crear una nueva cuenta se procede a llenar los campos solicitados y se hace clic en “SignUP”, como se muestra en la figura 11.

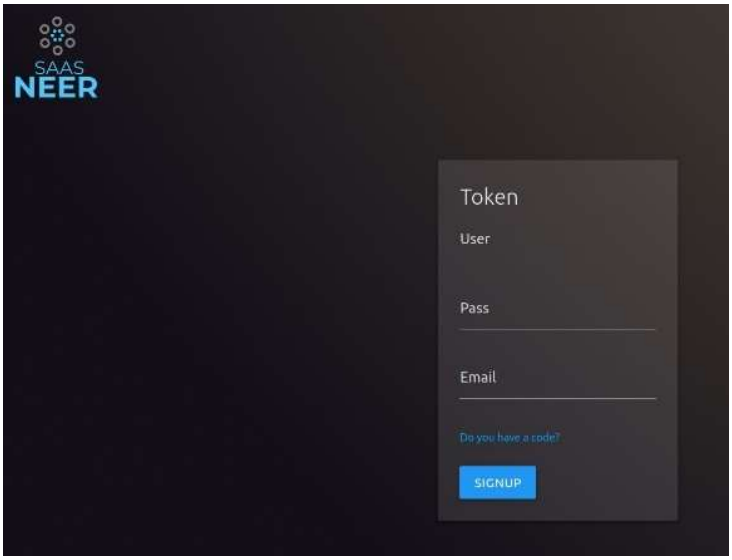


Figura 11: Vista de registro de nuevo usuario en *SaaS Neer*.

Una vez realizado este procedimiento, *Aws Cognito* generará la nueva cuenta y enviará un correo electrónico a la cuenta del nuevo usuario con un código de verificación. Por otro lado, *SaaS Neer* redirigirá a la página de verificación de cuenta, como se muestra en la figura 12.

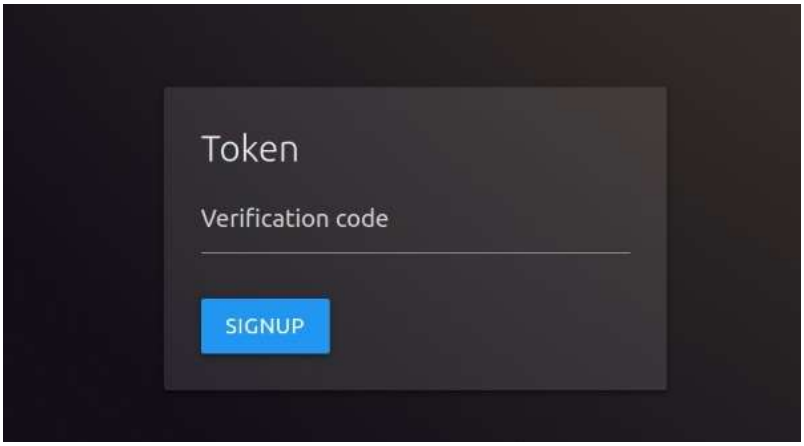


Figura 12: Vista de verificación de alta de usuario en *SaaS Neer*.

Una vez finalizado el registro, será posible ingresar a la página principal de la aplicación, como se muestra en la figura 13. Como se observa, todos los datos están en cero debido a que no se ha realizado ningún análisis anteriormente.

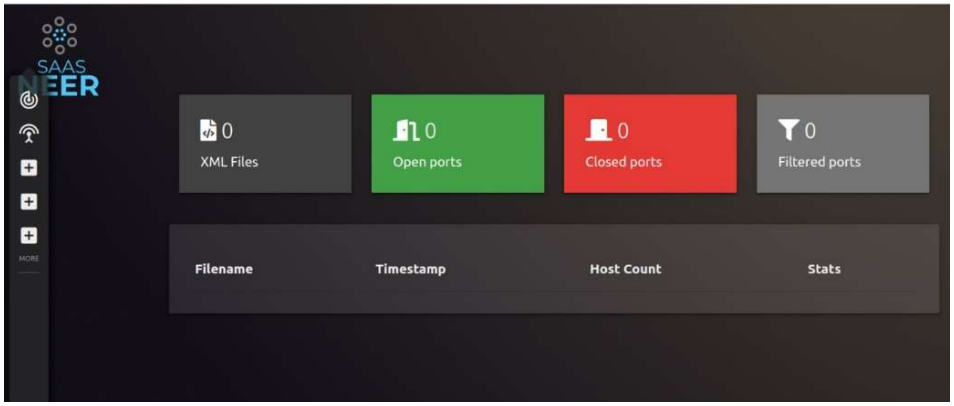


Figura 13: Vista principal de *SaaS Neer*.

Análisis de aplicaciones

Para realizar un análisis en busca de vulnerabilidades se debe seleccionar del menú la opción de "New Nmap Scan" como se muestra en la figura 14. *SaaS Neer* abrirá una vista emergente, en la cual nos solicita la dirección *IP* o nombre de *host* que se va a analizar.

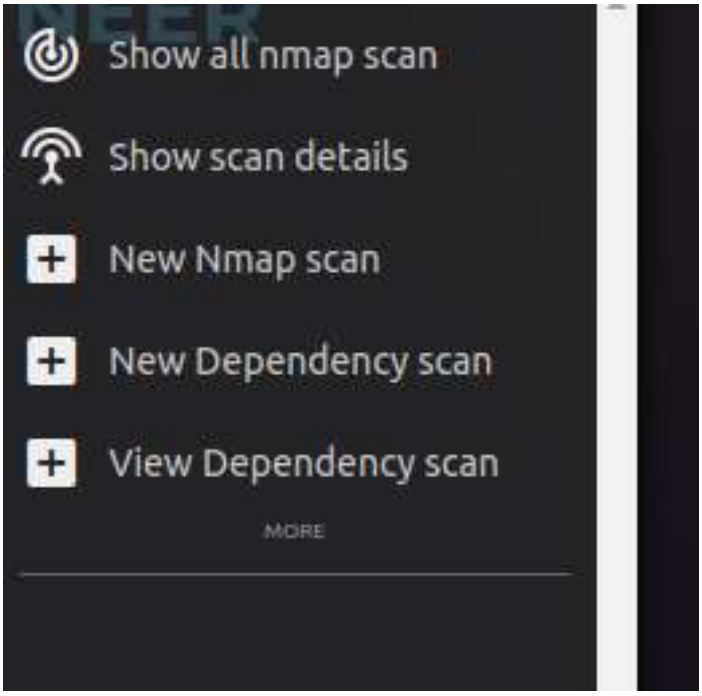


Figura 14: Menú lateral de SaaS Neer.

Para llevar a cabo este ejercicio se utilizó una máquina virtual, la cual contaba con un gran número de vulnerabilidades de todo tipo para comprobar el funcionamiento de *SaaS Neer*. Una vez que la máquina virtual se encontraba en ejecución, se procedió a la prueba. Se ingresa la dirección *IP* de la máquina y se procede al análisis, como se muestra en la figura 15.

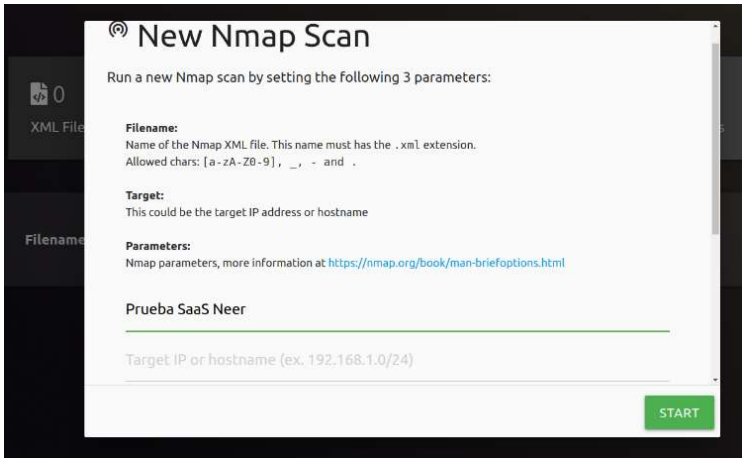


Figura 15: Vista para nuevo análisis de puertos y servicios en *SaaS Neer*.

Una vez llevado a cabo el análisis, se puede apreciar en la página principal el archivo generado con el nombre que le otorgamos al inicio. Además, es posible visualizar el número de puertos abiertos, número de puertos cerrados y número de puertos filtrados, como se muestra en la figura 16.

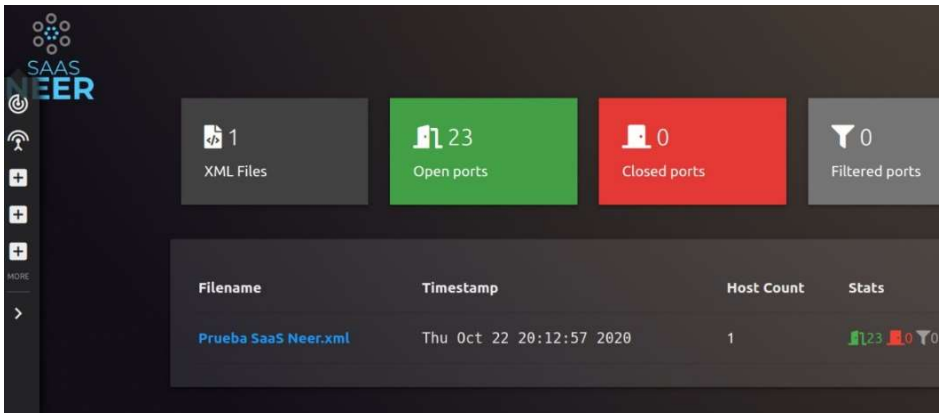


Figura 16: Vista principal de *SaaS Neer* después de un análisis de puertos.

