CAPÍTULO I
EL PROYECTO

I. CAPITULO I EL PROYECTO

I.1 Introducción

En la actualidad se presenta una variedad de delitos informáticos, en su mayoría son ataques a las aplicaciones web, utilizando las vulnerabilidades. La manipulación de la información puede provocar pérdidas económicas significativas brindando beneficios a los que realizan este tipo de actos.

El uso de las aplicaciones web es imprescindible en el ambiente informatizado donde se requiere información de manera rápida y en tiempo real, estas aplicaciones se adaptan en cualquier ámbito como comercio electrónico, bancos, centros educativos virtuales, redes sociales, blogs, foros, etc.

Estas aplicaciones web en su mayoría necesitan de la interacción entre el usuario y el navegador web, la información confidencial está viajando a través de estas aplicaciones, es ahí donde es propenso la integridad de la información, si no se proporciona niveles de seguridad, es por lo que está expuesto a diversos tipos de amenazas, inyección de código SQL, lo cual puede evitarse, la autenticación y gestión de sesiones.

La gran mayoría de los datos sensibles del mundo están almacenados en sistemas gestores de bases de datos comerciales tales como Oracle, Microsoft SQL Server, Microsoft Access entre otros. Atacar una base de datos es uno de los objetivos favoritos para los criminales informáticos.

Un estudio de 2003 encontró que hay un ataque cada 39 segundos en promedio en la web. Los nombres de usuario y las contraseñas inseguros brindan a los atacantes mayores posibilidades de éxito. Desafortunadamente, la web ha crecido tanto que estos estudios ya no son realistas.

Estos ataques son registrados y monitoreados por nuestro sistema de firewall, y el firewall de la aplicación web en el sitio web se asegura de que los ataques no tengan éxito. Un informe de 2019 encontró que las violaciones de seguridad habían aumentado en un 67% en los últimos cinco años.

Sin embargo, un estudio que compiló las infracciones divulgadas públicamente encontró que 18,5 millones de registros se vieron comprometidos todos los días en 2018. Esto equivale a 214 registros por segundo.

Uno de los principales problemas de seguridad al que se enfrentan las aplicaciones web son las vulnerabilidades que puedan presentar para explotar ataques de inyección en sus interfaces de interacción con el usuario.

Un ataque de inyección es un método de infiltración de código intruso que se vale de una vulnerabilidad informática en una aplicación web en el nivel de validación de las entradas de datos.

Por medio de estos ataques se puede realizar actos no autorizados como adquisición de cuentas de usuarios, acceso a información confidencial almacenada en la base de datos, cambiar configuraciones de usuario, modificar la información de una aplicación entre otros. Estos ataques son amenazas comunes en aplicaciones web y su explotación puede ocasionar serios problemas en un sistema informático es fundamental elaborar una solución al problema de esta índole.

La presente tesis cuenta con los siguientes componentes:

Componente 1

Estudio de vulnerabilidad en las aplicaciones web.

En el componente 1 se describirán las principales vulnerabilidades que afectan a las aplicaciones web.

Componente 2

Caso de estudio y aplicación práctica.

Realizar una aplicación práctica utilizando el prototipo de la aplicación web “Sistema de Gestión de Proyectos de Grado”.

I.2 Descripción del proyecto

I.2.1 Antecedentes

En los últimos años, la Word Wide Web (WWW) ha experimentado un crecimiento asombroso de muchas aplicaciones Web en línea que se han desarrollado para cumplir con ciertos propósitos. Día a día, todo el mundo está en constante comunicación con “la tecnología informática” para hacer uso de sus servicios, realizar sus actividades pero al parecer, desconocen el gran balance en que se encuentran las aplicaciones web en asuntos de seguridad, no todas las aplicaciones web que se visitan en línea son seguras, cuando cierto grupo de personas acceden a una aplicación web de interés, son objetivos primordiales por

parte de atacantes o hackers que constantemente están en la búsqueda de fallos para comprometer la información de los usuarios como la de una organización, empresa o institución en su totalidad.

La seguridad en los sistemas informáticos es de gran importancia que viene siendo objeto de estudio desde 1970. El concepto de seguridad hace referencia a las medidas y al control destinados a la protección contra la denegación de servicio y la ausencia de autorización (ya sea de forma accidental o intencionada) para descubrir, modificar o destruir datos.

En los últimos años hemos asistido a un vertiginoso aumento de la penetración de internet tanto particular como profesional en Bolivia 7,5 millones de bolivianos, un 65% de la población, son usuarios de Internet. Unas 500.000 personas se insertaron en la red de redes en el último año, lo que significa un incremento del 7,1% entre 2019 y 2020.

Este fenómeno que ya venía dándose con cierta antelación en otros países de Europa y del mundo supuso el aumento del uso y también del desarrollo de aplicaciones utilizadas a través de los entornos web, especialmente desde el año 2009 con la popularización de los servicios en la nube (cloud computing).