Reporte de análisis de aplicaciones

El siguiente paso es seleccionar el archivo *XML* generado previamente. En la siguiente se sección se muestra información más detallada, como los servicios detectados corriendo bajo los puertos abiertos. Como se muestra en la figura 17, para llevar a cabo el análisis de vulnerabilidades se selecciona la opción "CHECK FOR CVE AND EXPLOITS". De esta forma, *SaaS Neer* mandará un *GET* a la *API* de *CVE Search* con el *CPE* 2.3 de los servicios encontrados en el análisis. Una vez finalizado este procedimiento, se puede generar un reporte en *PDF* como se muestra en la figura 18.

Figura 17: Vista de resumen de análisis de puertos y servicios.

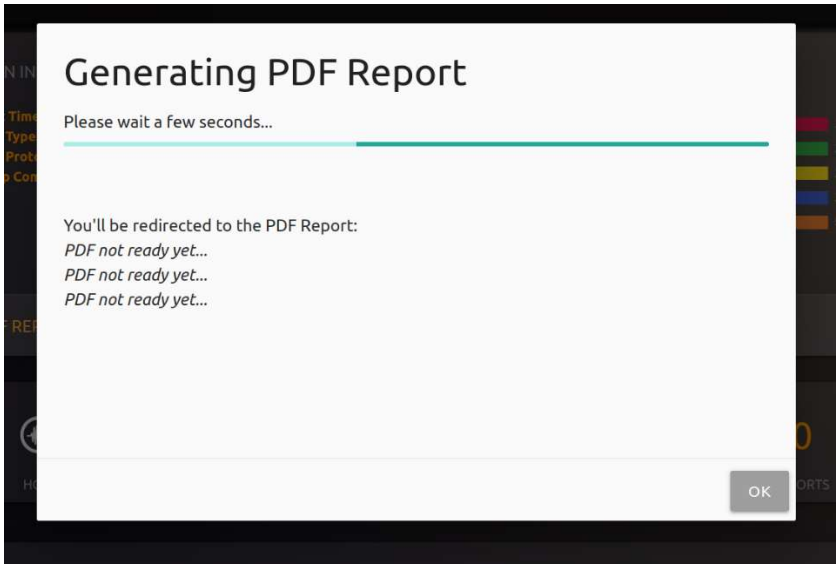


Figura 18: Generación de reporte en PDF en *SaaS Neer*.

En la primera sección del reporte se visualiza un resumen de los puertos y servicios encontrados, como se muestra en la figura 19.

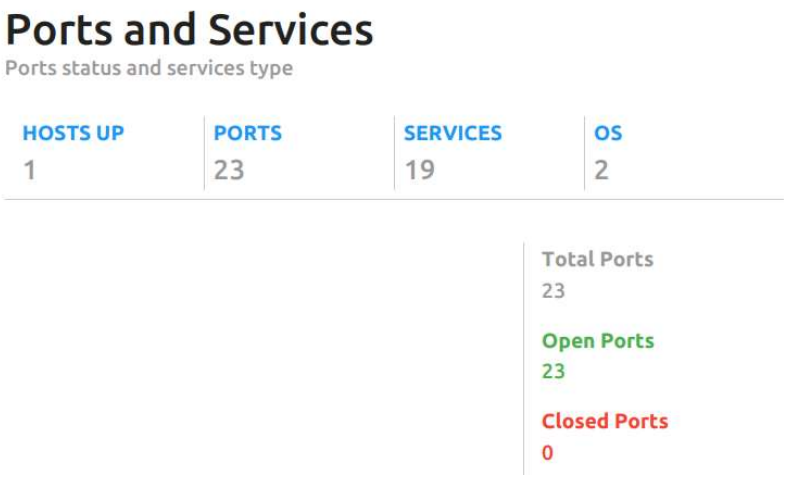


Figura 19: Resumen de puertos y servicios en reporte en formato PDF

Posteriormente, se enlista cada puerto con su respectivo servicio como se muestra en la figura 20.

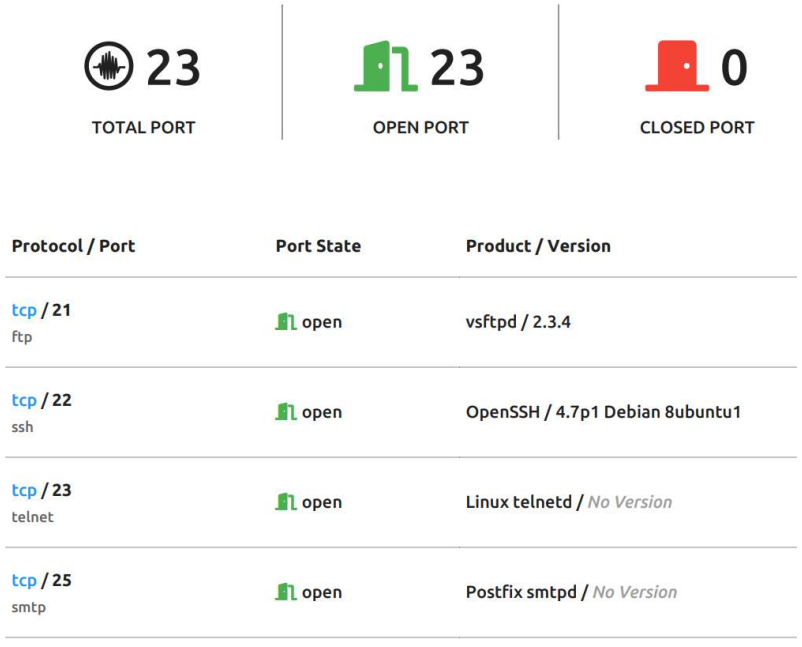


Figura 20: Lista de puertos y servicios en reporte PDF.

Por último, se puede apreciar la lista de *CVE* de cada servicio encontrado durante el análisis, como se muestra en la figura 21.

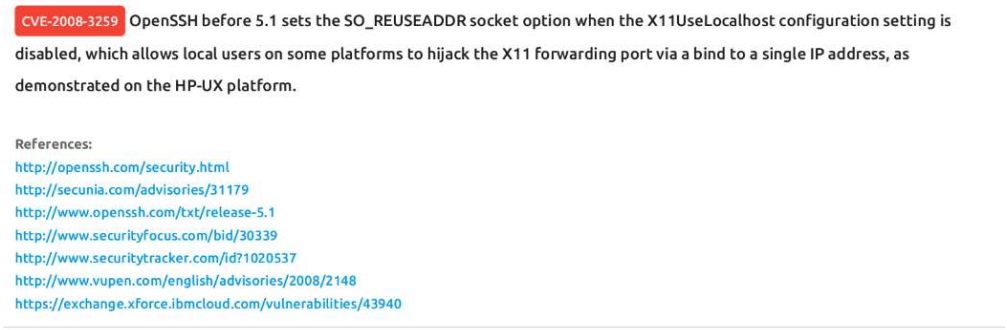


Figura 21: Muestra de un CVE en reporte PDF.

Análisis de dependencias

Por otra parte, para realizar el análisis a una dependencia de un desarrollo se elige la opción "New Dependency Scan" de la figura 22. Posteriormente, se elige la librería que se desea escanear, como se muestra en la figura 23.

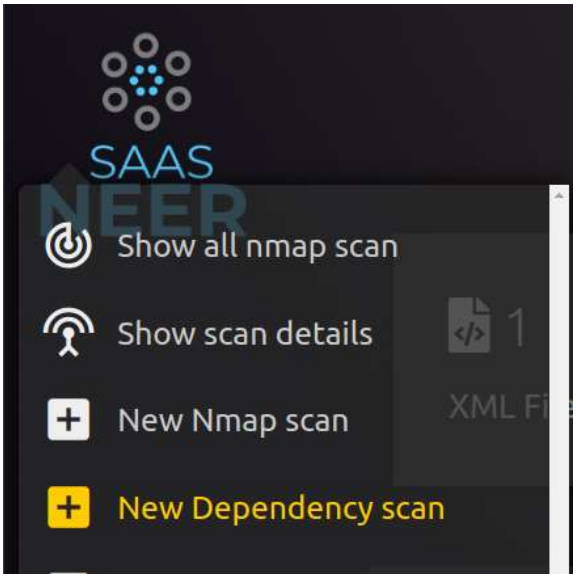


Figura 22: Escaneo de dependencias en *SaaS Neer*

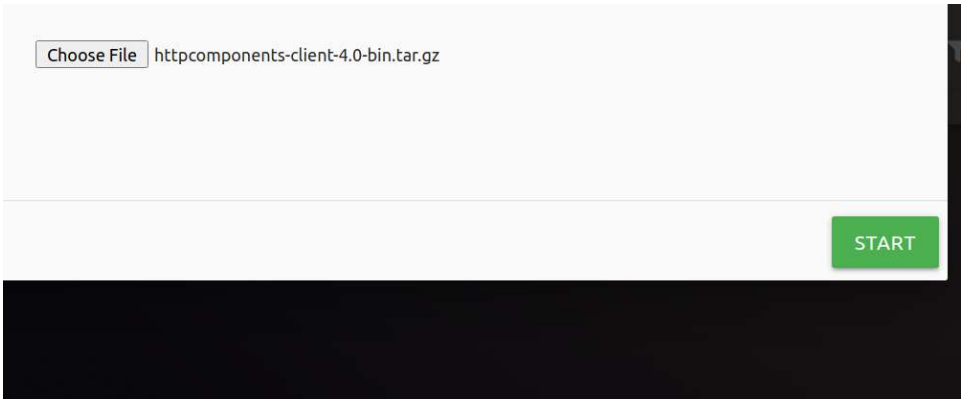


Figura 23: Escaneo de una librería en *SaaS Neer*

Reporte del análisis de dependencias

Al finalizar el análisis, se procede a elegir la opción "View Dependency Scan" para mostrar el reporte generado, como se muestra en la figura 24.

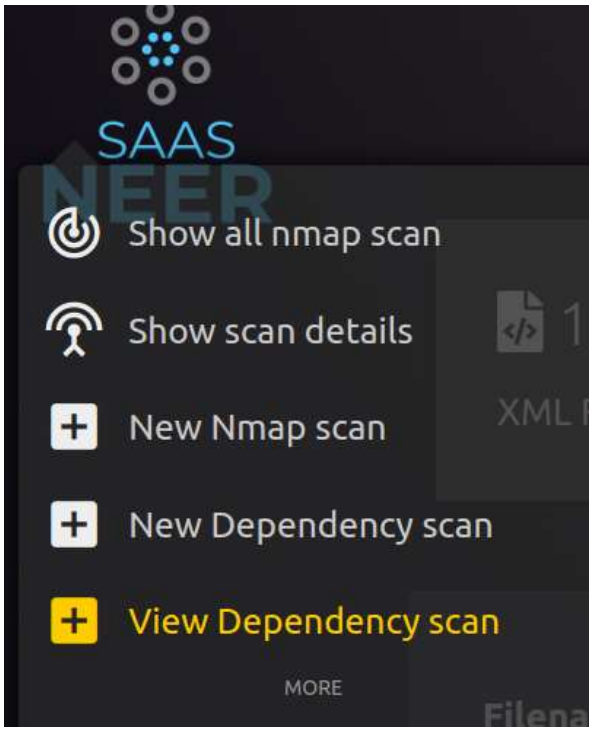


Figura 24: Menú para mostrar resultado de análisis de dependencias en *SaaS Neer*

En la primera sección del reporte se puede observar un resumen del análisis, el cual contiene la fecha y hora del reporte, el número de dependencias analizadas, número de vulnerabilidades encontradas y número de dependencias vulnerables, como se muestra en la figura 25.

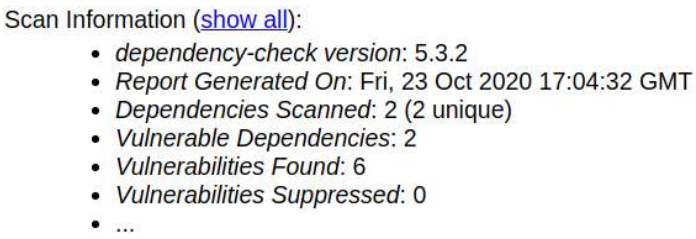


Figura 25: Resumen de análisis de dependencias en reporte.

En la siguiente sección se visualiza el *CPE* 2.3 de librería, el nivel de riesgo y el número de *CVE* encontrados, como se muestra en la figura 26.

Vulnerability IDs	Package	Highest Severity	CVE Count
cpe:2.3:a:apache:httpclient:4.0:*:*:*:*:*:*:*	pkg:maven/org.apache.httpcomponents/httpclient@4.0	MEDIUM	2

Figura 26: Conteo de vulnerabilidades por dependencia.

Por último, se muestra el detalle de cada CVE en el cual se encuentra el nombre, descripción y sus respectivas referencias, como se muestra en la figura 27.

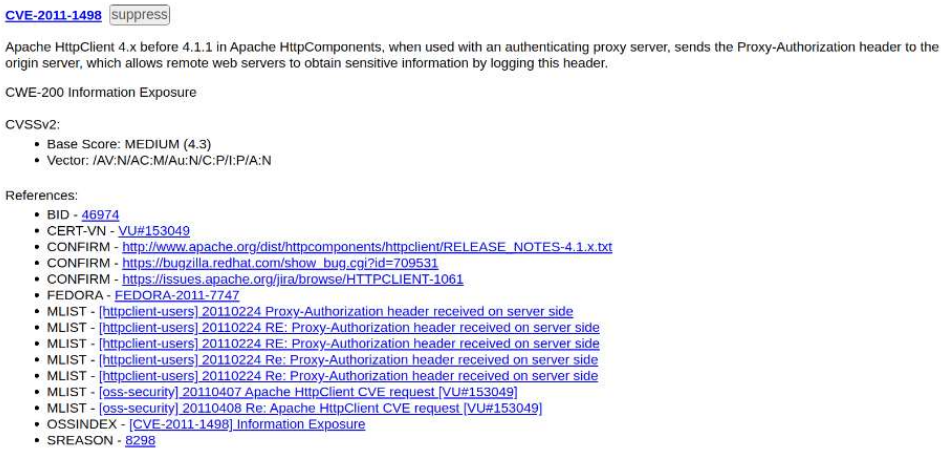


Figura 27: Muestra de CVE de una dependencia.

Si el desarrollador desea visualizar el marco de seguridad propuesto en este trabajo solo debe ingresar al menú y hacer clic en “View security framework” como se muestra en la figura 28.

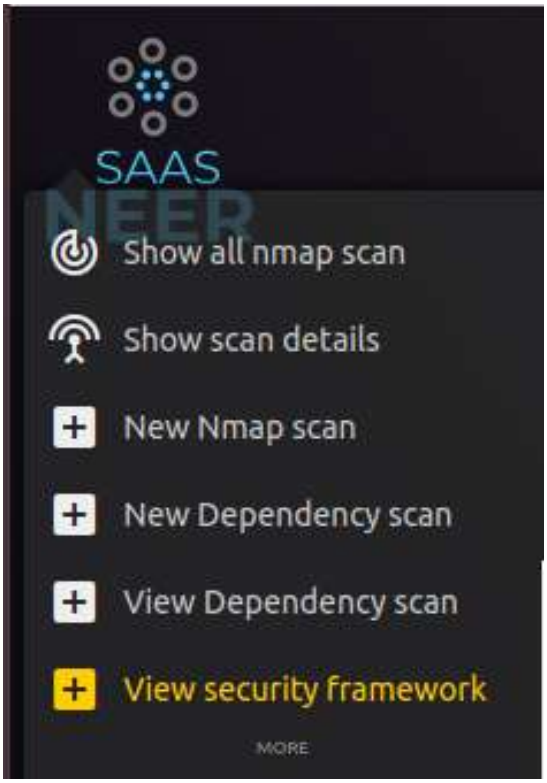


Figura 28: Menú para mostrar marco de seguridad propuesto en este trabajo.

Posteriormente, se visualizará el marco de seguridad con sus respectivas nomenclaturas. En la figura 29 se muestra una parte del marco.

Attack	R	PO	I	NR	STRIDE	Risk	Sec
ARP Spoofing	PC	M	L	L	S	Redirect to malicious host	Rel
Backdoor and debug options	D	H	H	H	S	Changes to the app	The
Broken authentication	D	H	H	H	T	Identity theft	Cor de s App Avo
Buffer overflow	D	H	H	H	E	Execution of malicious code	Ran
Code Injection	D	H	L	M	S	Loss of confidentiality	Act Det

Figura 29: Fragmento del marco de seguridad mostrado en *SaaS Neer.*

4.4 Implementación del servicio

En esta sección se describe el procedimiento realizado para la instalación e implementación de *SaaS Neer* en una instancia proporcionada por *Amazon AWS EC2*. Primeramente, se creó la imagen de una máquina virtual utilizando el asistente de configuración de *EC2*. El tipo de máquina que se eligió fue una *Linux,* debido al ahorro de recursos y porque aplica para los precios de *Free Tier*, lo que otorga un total de 720 horas de corrida sin costo de la instancia. Luego, se instaló el sistema operativo *Ubuntu Bionic*, por su alta compatibilidad con aplicaciones y por ser una versión de soporte extendido.

Al finalizar la configuración de la máquina virtual se procede con la generación de una llave *RSA* con extensión "pem", extensión de archivo utilizada por los archivos de certificado para almacenar y transportar claves criptográficas, la cual nos permitirá realizar la conexión a la instancia mediante un cliente SSH. El cliente elegido para esta tarea fue *PuTTY*, debido a las siguientes ventajas que proporciona:

- Es gratuito y de código abierto.
- Disponible para varias plataformas (Windows y Linux).
- Es una aplicación portable.
- Interfaz sencilla y manejable.

- Muy completo y ofrece una gran flexibilidad con multitud de opciones.
- Está en constante desarrollo.