Este cambio de filosofía que ha llevado nuestras aplicaciones de entornos de escritorio (o de cliente-servidor en el ámbito empresarial) a servicios consumidos a través de un cliente web ha supuesto también un cambio en la filosofía de desarrollo y de aplicación de medidas de seguridad. El entorno relativamente controlado de un PC aislado donde la seguridad podía basarse en la protección entre los distintos usuarios del mismo, ha pasado a convertirse en un entorno totalmente global donde gran parte de nuestras aplicaciones y los datos que contienen son potencialmente accesibles desde cualquier parte del mundo.

Por desgracia, esta popularización de servicios no siempre ha ido de la mano de una mejora en las medidas de seguridad o en el enfoque que debían darse a los mismos. El gran número de desarrollos a realizar y la relativa facilidad para llevar a cabo los mismos ha hecho que en ocasiones no se hayan realizado de manera correcta o por los profesionales más adecuados, esto ha ido, entre otros factores, en detrimento de la seguridad de esos desarrollos.

Pese a lo anterior, en los últimos tiempos va ganando peso la idea de que, por sus especiales características, la seguridad es un punto crucial en el desarrollo de aplicaciones para entornos

web y se perciben mejoras en la misma, en parte por los siguientes motivos:

- Concienciación acerca de la importancia de la seguridad, tanto de los profesionales del desarrollo, como de los puestos directivos de las empresas, e incluso de los propios usuarios.
- Imposición dada por las legislaciones más recientes, tanto a nivel nacional como internacional.
- Mejora en las herramientas utilizadas, tanto en el desarrollo: lenguajes de programación, frameworks, etcétera, como de los productos de software ya desarrollados que hacen que las empresas y particulares que se decantan por estas opciones puedan tener unas mínimas condiciones de seguridad con un coste bajo.

Es así que se tiene las siguientes investigaciones realizadas con anterioridad referentes al tema de seguridad:

- HACKING WEB (ANÁLISIS DE ATAQUES SQL Inyección, XSS), (Nasser Segundo Shalabe Jimenez), 2019, Colombia, UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA.

  Este proyecto se basa en la seguridad de desarrollo de aplicativos web y gestores de bases de datos que ha sido afectados, por causa de errores de programación, configuraciones por defecto, lo cual ha generado el descubrimiento de vulnerabilidades para los atacantes y proporcionar ataques de impacto como SQL y XSS, que durante tiempo han sido el top 10 de OWASP.

- IMPLANTACIÓN DE SEGURIDAD EN ENTORNOS WEB (Juan Manuel Saura Martín), 2016, España, UNIVERSIDAD POLITÉCNICA DE CARTAGENA.

  El objetivo de este proyecto, es realizar un estudio exhaustivo de la implicación de desarrollar sistemas Web seguros. En particular se tendrá en cuenta los requisitos necesarios para adoptar un esquema de seguridad específico y las consideraciones pertinentes para el caso del lenguaje de programación PHP y un servidor de páginas

Web como Apache.

- SEGURIDAD EN APLICACIONES WEB: UNA VISIÓN PRÁCTICA (Luis Asensio Hidalgo), 2014, España, UNIVERSIDAD CARLOS III DE MADRID.

  La idea fundamental de este proyecto es situarse en un punto intermedio de todo este material, dando una visión más cercana y concreta de los puntos desarrollados en esa bibliografía.
  Se compone de un primer análisis sobre la legislación vigente en materia de desarrollo de aplicaciones, así como los estándares relativos a la seguridad de la información. Este análisis se centra en las implicaciones a nivel de seguridad y del desarrollo orientado a la web, no limitándose a reproducir los párrafos de la legislación aplicable o los estándares.

- REQUISITOS DE SEGURIDAD PARA APLICACIONES WEB (Yisel Niño Benitez), 2018, Cuba, UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

  El ritmo vertiginoso de los procesos de desarrollo de software actuales incrementa el riesgo de presentar vulnerabilidades en un sistema de software. El aseguramiento de la información y de los sistemas que la procesan es, por tanto, un objetivo crucial para las organizaciones. La gestión de la Seguridad Informática desde el inicio del desarrollo de software evita que los mecanismos de seguridad deban ser ajustados dentro de un diseño ya existente, lo que provocaría cambios que generalmente generan vulnerabilidades en el software, y un incremento de costo y el tiempo para solucionarlos. Sin embargo, un dilema común que encuentran los ingenieros de software durante el desarrollo de un Sistema es la falta de requerimientos de seguridad que permitan darle seguimiento desde etapas tempranas. En el trabajo se exponen varios elementos sobre el marco teórico referente a la Seguridad Informática y la Ingeniería de Requisitos. Además, se describe una propuesta preliminar de Requisitos No Funcionales de Seguridad para el desarrollo de aplicaciones web en la

Universidad de las Ciencias Informáticas con el objetivo de reducir las vulnerabilidades.

- SEGURIDAD DE APLICACIONES WEB: VULNERABILIDADES EN LOS CONTROLES DE ACCESO, 2017, Colombia, Grupo de Investigación en Seguridad de las Tecnologías de Información y Comunicaciones Facultad Regional Santa Fe - UNIVERSIDAD TECNOLÓGICA NACIONAL

  No hay dudas respecto de que la seguridad de las aplicaciones web es un tema de interés actual y cotidiano. Las complejas y sensibles funcionalidades de las actuales aplicaciones web han movido el perímetro de seguridad de las organizaciones, y una parte significativa del mismo ahora reside en las propias aplicaciones web. Y los privilegios de acceso a funcionalidades y datos ya no son uniformes y abiertos, sino que requieren de complejos esquemas., resultando esencial la fortaleza de los mecanismos de control de acceso. Las debilidades en los controles de acceso pueden surgir de diferentes fuentes: un diseño pobre de la aplicación hace muy difícil y hasta imposible el chequeo por accesos no autorizados, un simple descuido puede dejar desprotegidas funcionalidades críticas, o suposiciones erróneas acerca del comportamiento de los usuarios dejan a una aplicación web sin protección y pasible de un quiebre de seguridad. En muchos casos, detectar una brecha en los controles de acceso puede resultar trivial, pero en otros casos, puede ser muy difícil, quedando ocultos defectos sutiles dentro de la lógica de la aplicación, especialmente en aplicaciones complejas y de alta seguridad. La lección más importante es que cuando se chequea la robustez de los controles de acceso se debe mirar en todas direcciones, debiendo ser paciente y testear cada paso particular de todas las funcionalidades de la aplicación.

I.2.2 Justificaciones

I.2.2.1 Justificación tecnológica

El presente proyecto de investigación induce a la utilización de las políticas de seguridad ante las amenazas que existen en el ámbito de las aplicaciones web.

I.2.2.2 Justificación económica

Con la disminución de las vulnerabilidades y ataques a las aplicaciones web se permite el normal funcionamiento de las actividades en los servidores web, evitando perdidas de información y pérdidas económicas para empresas, instituciones o entidades bancarias que utilizan aplicaciones web.

I.2.2.3 Justificación social

Es necesario informar a la sociedad en general de la exposición de la información en las aplicaciones web, y cuán importante es tomar medidas de seguridad para no sufrir ataques que puedan infringir principios de seguridad de la información como la integridad o la confidencialidad, llevar de esta manera a una sociedad informatizada y automatizada.

I.2.3 Planteamiento del problema

Los ataques son amenazas constantes en las aplicaciones web, esto ocasiona que los sistemas de información sean más vulnerables, afectando la integridad y disponibilidad de la información, por tanto, es puesto en riesgo la continuidad de la actividad que realizan en la web.

Las causas de vulnerabilidad son debido a la configuración inadecuada en los proveedores de los servicios de alojamiento web.

El mal empleo de políticas de seguridad en las aplicaciones web durante la implementación puede generar vulnerabilidades que pueden ser aprovechados por los usuarios no autorizados, provocando un funcionamiento inadecuado de las aplicaciones web.

La falta de los sistemas de seguridad en las aplicaciones web, puede ocasionar un mal funcionamiento, infiltración de la información no propia, pérdida de información entre otros.

Los mecanismos de seguridad para los servidores y aplicaciones web pueden disminuir las vulnerabilidades ante los ataques, obteniendo resultados de una aplicación web funcional, menos expuesta a los ataques informáticos.

I.2.4 Objetivos

I.2.4.1 Objetivo general

Mejorar el nivel de seguridad de las aplicaciones web con la utilización del prototipo “Sistema de Gestión de Proyectos de Grado”.

I.2.4.2 Objetivos específicos

- Realizar un estudio de vulnerabilidades en las aplicaciones web.
- Realizar un caso de estudio y una aplicación práctica en el prototipo de la aplicación web “Sistema de Gestión de Proyectos de Grado”.

I.2.5 Sistema de marco lógico (SML)

I.2.5.1 Cuadro de involucrados

GRUPOS	INTERESES	PROBLEMAS PERCIBIDOS	RECURSOS Y MANDATOS
Desarrolladores	Adquirir conocimientos sobre los problemas de seguridad en las aplicaciones web.	Inmadurez en el conocimiento de los problemas de seguridad. Rápida evolución del perfil de amenazas. Restricciones de recursos y de tiempo. Mala configuración de los servidores web.	Concientización de los problemas de seguridad en las aplicaciones web. Investigación sobre los ataques y las defensas en las aplicaciones web. Contar en los equipos de diseño o de desarrollo con un especialista en seguridad con una dedicación completa. Actualizar el software de los servidores web, evitando así vulnerabilidades conocidas.
Empresas, entidades e instituciones	Adquirir una aplicación web con un alto nivel de seguridad. Interactuar mejor con sus clientes.	Diseño pobre de la aplicación web hace muy difícil y hasta imposible el chequeo por accesos no autorizados. Robo de datos sensibles. Bajo nivel de sensibilidad de los datos manejados en la aplicación web.	Contratar a un equipo de desarrolladores de aplicaciones web experimentados y que cuenten con un especialista en seguridad. Hacerse de los servicios de un profesional en el área de ethical hacking.
Usuarios	Utilizar aplicaciones web con un alto nivel de seguridad.	Falta de comunicación con el usuario. Falta de seguimiento de la percepción del usuario.	Concienciar a los usuarios de la correcta utilización de las aplicaciones web.