Posteriormente, se ejecutó una secuencia de comandos para la instalación de la herramienta y sus respectivos componentes para su correcto funcionamiento. En la figura 30 se visualiza una captura de pantalla durante la implementación de la herramienta. Los comandos de instalación en un ambiente *Linux* son los siguientes:

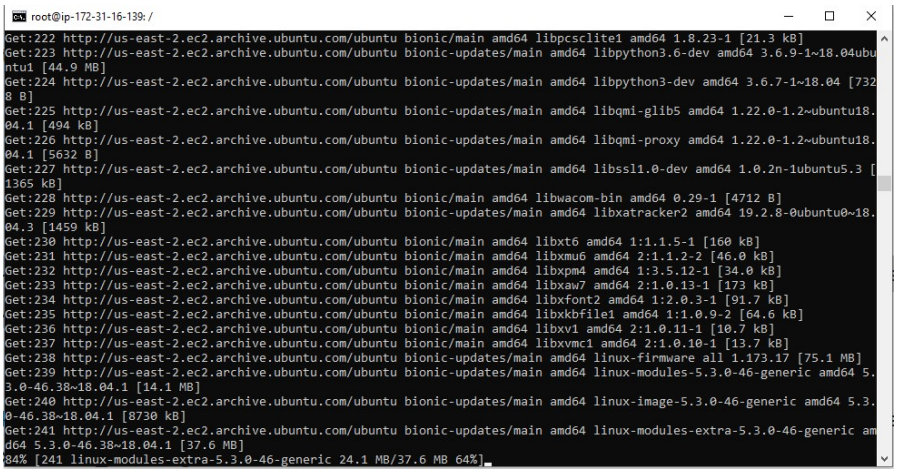


Figura 30: Implementación de la herramienta.

Actualización de sistema operativo e instalación de *Python*, *Nmap* y otras librerías.

apt-get update && apt-get install -y --allow-downgrades --allow-remove-essential --allow-change-held-packages \ python3 python3-pip curl wget git wkhtmltopdf libssl1.0-dev vim nmap tzdata

Instalación de dependencias

mkdir /opt/xml && mkdir /opt/notes && \ wget -P /opt/ https://github.com/wkhtmltopdf/wkhtmltopdf/releases/download/0.12.4/wkhtmltox-0.12.4_linux-generic-amd64.tar.xz && \ cd /opt/ && tar -xvf /opt/wkhtmltox-0.12.4_linux-generic-amd64.tar.xz

Instalación de *Django* y descarga de proyecto de *GitHub*

pip3 install Django requests xmltodict && \ cd /opt/ && django-admin startproject nmapdashboard && cd /opt/nmapdashboard && \ git clone https://Syscore1:%24DyM%40git180@github.com/Syscore1/SaaSneer.git nmapreport && \cd nmapreport && git checkout master

Ejecución de servicio

```
Python3 migrate.py runserver
```

4.5 Conclusiones de *SaaS Neer*

Al estar en un ambiente de nube, *Saas Neer* cuenta con una alta disponibilidad y fácil acceso. La herramienta fue diseñada para encontrar vulnerabilidades en aplicaciones en *SaaS* y en sus dependencias. Sin embargo, su uso se puede ampliar para encontrar vulnerabilidades en otros tipos de *host* como un *router*. Por lo tanto, gracias al empleo de esta herramienta, el desarrollador de aplicaciones en *SaaS* tiene conocimiento sobre los ataques que pueden ocurrir en sus implementaciones.

CAPITULO 5. Resultados y conclusiones

En este capítulo se describen los resultados y conclusiones obtenidos a partir del desarrollo de la revisión sistemática, estimación de riesgos, marco de seguridad y desarrollo de la herramienta *SaaS Neer*. Primeramente, se muestran los resultados que se pueden obtener con el uso de *SaaS Neer* y el marco de seguridad en conjunto. Posteriormente, se presentan las conclusiones de este trabajo. Además, se muestra como se satisface el impacto descrito en la sección 1.6. Por último, se menciona cómo fueron cumplidos los objetivos específicos de este desarrollo tecnológico.

5.1 Resultados

En esta sección se describe la forma en que el desarrollador de una aplicación en SaaS se beneficia de la utilización de *SaaS Neer* y la validación tanto del marco de referencia como de la herramienta. Primeramente, se mostrará un ejemplo llevado a cabo por el autor para demostrar la utilidad de la herramienta. Posteriormente, para validar el marco de referencia y la herramienta, se analizará una aplicación en desarrollo, llamada *Vet-O-Pet* y realizada en conjunto por el Ing. Cesar Javier Maldonado Flores, estudiante de la Maestría en Cómputo Aplicado de La Universidad Autónoma de Ciudad Juárez, y el autor. Por último, se presentan las conclusiones de este desarrollo tecnológico.

La herramienta presentada en este trabajo genera reportes en *PDF* detallados con la información necesaria para que el desarrollador pueda apoyarse de los mismos enlaces proporcionados por los reportes y con el marco de seguridad presentado en este trabajo para reducir el número de vulnerabilidades en sus desarrollos.

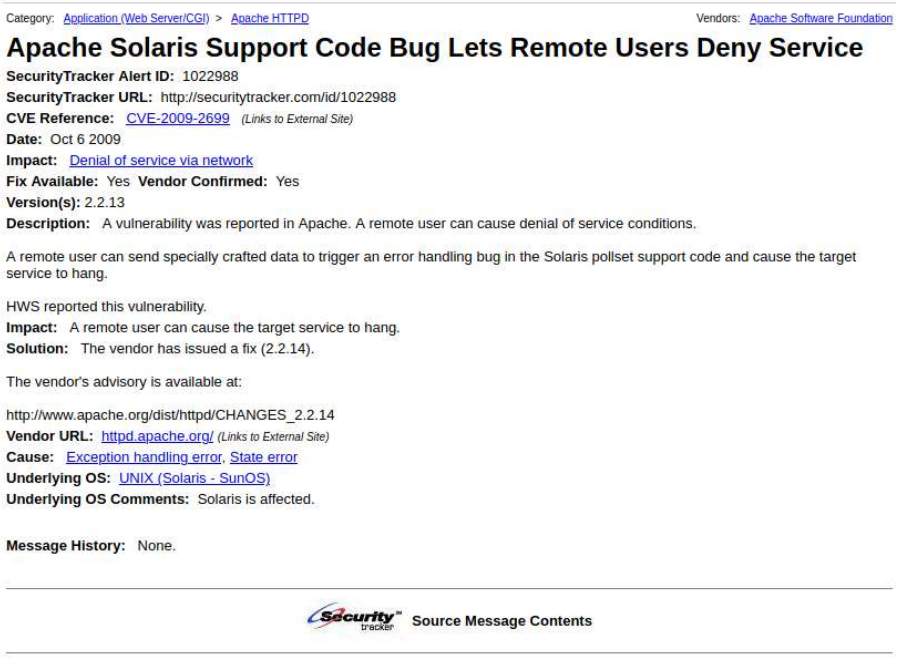
5.1.1 Resultados recaudados por el autor

En la sección 4.3.6 se muestra el procedimiento realizado por el autor para validar la herramienta en un ambiente de prueba. Una vez finalizados los análisis de vulnerabilidades tanto en la aplicación como en sus dependencias, el desarrollador encontrará la lista de CVE proporcionados por *SaaS Neer*. En la misma sección de cada CVE se proporciona la falla de seguridad que puede provocar. Por ejemplo, la CVE-2009-2699 que se muestra en la figura 31 puede provocar que un atacante ocasione una denegación de servicio. Además, se proporciona una serie de enlaces en los cuales se puede encontrar más información del CVE.

Figura 31: Muestra de CVE para ejemplo de uso de un desarrollador.

En este caso, como se muestra en la figura 32, al abrir el enlace http://securitytracker.com/id?1022988 se puede encontrar datos relevantes como:

- Si la vulnerabilidad ya fue solucionada.
- Si el proveedor de la aplicación ya confirmó la vulnerabilidad y su solución.
- Lo que puede provocar, en este caso es un ataque de denegación de servicio.
- La solución, en este caso se informa que a partir de la siguiente versión del servicio ya no se presenta esta vulnerabilidad.



Category: Application (Web Server/CGI) > Apache HTTPD Vendors: Apache Software Foundation

Apache Solaris Support Code Bug Lets Remote Users Deny Service

SecurityTracker Alert ID: 1022988
SecurityTracker URL: http://securitytracker.com/id/1022988
CVE Reference: CVE-2009-2699 *(Links to External Site)*
Date: Oct 6 2009
Impact: Denial of service via network
Fix Available: Yes **Vendor Confirmed:** Yes
Version(s): 2.2.13
Description: A vulnerability was reported in Apache. A remote user can cause denial of service conditions.

A remote user can send specially crafted data to trigger an error handling bug in the Solaris pollset support code and cause the target service to hang.

HWS reported this vulnerability.
Impact: A remote user can cause the target service to hang.
Solution: The vendor has issued a fix (2.2.14).

The vendor's advisory is available at:

http://www.apache.org/dist/httpd/CHANGES_2.2.14
Vendor URL: httpd.apache.org/ *(Links to External Site)*
Cause: Exception handling error, State error
Underlying OS: UNIX (Solaris - SunOS)
Underlying OS Comments: Solaris is affected.

Message History: None.

Security tracker **Source Message Contents**

Figura 32: Muestra de una referencia de CVE.

En el ejemplo presentado anteriormente, el desarrollador se entera que, al actualizar su servidor Apache a una versión más reciente evitará esta vulnerabilidad, por lo que ya no representará un problema de seguridad. Al conocer que esta vulnerabilidad puede ocasionar un ataque de denegación de servicio, el desarrollador puede consultar el marco de seguridad presentado en la sección 3.3 para conocer quién es el responsable de aplicar la medida de seguridad, la probabilidad de ocurrencia, el nivel de impacto, el nivel de riesgo, el tipo de amenaza STRIDE, el riesgo y las medidas adicionales que se pueden implementar para mitigar el riesgo.