Tabla 1: ***Cuadro de Involucrados***

I.2.5.2 Árbol de problemas

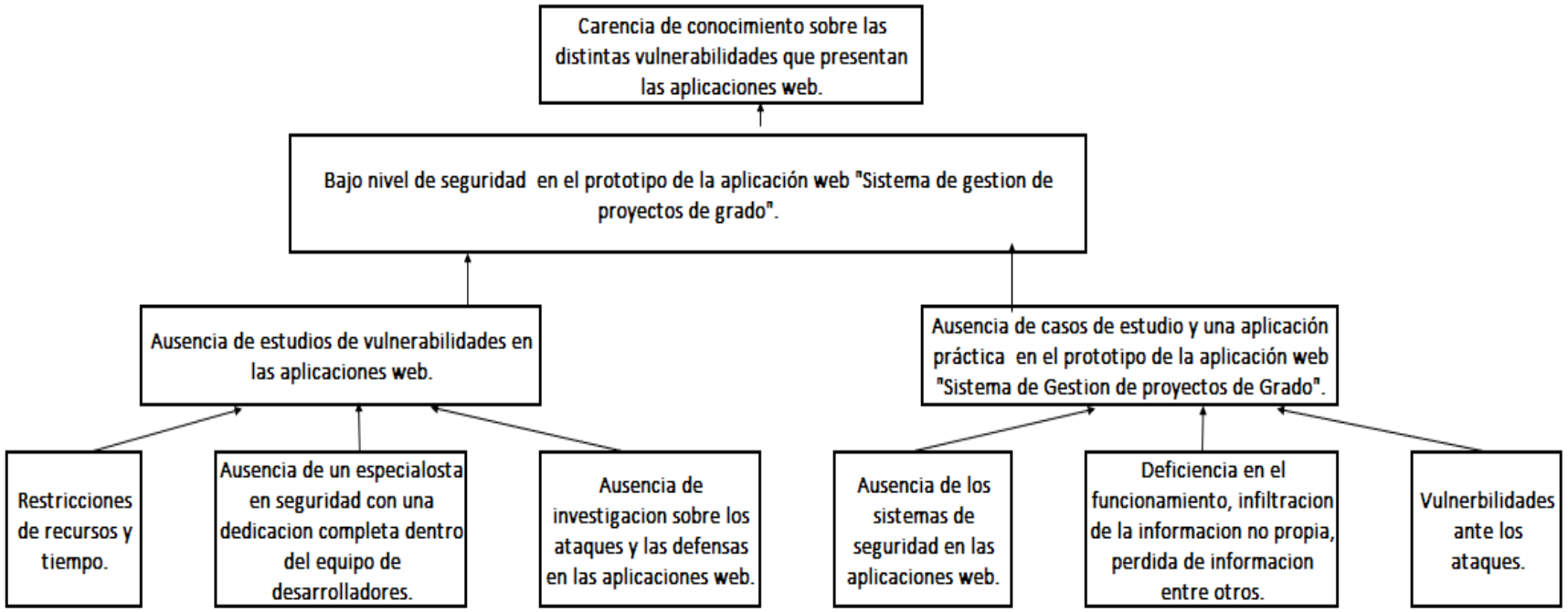


Figura 1. Árbol de Problemas

I.2.5.3 Árbol de objetivos

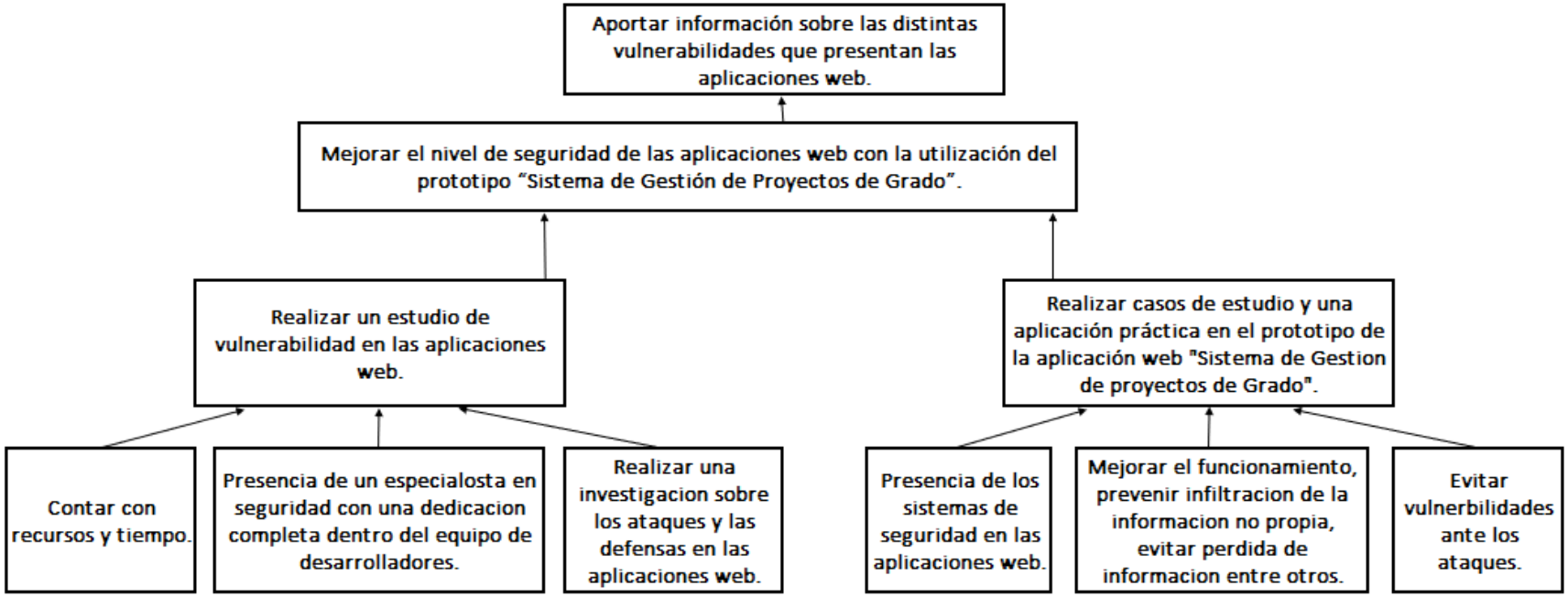