Continuando con el ejemplo en el cual puede ocurrir un ataque de denegación de servicio por tener implementado un servidor apache vulnerable, además de las medidas recomendadas por las referencias generadas en los reportes, el desarrollador de la aplicación puede consultar el marco de seguridad para obtener más información del ataque de denegación de servicio. En este marco se encuentra la siguiente información:

- Responsable de realizar las medidas de seguridad. Proveedor y desarrollador son responsables de aplicar las medidas de seguridad.
- Probabilidad de ocurrencia. La probabilidad de ocurrencia es alta.
- Nivel de impacto. El nivel de impacto es alto.
- Nivel de riesgo. El nivel de riesgo es alto.

- Tipo de amenaza STRIDE. El tipo de amenaza STRIDE es de denegación de servicio.
- Riesgos. Los riesgos que se corren son el aumento de costo en los servicios.
- Medidas de seguridad. Las medidas de seguridad para mitigar el ataque son la implementación de una autenticación robusta, uso de IDS/IPS por parte del proveedor, análisis periódico, filtración de paquetes ICMP/SYN, filtración de direcciones IP privadas.

5.1.2 Validación

Para la validación del marco de referencia y de la herramienta el autor y el desarrollador mencionado en la sección 5.1 realizaron pruebas con el SaaS *Vet-O-Pet*. El SaaS se encuentra alojado en Amazon AWS con la dirección www.Vet-O.pet utilizando el servicio AWS S3, que otorga almacenamiento de objetos[3], para el alojamiento de la aplicación y AWS RDS, servicio de Amazon AWS que otorga bases de datos relacionales[4], para el almacenamiento de datos. En la figura 33 se muestra una captura de la aplicación alojada en AWS S3.

Figura 33: *Vet-O-Pet* un SaaS en desarrollo.

Primeramente, se realizó el análisis de la dirección del SaaS, www.Vet-O.pet en la herramienta *SaaS Neer*. El análisis mostró el puerto número 80 abierto, lo cual es normal debido a la naturaleza de la aplicación. La herramienta no encontró ninguna vulnerabilidad, sin embargo, para confirmar que este resultado es válido, se consultó una base de datos de vulnerabilidades en [136], en donde se puede verificar que hasta el momento no se han encontrado vulnerabilidades en el servicio S3 proporcionado por AWS, como se muestra en la figura 34. Sin embargo, al ser un SaaS en proceso de elaboración,

[3] https://aws.amazon.com/s3/ AWS S3
[4] https://aws.amazon.com/rds/ AWS RDS

los desarrolladores al estudiar el marco de referencia presentado en este trabajo conocen las medidas de seguridad que se deben implementar para mejorar la seguridad de *Vet-O-Pet* durante su desarrollo.

Figura 34: Actualmente no se han encontrado vulnerabilidades en AWS S3.

Para realizar el análisis de AWS RDS se ejecuta el mismo procedimiento. En la herramienta *SaaS Neer* ingresamos la dirección de acceso proporcionada por AWS RDS de *Vet-O-Pet*, en este caso es *database-.crusljhevue8.us-east-1.rds.amazonaws.com.* En esta ocasión, *SaaS Neer* detecta que la base de datos utiliza un gestor SQL Server 2012 SP2 en el puerto 1433 como se muestra en la figura 35.

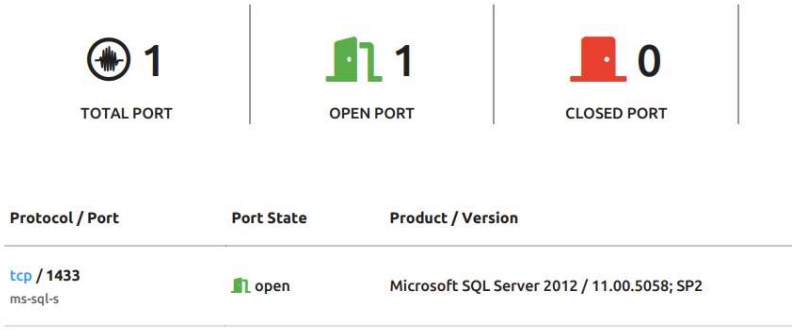


Figura 35: Puertos abiertos encontrados en el servicio AWS RDS de *Vet-O-Pet.*

Al abrir el reporte proporcionado por *SaaS Neer* se muestran las vulnerabilidades encontradas. Por ejemplo, en la figura 36 se muestra la vulnerabilidad "SQL Server Remote Code Execution Vulnerability" la cual puede provocar un ataque de ejecución de código remoto. Este ataque se puede

encontrar en el marco de referencia como *Code Injection* para conocer cómo afecta este tipo de ataques y como prevenirlos. Para obtener datos más específicos del ataque que ocurre en el servicio escaneado se consultan las referencias proporcionadas en el reporte de *SaaS Neer*. Es posible encontrar la fecha de publicación, descripción detallada, versiones afectadas, consecuencias del ataque, solución propuesta por el fabricante, entre otros.

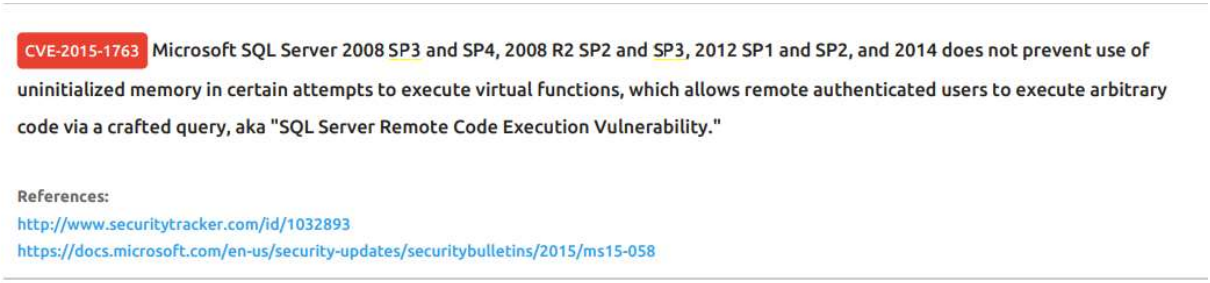


Figura 36: Una de las vulnerabilidades encontradas en el servicio AWS RDS de *Vet-O-Pet*

Con la validación presentada anteriormente se satisface el impacto presentado en la sección 1.6, debido a que: (1) el desarrollador al estudiar el marco de referencia conoce las medidas de seguridad necesarias para el diseño de sus aplicaciones. (2) Se logra la detección de vulnerabilidades aplicaciones SaaS mediante la herramienta *SaaS Neer*. (3) Mediante el estudio del marco de referencia el desarrollador es consciente de los posibles ataques que pueden ocurrir en su aplicación y con la utilización de la herramienta conoce de manera precisa cuales ataques pueden ocurrir. (4) Mediante el reporte proporcionado por *SaaS Neer* y el marco de referencia el desarrollador tiene la posibilidad de corregir los problemas de seguridad de sus aplicaciones.

Ahora el desarrollador cuenta con un marco de seguridad y una herramienta para reducir las vulnerabilidades en sus aplicaciones en SaaS. Así que, estas aplicaciones contarán con los atributos de seguridad como confidencialidad, disponibilidad e integridad. Además, existirá una certeza económica debido a que el servicio que proporciona la aplicación no estará fuera de línea por causa de un ataque. Por lo tanto, se considera que cualquier desarrollador de aplicaciones para SaaS debe contar con este marco de seguridad y el uso de *SaaS Neer.*

5.2 Conclusiones

A continuación, se describe cómo fueron cumplidos los objetivos establecidos en las secciones 1.3 y 1.4. Además, se presentan las conclusiones de este desarrollo tecnológico. Los objetivos específicos de este trabajo fueron abordados de la siguiente forma:

- Desarrollar una estimación de riegos con el fin de recomendar medidas de prevención. Primeramente, se elaboró una revisión sistemática, presentada en la sección 3.1, para contar con una base sólida de conocimiento que permita el desarrollo de la estimación de riesgos. Esto dio como resultado la publicación de un artículo como se menciona en la sección 3.4. De esta forma fue posible realizar la estimación de riesgos presentada en la sección 3.2.
- Desarrollar un marco de seguridad que los desarrolladores puedan utilizar como referencia para el diseño seguro de sus aplicaciones en SaaS. En la sección 3.3 se presenta el marco de seguridad elaborado para cumplir con este objetivo. Para esto, fue necesario realizar previamente la revisión sistemática y la estimación de riesgos debido a que se requería conocer los distintos ataques que pueden ocurrir en una aplicación en SaaS.
- Desarrollar una herramienta para la evaluación de seguridad en aplicaciones implementadas en SaaS. En el capítulo 4 se presenta *SaaS Neer*, la cual es una herramienta desarrollada por el autor para el análisis de seguridad de aplicaciones en SaaS y de dependencias. Esta herramienta tiene la capacidad de generar reportes indicando las vulnerabilidades, los posibles ataques con su respectivo impacto y las medidas que se pueden implementar para mejorar la seguridad.

Primeramente, para llevar a cabo este trabajo se realizó una revisión sistemática a partir de un total de 6723 artículos de los cuales 47 fueron seleccionados con base a un proceso de criterio de búsqueda. Este criterio de búsqueda utilizó el marco de SALSA permitiendo ahorrar tiempo debido a que no fue necesario la lectura completa de los 6723 artículos previamente encontrados. Además, SALSA permitió que solo se emplearan artículos de calidad relevantes para el proyecto. Según la documentación encontrada se conoció que una aplicación en SaaS puede sufrir un total de 30 ataques distintos. Siendo los más comunes "The Zombie attack", "Man-in-the-middle Attack", "Code Injection" and "Social Engineering". El resultado de la revisión sistemática fue publicado en la revista *"Journal of Computer Security"*. Posteriormente, con el conocimiento obtenido con la revisión sistemática y utilizando como base la guía del NIST se realizó una estimación de riesgos logrando establecer las probabilidades de ocurrencia, nivel de riesgo e impacto de cada ataque.

Lo descrito en el párrafo anterior permitió el desarrollo del marco de seguridad. Éste engloba toda la información recolectada por la revisión sistemática y la estimación de riesgos, lo que le permite al desarrollador conocer los datos más relevantes de un ataque. Estos datos abarcan las probabilidades

de ocurrencia de un ataque, descripción, impacto, nivel de riesgo, medidas de seguridad, responsable de implementar medidas de seguridad, entre otros.

Por último, en base al marco de seguridad se creó una herramienta capaz de detectar los servicios que está corriendo una aplicación y buscar sus vulnerabilidades. Además, de igual forma puede detectar vulnerabilidades en las dependencias en un proyecto. Una vez finalizados los análisis, la herramienta genera reportes detallados en los cuales se enlistan las vulnerabilidades detectadas, el ataque que puede ocurrir, su impacto y las medidas que se pueden implementar para mitigar el problema de seguridad.

En la sección 1.1 se menciona que una aplicación bajo una arquitectura SaaS, además de heredar los problemas de seguridad de una aplicación tradicional, se suma los ataques que pueden ocurrir por estar implementada en una arquitectura en SaaS. Incluso, algunos de estos ataques son más devastadores bajo esta arquitectura. Por lo tanto, el impacto de un ataque puede resultar en una pérdida mayor de información y de dinero para los desarrolladores.

El desarrollador, con la ayuda de la herramienta y el marco de seguridad, podrá mitigar los problemas de seguridad de sus aplicaciones. Así que, las probabilidades de ocurrencia y nivel de riesgo de un ataque se reducen. Por lo tanto, las medidas de seguridad que se presentan, tanto en los reportes de la herramienta como en el marco de seguridad, pueden prevenir la pérdida de información y de dinero a los desarrolladores de aplicaciones en SaaS.

Para un trabajo futuro se propone la implementación de inteligencia artificial para estimar cuándo puede ocurrir un cierto tipo de ataque. De esta forma, el desarrollador podrá enfocar su tiempo y esfuerzo para implementar las medidas de seguridad en un ataque en específico. Por último, la inteligencia artificial podría ayudar a reducir los problemas de seguridad de aplicación en SaaS.

Índice de figuras

Índice de tablas

Referencias

[1] K. Skemp MA, "Cloud computing," *Salem Press Encyclopedia of Science*. Salem Press, 2016.

[2] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *J. Netw. Comput. Appl.*, vol. 79, pp. 88–115, 2017.

[3] J. N. Goel and B. M. Mehtre, "Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology," *Procedia Comput. Sci.*, vol. 57, pp. 710–715, 2015.

[4] T. Halabi, M. Bellaiche, and A. Abusitta, "Online Allocation of Cloud Resources Based on Security Satisfaction," *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Trust, Security And Privacy In Computing And Communication*. IEEE, p. 379, 2018.

[5] J. Zhang, Q. Wu, R. Zheng, J. Zhu, M. Zhang, and R. Liu, "A Security Monitoring Method Based on Autonomic Computing for the Cloud Platform.," *J. Electr. Comput. Eng.*, pp. 1–9, Mar. 2018.

[6] K. Munir and S. Palaniappan, "Framework for secure cloud computing," *Adv. Int. J. Cloud Comput. Serv. Archit.*, vol. 3, no. 2, 2013.

[7] P. K. Sharma, J. H. Ryu, K. Y. Park, J. H. Park, and J. H. Park, "Li-Fi based on security cloud framework for future IT environment," *Human-centric Comput. Inf. Sci.*, vol. 8, no. 1, p. 23, 2018.

[8] K. Vijayakumar and C. Arun, "Analysis and selection of risk assessment frameworks for cloud based enterprise applications.," *Biomed. Res.*, vol. 28, pp. S129–S136, Nov. 2017.

[9] M. O. Alassafi, A. Alharthi, R. J. Walters, and G. B. Wills, "A framework for critical security factors that influence the decision of cloud adoption by Saudi government agencies," *TELEMATICS AND INFORMATICS*, vol. 34, no. 7. pp. 996–1010.

[10] M. Á. Díaz de León Guillén, V. Morales-Rocha, and L. F. Fernández Martínez, "A systematic review of security threats and countermeasures in SaaS," *J. Comput. Secur.*, no. Preprint, pp. 1–19.

[11] E. Loukis, M. Janssen, and I. Mintchev, "Determinants of software-as-a-service benefits and impact on firm performance," *Decis. Support Syst.*, vol. 117, pp. 38–47, 2019.

[12] W. Wang, "Data Security of SaaS Platform based on Blockchain and Decentralized Technology," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020, pp. 848–851.

[13] M. Ali, S. U. Khan, and A. V. Vasilakos, “Security in cloud computing: Opportunities and challenges,” *Inf. Sci. (Ny).*, vol. 305, pp. 357–383, Jun. 2015.

[14] G.-Y. Chan, F.-F. Chua, and C.-S. Lee, “Intrusion detection and prevention of web service attacks for software as a service: Fuzzy association rules vs fuzzy associative patterns,” *J. Intell. FUZZY Syst.*, vol. 31, no. 2, SI, pp. 749–764, 2016.

[15] M. Bishop, “What is computer security?,” *IEEE Secur. Priv.*, vol. 99, no. 1, pp. 67–69, 2003.

[16] J. Jang-Jaccard and S. Nepal, “A survey of emerging threats in cybersecurity,” *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 973–993, 2014.

[17] L. B. A. Rabai, M. Jouini, A. Ben Aissa, and A. Mili, “A cybersecurity model in cloud computing environments,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 25, no. 1, pp. 63–75, 2013.

[18] J. Andress, *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*. Syngress, 2014.

[19] N. Boudriga, *Security of mobile communications*. CRC Press, 2009.

[20] K. Ingham and S. Forrest, “A history and survey of network firewalls,” *Univ. New Mex. Tech. Rep*, 2002.

[21] J. E. Canavan, *Fundamentals of network security*. Artech House, 2001.

[22] A. Liska, *Building an intelligence-led security program*. Syngress, 2014.

[23] R. K. C. Chang, “Defending against flooding-based distributed denial-of-service attacks: a tutorial,” *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 42–51, 2002.

[24] D. Huang, A. Chowdhary, and S. Pisharody, *Software-Defined Networking and Security: From Theory to Practice*. CRC Press, 2018.

[25] E. Geier, “Intro to Next Generation Firewalls [OL].” 2011.

[26] S. Lewis and B. Anderson, *Mobile IP: Design Principles And Practice*. Scientific e-Resources, 2018.

[27] T. Nash, “An undirected attack against critical infrastructure,” *Tech. Report, US-CERT Control Syst. Secur. Cent.*, 2005.

[28] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education Upper Saddle River, NJ, USA, 2012.

[29] K. K. Mookhey and N. Burghate, *Linux: Security, Audit and Control Features*. ISACA, 2005.

[30] A. Young and M. Yung, “Cryptovirology: Extortion-based security threats and countermeasures,” in *Proceedings 1996 IEEE Symposium on Security and Privacy*, 1996,

pp. 129–140.

[31] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, “A taxonomy of computer program security flaws,” *ACM Comput. Surv.*, vol. 26, no. 3, pp. 211–254, 1994.

[32] G. Hoglund and J. Butler, *Rootkits: subverting the Windows kernel*. Addison-Wesley Professional, 2006.

[33] B. C. Brown, *How to Stop E-mail Spam, Spyware, Malware, Computer Viruses, and Hackers from Ruining Your Computer Or Network: The Complete Guide for Your Home and Work*. Atlantic Publishing Company, 2010.

[34] A. Henry, “The Difference Between Antivirus and Anti-Malware (and Which to Use),” in *Paper, GW Juette and LE Zeffanella,“Radio noise currents in short sections on bundle conductors (Presented Conference Paper style),” presented at*, 2013.

[35] D. M. Chess and S. R. White, “An undetectable computer virus,” in *Proceedings of Virus Bulletin Conference*, 2000, vol. 5, pp. 1–4.

[36] T. Fox-Brewster, “Netflix is dumping anti-virus, presages death of an industry,” *Forbes*, 2015.

[37] W. Wong and M. Stamp, “Hunting for metamorphic engines,” *J. Comput. Virol.*, vol. 2, no. 3, pp. 211–229, 2006.

[38] H. Kiem, N. T. Thuy, and T. M. N. Quang, “A machine learning approach to anti-virus system,” *training*, vol. 500, p. 1, 2004.

[39] M. Muhil, U. H. Krishna, R. K. Kumar, and E. A. M. Anita, “Securing Multi-cloud Using Secret Sharing Algorithm,” *Procedia Comput. Sci.*, vol. 50, pp. 421–426, 2015.