Figura 2. Árbol de Objetivos

I.2.5.4 Árbol de alternativas

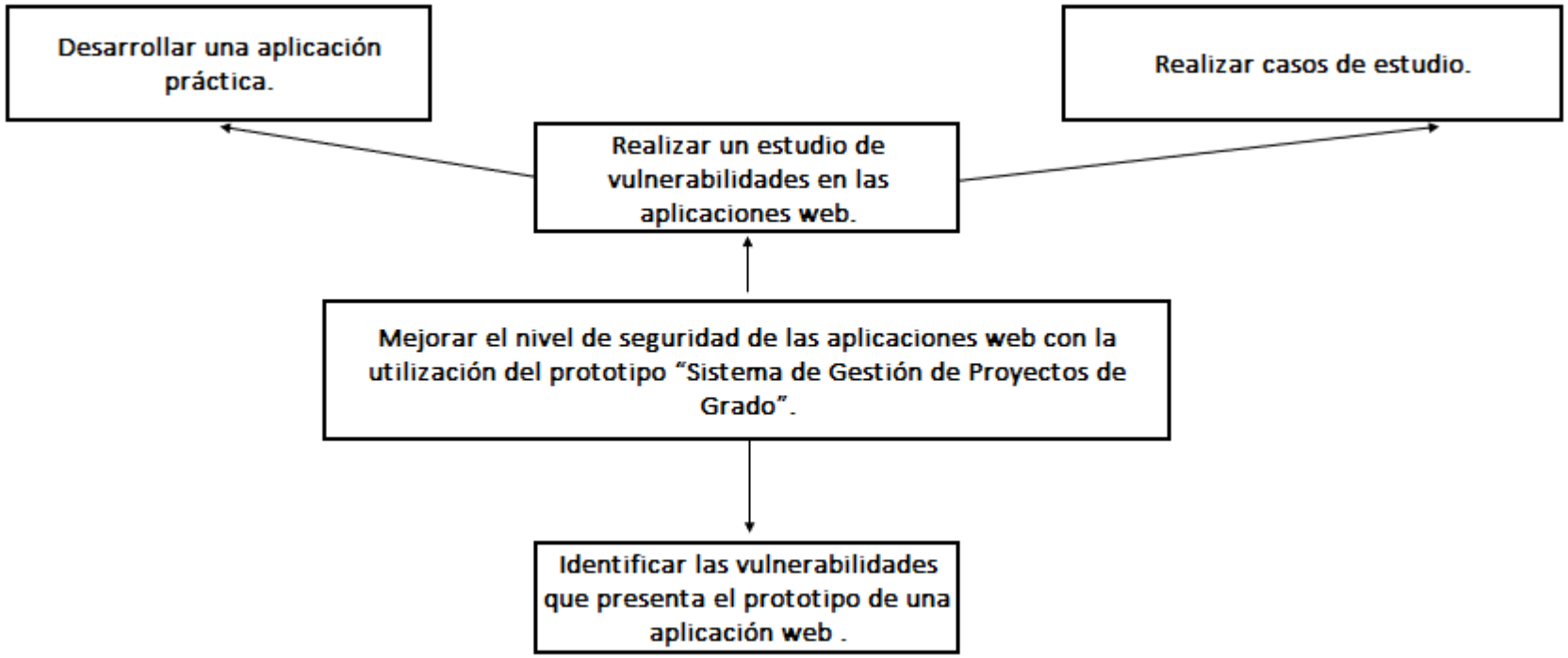


Figura 3. Árbol de alternativas

I.2.5.5 Matriz de marco lógico (MML)

Resumen narrativo	Indicadores	Fuentes de verificación	Supuestos
Fin: Aportar información sobre las distintas vulnerabilidades que presenta el prototipo de la aplicación web utilizada para el desarrollo de las pruebas.	Al finalizar el presente proyecto se proporcionará información detallada sobre las vulnerabilidades encontradas en el prototipo de la aplicación web para su posterior revisión durante el desarrollo de la aplicación web al cabo de un año como mínimo.	Informe técnico de las vulnerabilidades identificadas en el prototipo de la aplicación web.	El equipo de programadores tiene conocimiento sobre las distintas vulnerabilidades que presenta el prototipo de la aplicación web para reducir los riesgos de seguridad que presenta la aplicación web durante su desarrollo.
Objetivo General (Propósito): Mejorar el nivel de seguridad de las aplicaciones web con la utilización del prototipo "Sistema de Gestión de Proyectos de Grado".	Al finalizar el proyecto y tomando en cuenta el caso de estudio y la aplicación práctica que se realizó en el prototipo de la aplicación web, la seguridad del prototipo de la aplicación web ha sido mejorada en un 80%, se logró completar el proyecto, brindando información de primera mano sobre las distintas vulnerabilidades que presentan las aplicaciones web, disponible para su posterior revisión.	Informe de conformidad por parte del administrador de la aplicación. Revisión y visto bueno por parte de los tribunales y la docente de la materia de Taller III. Carta que certifica la realización del caso de estudio de vulnerabilidades y aplicación práctica sobre el prototipo de la aplicación web "Sistema de Gestión de Proyectos de Gestión de Proyectos de Grado".	Se ha desarrollado el estudio en las condiciones adecuadas y bajo consentimiento del administrador del prototipo de la aplicación web "Sistema de Gestión de Proyectos de Grado". Se desarrolla la aplicación web con menos vulnerabilidades. La aplicación web presenta un mejor nivel de seguridad.

Objetivos específicos (Componentes): • Realizar un estudio de vulnerabilidad en las aplicaciones web. • Realizar un caso de estudio y una aplicación práctica en el prototipo de la aplicación web "Sistema de Gestión de Proyectos de Grado".	**Componente 1** A los cuatro meses de iniciado el proyecto se cuenta con un estudio de las principales vulnerabilidades que presentan en las aplicaciones web "Sistema de gestión de proyectos de grado". **Componente 2** A los siete meses de iniciado el proyecto se realizó las pruebas de vulnerabilidades en el prototipo de la aplicación web "Sistema de gestión de proyectos de grado".	Carta de consentimiento para la realización de las pruebas de vulnerabilidades en el prototipo de la aplicación web "Sistema de Gestión de Proyectos de Grado".	Contar con la autorización del administrador del prototipo de la aplicación web "Sistema de Gestión de Proyectos de Grado" para realizar las pruebas de vulnerabilidades.
Actividades: **Componente I** **1.** Estudio de vulnerabilidad en las aplicaciones web. 3.1 Introducción 3.2 Propósito 3.3 Objetivo general 3.4 Objetivos específicos 3.5.- Web 3.5.1.-XSS (Cross Site Scripting) 3.5.2. CSRF (Cross Site Request Forgery) 3.5.3.-Inyección de Código (Code Injection)	**Componente 1** Estudio de vulnerabilidades de las aplicaciones web. **Componente 2** Caso de estudio y aplicación práctica. Bibliografía revisada.	Cumplimiento del cronograma del proyecto. Documentación de las principales vulnerabilidades en las aplicaciones web para su posterior aplicación práctica. Documentación de las pruebas de vulnerabilidades. Carta de satisfacción del administrador del prototipo de la aplicación web "Sistema de Gestión	Corrección de observaciones. Garantizar la existencia de la autorización del desarrollador del prototipo de la aplicación web "Sistema de gestión de proyectos de grado". Contar con el prototipo de la aplicación web "Sistema de gestión de proyectos de grado" para el desarrollo de las pruebas de vulnerabilidades.

3.5.4.-Buffer Overflow 3.6.-Bases de datos 3.6.1.-SQL Injection **Componente II** 4**.** Caso de estudio y aplicación práctica. 4.1. Introducción. 4.2. Propósito. 4.3 Objetivo general. 4.4 Objetivos específicos. 4.5. Arquitectura de las aplicaciones web mediante capas. 4.5.1. Capa de presentación visual. 4.5.2. Capa lógica de negocio. 4.5.3. Capa servicios de datos web. 4.5.4. Capa de alojamiento web. 4.6. Seguridad mediante capas en las aplicaciones web. 4.7. Prototipo. 4.7.1. Caso de estudio 1 4.7.1.1 Descripción 4.7.1.2 Prueba 4.7.1.3 Resultado 4.7.1.4. Recomendación 4.7.2. Caso de estudio 2 4.7.2.1 Descripción 4.7.2.2 Prueba 4.7.2.3 Resultado 4.7.2.4. Recomendación 4.7.3. Caso de estudio 3 4.7.3.1 Descripción		de Proyectos de Grado", expresando su satisfacción por el estudio de vulnerabilidades realizado a la aplicación durante el proceso de desarrollo. Revisión y visto bueno por parte del docente de la materia de Taller III.	Contar con el software y hardware para la instalación del sistema.