[40] J. B. Hong, A. Nhlabatsi, D. S. Kim, A. Hussein, N. Fetais, and K. M. Khan, “Systematic identification of threats in the cloud: A survey,” *Comput. Networks*, vol. 150, pp. 46–69, 2019.

[41] M. Hawedi, C. Talhi, and H. Boucheneb, “Security as a Service for Public Cloud Tenants(SaaS),” *Procedia Comput. Sci.*, vol. 130, pp. 1025–1030, 2018.

[42] N. vurukonda and B. T. Rao, “A Study on Data Storage Security Issues in Cloud Computing,” *Procedia Comput. Sci.*, vol. 92, pp. 128–135, 2016.

[43] S. K. A. Manoj and D. L. Bhaskari, “Cloud Forensics-A Framework for Investigating Cyber Attacks in Cloud Environment,” *Procedia Comput. Sci.*, vol. 85, pp. 149–154, 2016.

[44] M. Hamdaqa and L. Tahvildari, “Cloud computing uncovered: a research landscape,” in *Advances in Computers*, vol. 86, Elsevier, 2012, pp. 41–85.

[45] S. A. Aljawarneh and M. B. Yassein, “A Conceptual Security Framework for Cloud Computing Issues,” *Int. J. Intell. Inf. Technol.*, vol. 12, no. 2, pp. 12–24, 2016.

[46] T. Pai and P. S. Aithal, “A Review on Security Issues and Challenges in Cloud Computing Model of Resource Management,” 2017.

[47] G. Ramachandra, M. Iftikhar, and F. A. Khan, “A Comprehensive Survey on Security in Cloud Computing,” *Procedia Comput. Sci.*, vol. 110, pp. 465–472, 2017.

[48] P. Mell and T. Grance, *The NIST definition of cloud computing [electronic resource] / Peter Mell, Timothy Grance.* Gaithersburg, MD : Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, 2011, 2011.

[49] W. T. Tsai, X. Y. Bai, and Y. Huang, “Software-as-a-service (SaaS): Perspectives and challenges,” *Sci. China Inf. Sci.*, vol. 57, no. 5, pp. 1–15, 2014.

[50] T. Pflanzner and A. Kertesz, “A survey of IoT cloud providers,” in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 730–735.

[51] A. Koneru, N. B. N. S. R. Bhavani, K. P. Rao, G. S. Prakash, I. P. Kumar, and V. V. Kumar, “Sentiment Analysis on Top Five Cloud Service Providers in the Market,” in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 293–297.

[52] R. S. Ross, S. W. Katzke, and L. A. Johnson, *Minimum security requirements for federal information and information systems [electronic resource].* Gaithersburg, MD : Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, 2006, 2006.

[53] S. Alam, M. Muqeem, and S. A. Khan, “Review on security aspects for cloud architecture,” *Int. J. Electr. Comput. Eng.*, vol. 8, no. 5, pp. 3129–3139, 2018.

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

[55] P. Foreman, *Vulnerability management*. Auerbach Publications, 2009.

[56] W. A. Arbaugh, W. L. Fithen, and J. McHugh, “Windows of vulnerability: A case study analysis,” *Computer (Long. Beach. Calif).*, vol. 33, no. 12, pp. 52–59, 2000.

[57] G. Kostopoulos, *Cyberspace and Cybersecurity*. CRC Press, 2017.

[58] J. M. O. Candel, *Seguridad en aplicaciones Web Java*. .

[59] Z. Chen, Y. Zhang, and Z. Chen, “A categorization framework for common computer vulnerabilities and exposures,” *The Computer Journal*, 2010. .

[60] K. V Pradeep and V. Vijayakumar, “Survey on the Key Management for Securing the Cloud,” *Procedia Comput. Sci.*, vol. 50, pp. 115–121, 2015.

[61] J. A. Jones, “An Introduction to Factor Analysis of Information Risk (FAIR).,” *Norwich Univ. J. Inf. Assur.*, vol. 2, no. 1, pp. 1–66, Feb. 2006.

[62] T. Halabi and M. Bellaiche, “A broker-based framework for standardization and management of Cloud Security-SLAs,” *Comput. Secur.*, vol. 75, pp. 59–71, 2018.

[63] Z. M. Yusop and J. H. Abawajy, “Analysis of Insiders Attack Mitigation Strategies,” *Procedia - Soc. Behav. Sci.*, vol. 129, pp. 611–618, 2014.

[64] N. Khan and A. Al-Yasiri, “Identifying Cloud Security Threats to Strengthen Cloud Computing Adoption Framework,” *Procedia Comput. Sci.*, vol. 94, pp. 485–490, 2016.

[65] N. Subramanian and A. Jeyaraj, “Recent security challenges in cloud computing,” *Comput. Electr. Eng.*, vol. 71, pp. 28–42, Oct. 2018.

[66] V. Casola, A. De Benedictis, M. Rak, and E. Rios, “Security-by-design in Clouds: A Security-SLA Driven Methodology to Build Secure Cloud Applications,” *Procedia Comput. Sci.*, vol. 97, pp. 53–62, 2016.

[67] A. Aldaej, Ravichandran, M. G. Ahamad, and M. R. A. Dhivakar, “A study on state of the art in security and privacy issues on cloud computing,” *J. Eng. Appl. Sci.*, vol. 12, no. Specialissue11, pp. 9220–9226, 2017.

[68] P. R. Newswire, “Cloud Security Alliance Releases ‘The Treacherous Twelve’ Cloud Computing Top Threats in 2016,” *CSA-cloud-threats*. Y, 29-Feb-2016.

[69] Y. Asnar and N. Zannone, “Perceived Risk Assessment,” in *Proceedings of the 4th ACM Workshop on Quality of Protection*, 2008, pp. 59–64.

[70] Y. Huanchun, “A Study on the Risk Hierarchical Model and Risk Conduction Based on the Enterprise Complex Information System,” in *2009 International Forum on Information Technology and Applications*, 2009, vol. 3, pp. 595–599.

[71] R. Charanya, M. Aramudhan, K. Mohan, and S. Nithya, “Levels of security issues in cloud computing,” *Int. J. Eng. Technol.*, vol. 5, no. 2, pp. 1912–1920, 2013.

[72] N. Srinivasu, O. Sree Priyanka, M. Prudhvi, and G. Meghana, “Multilevel classification of security threats in cloud computing,” *Int. J. Eng. Technol.*, vol. 7, no. 1.5 Special Issue 5, pp. 253–257, 2018.

[73] P. R. Kumar, P. H. Raj, and P. Jelciana, “Exploring Data Security Issues and Solutions in Cloud Computing,” *Procedia Comput. Sci.*, vol. 125, pp. 691–697, 2018.

[74] S. Iqbal *et al.*, “On cloud security attacks: A taxonomy and intrusion detection and prevention as a service,” *J. Netw. Comput. Appl.*, vol. 74, pp. 98–120, 2016.

[75] M. Wu and Y. B. Moon, “Taxonomy of Cross-Domain Attacks on CyberManufacturing System,” *Procedia Comput. Sci.*, vol. 114, pp. 367–374, 2017.

[76] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, “Intrusion detection techniques in cloud environment: A survey,” *J. Netw. Comput. Appl.*, vol. 77, pp. 18–47, 2017.

[77] A. Philpott, “Identity theft – dodging the own-goals,” *Netw. Secur.*, vol. 2006, no. 1, pp. 11–13, 2006.

[78] P. Derbeko, S. Dolev, E. Gudes, and S. Sharma, “Security and privacy aspects in MapReduce on clouds: A survey,” *Comput. Sci. Rev.*, vol. 20, pp. 1–28, 2016.

[79] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, “A survey of intrusion detection techniques in Cloud,” *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, 2013.

[80] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, “DDoS attacks in cloud computing: Issues, taxonomy, and future directions,” *Comput. Commun.*, vol. 107, pp. 30–48, 2017.

[81] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl, *Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space*. 2011.

[82] M. A. Khan, “A survey of security issues for cloud computing,” *J. Netw. Comput. Appl.*, vol. 71, pp. 11–29, 2016.

[83] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen, “Detecting past and present intrusions through vulnerability-specific predicates,” in *ACM SIGOPS Operating Systems Review*, 2005, vol. 39, no. 5, pp. 91–104.

[84] D. Ye, T.-Y. Zhang, and G. Guo, “Stochastic coding detection scheme in cyber-physical systems against replay attack,” *Inf. Sci. (Ny).*, vol. 481, pp. 432–444, 2019.

[85] L. Coppolino, S. D’Antonio, G. Mazzeo, and L. Romano, “Cloud security: Emerging threats and current solutions,” *Comput. Electr. Eng.*, vol. 59, pp. 126–140, 2017.

[86] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, “Advanced social engineering attacks,” *J. Inf. Secur. Appl.*, vol. 22, pp. 113–122, 2015.

[87] A. Vasudeva and M. Sood, “Survey on sybil attack defense mechanisms in wireless ad hoc networks,” *J. Netw. Comput. Appl.*, vol. 120, pp. 78–118, 2018.

[88] G. Popov, B. K. Lyon, and B. Hollcroft, *Risk assessment: A practical guide to assessing operational risks*, 2016th ed. John Wiley & Sons, 2016.

[89] A. E.-S. O’Reilly and A. D. Pass, “Risk assessments in the mental health field,” *Salem Press Encyclopedia of Health*. Salem Press, 2018.