4.7.3.2 Prueba 4.7.3.3 Resultado 4.7.3.4. Recomendación 4.7.4. Caso de estudio 4 4.7.4.1 Descripción 4.7.4.2 Prueba 4.7.4.3 Resultado 4.7.4.4. Recomendación			

Tabla 2: ***Matriz de Marco Lógico***

I.2.6 Metodología de desarrollo

El método científico es el que se emplea para el desarrollo del presente proyecto de investigación, el método científico es un conjunto de pasos científicos bien estructurados que nos ayudan a formular, afirmar o corregir una teoría.

Se inicia con la Fase de Observación, donde el sujeto conocedor entra en contacto con el fenómeno, y sabe que algo lo induce a continuar investigando respecto al tema; el cual lleva a la Fase del Planteamiento de la hipótesis, que fundamentada en conocimientos previos y en los datos por recoger, podría ser demostrada; por ultimo tenemos la Fase de Comprobación, el cual depende del grado de generalidad y sistematicidad de la hipótesis.

Durante el desarrollo se complementa con el modelo 4+1 vistas para definir a la arquitectura por medio de 5 distintas vistas, tales como: vista de escenarios, vista lógica, vista de procesos, vista de desarrollo y vista física.

I.2.7 Resultados esperados

La identificación de vulnerabilidades que afectan a la seguridad en las aplicaciones web produce una disminución de las mismas, menos ataques a las aplicaciones web, permitiendo el normal funcionamiento, viendo los factores de vulnerabilidad, políticas de seguridad y otros aspectos de seguridad.

Componente I

Estudio de vulnerabilidades en las aplicaciones web, el resultado que se espera para el componente I es la correcta clasificación de los principales problemas de vulnerabilidad que se presentan durante el desarrollo de una aplicación web

Componente II

Caso de estudio y aplicación práctica, para el componente II del presente proyecto el resultado que se espera es la correcta identificación de vulnerabilidades en el prototipo de la aplicación web, “Sistema de Gestión de Proyectos de Grado”. Se espera también la

realización de una correcta aplicación práctica para facilitar la identificación de los problemas de seguridad que presenta el prototipo de la aplicación web.

I.2.8 Beneficiarios

I.2.8.1 Beneficiarios directos

Los beneficiarios directos del presente proyecto son los desarrolladores de aplicaciones web.

Al identificar las vulnerabilidades de las aplicaciones web se disminuyen las vulnerabilidades que afectan a la seguridad de las aplicaciones, es así que los desarrolladores se benefician, al conocer estas fallas de seguridad realizan una aplicación con un mejor nivel de seguridad.

I.2.8.2 Beneficiarios indirectos

Los beneficiarios indirectos del presente proyecto son: empresas, instituciones, entidades, entre otros.

Los usuarios de las aplicaciones web también son beneficiarios indirectos, con el desarrollo del presente proyecto se busca un mejor nivel de seguridad en las aplicaciones web, asegurando así la información confidencial y personal de estos beneficiarios.

I.3 Cronograma de actividades

Nº días	Fecha inicio	Fecha Finaliz.	M1 Abril	M2 Mayo	M3 Junio	M4 Julio	M5 Agosto	M6 Septiembre	M7 Octubre	M8 Noviembre	M9 Febrero	M10 Marzo	M12 Abril	M13 Mayo
15	1/4/2021	16/4/2021												
14	17/4/2021	31/04/2021												
14	1/5/2021	15/5/2021												
29	16/5/2021	14/6/2021												
44	19/6/2021	2/8/2021												
40	3/8/2021	25/9/2021												
50	1/10/2021	30/11/2021												
40	1/2/2022	31/3/2022												
15	1/4/2022	15/4/2022												
25	18/4/2022	15/5/2022												

Tabla 3: *Cronograma de Actividades*

CAPÍTULO II
MARCO TEÓRICO DEL PROYECTO

II. CAPÍTULO II MARCO TEÓRICO DEL PROYECTO

II.1 Introducción

El presente capítulo tiene por finalidad exponer los conceptos de forma general en los cuales se basa este trabajo, considerando principalmente los tres aspectos que implica la investigación, los cuales son riesgos en el lado del cliente, en el canal de comunicación y en el servidor. También hacemos mención a las técnicas y herramientas que pueden llevar a los riesgos o a las soluciones del tema abordado.

II.2 Seguridad informática

La Seguridad Informática se refiere a las características y condiciones de sistemas de procesamiento de datos y su almacenamiento, para garantizar su confidencialidad, integridad y disponibilidad. [VVA13]

Considerar aspectos de seguridad significa, conocer el peligro, clasificarlo y protegerse de los impactos o daños de la mejor manera posible. Esto significa que solamente cuando estamos conscientes de las potenciales amenazas, agresores y sus intenciones dañinas (directas o indirectas) en contra de nosotros, podemos tomar medidas de protección adecuadas, para que no se pierda o dañe nuestros recursos valiosos.

En este sentido, la Seguridad Informática sirve para la protección de la información, en contra de amenazas o peligros, para evitar daños y para minimizar riesgos, relacionados con ella. [ALV05]

La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. Sin embargo, no existe una técnica que permita asegurar la seguridad de un sistema al 100%.