[90] O. Makarevich, I. Mashkina, and A. Sentsova, “The Method of the Information Security Risk Assessment in Cloud Computing Systems,” in *Proceedings of the 6th International Conference on Security of Information and Networks*, 2013, pp. 446–447.

[91] S. Sharma and B. Ram, “Causes of Human Errors in Early Risk Assesment in Software Project Management,” in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, 2016.

[92] D. Gritzalis, G. Iseppi, A. Mylonas, and V. Stavrou, “Exiting the Risk Assessment Maze: A Meta-Survey,” *ACM Comput. Surv.*, vol. 51, no. 1, 2018.

[93] W. M. Garrabrants, A. W. Ellis, L. J. Hoffman, and M. Kamel, “CERTS: a comparative evaluation method for risk management methodologies and tools,” in *[1990] Proceedings of the Sixth Annual Computer Security Applications Conference*, 1990, pp. 251–257.

[94] S. Liehtemtein, “Factors in the selection of a risk: assessment method,” *ComPuter&Seeuritv*, vol. 15, no. 5, pp. 401–422, 1998.

[95] L. Van Niekerk and L. Labuschagne, “The PECULIUM model: information security risk management for the south african SMME,” 2006.

[96] A. Syalim, Y. Hori, and K. Sakurai, “Comparison of risk analysis methods: Mehari, magerit, NIST800-30 and microsoft’s security management guide,” in *2009 International conference on availability, reliability and security*, 2009, pp. 726–731.

[97] “19th Americas Conference on Information Systems, AMCIS 2013, Volume 2,” in *19th Americas Conference on Information Systems, AMCIS 2013 - Hyperconnected World: Anything, Anywhere, Anytime*, 2013, vol. 3.

[98] M. Sajko, N. Hadjina, and D. Pešut, “Multi-criteria model for evaluation of information security risk assessment methods and tools,” in *The 33rd International Convention MIPRO*, 2010, pp. 1215–1220.

[99] S. Derakhshandeh and N. Mikaeilvand, “New framework for comparing information security risk assessment methodologies,” *Aust. J. Basic Appl. Sci.*, vol. 5, no. 9, pp. 160–166, 2011.

[100] S. Smojver, “Selection of information security risk management method using analytic hierarchy process (ahp),” in *Central European Conference on Information and Intelligent Systems*, 2011, p. 119.

[101] F. Macedo and M. M. Da Silva, “Comparative study of information security risk assessment models,” *Inst. Super. Técnico, UniversidadeTécnica Lisboa, Lisboa, Port.*, 2012.

[102] S. K. Pandey, “A comparative study of risk assessment methodologies for information systems,” *Bull. Electr. Eng. Informatics*, vol. 1, no. 2, pp. 111–122, 2012.

[103] K. V. D. Kiran, L. S. S. Reddy, and N. L. Haritha, “A compartive analysis on risk assessment information security models,” *Int. J. Comput. Appl.*, vol. 82, no. 9, 2013.

[104] L. Pan and A. Tomlinson, “A systematic review of information security risk assessment,”

Int. J. Saf. Secur. Eng., vol. 6, no. 2, pp. 270–281, 2016.

[105] G. Wangen, "Information security risk assessment: a method comparison," *Computer (Long. Beach. Calif).*, vol. 50, no. 4, pp. 52–61, 2017.

[106] M. Mylrea, S. N. G. Gourisetti, and A. Nicholls, "An introduction to buildings cybersecurity framework," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–7.

[107] P. Sirohi and A. Agarwal, "Cloud computing data storage security framework relating to data integrity, privacy and trust," in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, 2015, pp. 115–118.

[108] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Adaptable, model-driven security engineering for SaaS cloud-based applications," *Autom. Softw. Eng.*, vol. 21, no. 2, SI, pp. 187–224, Apr. 2014.

[109] M. Jouini and L. B. A. Rabai, "A security framework for secure cloud computing environments," in *Cloud security: Concepts, methodologies, tools, and applications*, IGI Global, 2019, pp. 249–263.

[110] W. M. Kang, S. Y. Moon, and J. H. Park, "An enhanced security framework for home appliances in smart home," *Human-centric Comput. Inf. Sci.*, vol. 7, no. 1, p. 6, 2017.

[111] S. Pirbhulal, O. W. Samuel, W. Wu, A. K. Sangaiah, and G. Li, "A joint resource-aware and medical data security framework for wearable healthcare systems," *Futur. Gener. Comput. Syst.*, vol. 95, pp. 382–391, 2019.

[112] I. F. del Amo, J. A. Erkoyuncu, R. Roy, R. Palmarini, and D. Onoufriou, "A systematic review of Augmented Reality content-related techniques for knowledge transfer in maintenance applications," *Comput. Ind.*, vol. 103, pp. 47–71, 2018.

[113] A. Chonka, Y. Xiang, W. Zhou, and A. Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks," *J. Netw. Comput. Appl.*, vol. 34, no. Advanced Topics in Cloud Computing, pp. 1097–1107, Jan. 2011.

[114] B. SeungHwan, J., Gelogo, Y.E., Park, "Next generation cloud computing issues and solutions," *Int. J. Control Autom.*, vol. 5, no. 1, pp. 63–70, 2012.

[115] W. Cilio, M. Linder, C. Porter, J. Di, D. R. Thompson, and S. C. Smith, "Mitigating power- and timing-based side-channel attacks using dual-spacer dual-rail delay-insensitive asynchronous logic," *Microelectronics J.*, vol. 44, pp. 258–269, Mar. 2013.

[116] S. Sundararajan, H. Narayanan, V. Pavithran, K. Vorungati, and K. Achuthan, "Preventing Insider Attacks in the Cloud BT - Advances in Computing and Communications," 2011, pp. 488–500.

[117] F. Rocha and M. Correia, “Lucy in the sky without diamonds: Stealing confidential data in the cloud,” *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*. p. 129, 2011.

[118] P. Calderon, *Nmap: Network Exploration and Security Auditing Cookbook*. Packt Publishing Ltd, 2017.

[119] L. Bell, M. Brunton-Spall, R. Smith, and J. Bird, *Agile application security: enabling security in a continuous delivery pipeline*. “ O’Reilly Media, Inc.,” 2017.

[120] N. El-Yabroudi, “Presentación de Flan Scan: Escáner ligero para detectar vulnerabilidades en la red de Cloudflare,” 2019. [Online]. Available: https://blog.cloudflare.com/es/introducing-flan-scan-es/. [Accessed: 10-Feb-2020].

[121] S. A. Aljawarneh, A. Alawneh, and R. Jaradat, “Cloud security engineering: Early stages of SDLC,” *Futur. Gener. Comput. Syst.*, vol. 74, pp. 385–392, 2017.

[122] Nagarjuna, C. C. K. Srinivas, and S. Sajida, “Security Techniques for Multi Tenancy Applications in Cloud,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 15, no. 8, pp. 80–83, Aug. 2015.

[123] M. Darwish, A. Ouda, and L. F. Capretz, “A cloud-based secure authentication (CSA) protocol suite for defense against Denial of Service (DoS) attacks,” *J. Inf. Secur. Appl.*, vol. 20, pp. 90–98, 2015.

[124] C. Saadi and H. Chaoui, “Cloud Computing Security Using IDS-AM-Clust, Honeyd, Honeywall and Honeycomb,” *Procedia Comput. Sci.*, vol. 85, pp. 433–442, 2016.

[125] M. Zineddine, “Vulnerabilities and mitigation techniques toning in the cloud. A cost and vulnerabilities coverage optimization approach using Cuckoo search algorithm with Lévy flights,” *Comput. Secur.*, vol. 48, pp. 1–18, Feb. 2015.

[126] J. T. F. T. Initiative, “Guide for conducting risk assessments,” National Institute of Standards and Technology, 2012.

[127] R. V. Rao and K. Selvamani, “Data Security Challenges and Its Solutions in Cloud Computing,” *Procedia Comput. Sci.*, vol. 48, pp. 204–209, 2015.

[128] S. Singh, Y.-S. Jeong, and J. H. Park, “A survey on cloud computing security: Issues, threats, and solutions,” *J. Netw. Comput. Appl.*, vol. 75, pp. 200–222, 2016.

[129] A. Pichan, M. Lazarescu, and S. T. Soh, “Cloud forensics: Technical challenges, solutions and comparative analysis,” *Digit. Investig.*, vol. 13, pp. 38–57, 2015.

[130] “WebMap,” 2020. [Online]. Available: https://github.com/SabyasachiRana/WebMap. [Accessed: 15-Feb-2020].

[131] F. A. Amo, L. M. Normand, and F. J. S. Pérez, *Introducción a la ingeniería del software: Modelos de desarrollo de programas*. Grupo Vanchri, 2014.

[132] S. Liawatimena *et al.*, "Django Web Framework Software Metrics Measurement Using Radon and Pylint," in *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, 2018, pp. 218–222.

[133] D. S. Foundation, "Django." [Online]. Available: https://www.djangoproject.com/. [Accessed: 05-Nov-2020].

[134] "FreeLogoDesign." [Online]. Available: https://www.freelogodesign.org/.

[135] V. Abramova and J. Bernardino, "NoSQL databases: MongoDB vs cassandra," in *Proceedings of the international C* conference on computer science and software engineering*, 2013, pp. 14–22.

[136] MITRE, "CVE Details." [Online]. Available: https://www.cvedetails.com/product/24793/Amazon-S3-Store.html?vendor_id=12126. [Accessed: 11-Jan-2021].