Un sistema informático puede ser protegido desde un punto de vista lógico o físico. Por otra parte, las amenazas pueden proceder desde programas dañinos que se instalan en la computadora del usuario o llegar por vía remota. Entre las herramientas más usuales de seguridad informática, se encuentran los programas antivirus, los cortafuegos o firewalls, la encriptación de la información y el uso de contraseñas.

Un sistema seguro debe ser íntegro con información modificable sólo por las personas autorizadas, confidencial que los datos tienen que ser legibles únicamente para los usuarios autorizados, irrefutable que el usuario no debe poder negar las acciones que realizó y tener

buena disponibilidad implica que debe ser estable. [VVA13]

De todas formas, como en la mayoría de los ámbitos de la seguridad, lo esencial sigue siendo la capacitación de los usuarios. Una persona que conoce cómo protegerse de las amenazas sabrá utilizar sus recursos de la mejor manera posible para evitar ataques o accidentes. Es decir, la seguridad informática busca garantizar que los recursos de un sistema de información sean utilizados tal como una organización o un usuario lo ha establecido, sin intromisiones. [VVA13]

II.2.1 Seguridad física

La seguridad física consiste en la aplicación de barreras físicas, y procedimientos de control como medidas de prevención y contra medidas ante amenazas a los recursos y la información confidencial, se refiere a los controles y mecanismos de seguridad dentro y alrededor de la obligación física de los sistemas informáticos, así como los medios de acceso remoto al y desde el mismo, implementados para proteger el hardware y medios de almacenamiento de datos. Cada sistema es único, por lo tanto, la política de seguridad a implementar no será única es por ello que se recomienda pautas de aplicación general y no procedimientos específicos.

La seguridad física está enfocada a cubrir las amenazas ocasionadas tanto por el hombre como por la naturaleza del medio físico donde se encuentra ubicado el centro. Las principales amenazas que se ven en la seguridad física son amenazas ocasionadas por el hombre como robos destrucción de la información, disturbios, sabotajes internos y externos, incendios accidentales, tormentas e inundaciones. [DRAG11]

II.2.2 Seguridad lógica

Consiste en la aplicación de las barreras y procedimientos que resguarden el acceso a los datos y solo se permita acceder a ellos al personal autorizado. [DRAG11]

Por lo tanto, se refiere a la seguridad en el uso de software y los sistemas, la protección de los datos, procesos y programas, así como la del acceso ordenado y autorizado de los usuarios a la información. La “seguridad lógica” involucra todas aquellas medidas establecidas por la administración -usuarios y administradores de recursos de tecnología de información- para minimizar los riesgos de seguridad asociados con sus operaciones cotidianas llevadas a cabo

utilizando la tecnología de información. Los principales objetivos que persigue la seguridad lógica son:

- Restringir el acceso a los programas y archivos.
- Asegurar que se estén utilizando los datos, archivos y programas correctos en y por el procedimiento correcto.

La seguridad lógica se encarga de los controles de acceso que están diseñados para salvaguardar la integridad de la información almacenada de una computadora, así como de controlar el mal uso de la información. [DRAG11]

La seguridad lógica se encarga de controlar y salvaguardar la información generada por los sistemas, por el software de desarrollo y por los programas en aplicación. [DRAG11]

Identifica individualmente a cada usuario y sus actividades en el sistema, y restringe el acceso a datos, a los programas de uso general, de uso específico, de las redes y terminales.

- La falta de seguridad lógica o su violación puede traer las siguientes consecuencias a la organización.
- Cambio de los datos antes o cuando se le da entrada a la computadora.
- Copias de programas y /o información.
- Código oculto en un programa.
- Entrada de virus.

La seguridad lógica puede evitar una afectación de perdida de registros, y ayuda a conocer el momento en que se produce un cambio o fraude en los sistemas. [DRAG11]

Un método eficaz para proteger sistemas de computación es el software de control de acceso. Los paquetes de control de acceso protegen contra el acceso no autorizado, pues piden al usuario una contraseña antes de permitirle el acceso a información confidencial. Sin embargo, los paquetes de control de acceso basados en componentes pueden ser eludidos por delincuentes sofisticados en computación, por lo que no es conveniente depender de esos paquetes por si solos para tener una seguridad adecuada. [DRAG11]

II.3 Seguridad de la información

La seguridad de la información son todas las medidas preventivas que permitan resguardar y proteger la información, manteniendo la confidencialidad, la disponibilidad e integridad. [JL08]

El concepto de seguridad de la información no debe ser confundido con la seguridad informática, ya que este último solo encarga de la seguridad en el medio de la informática y que la información se puede encontrar en diferentes medios o formas.

II.3.1 ISO 27001

ISO 27001 es una norma internacional emitida por la Organización Internacional de Normalización (ISO) y describe cómo gestionar la seguridad de la información en una empresa. La revisión más reciente de esta norma fue publicada en 2013 y ahora su nombre completo es ISO/IEC 27001:2013. La primera revisión se publicó en 2005 y fue desarrollada en base a la norma británica BS 7799-2. [KOOLE15]

ISO 27001 puede ser implementada en cualquier tipo de organización, con o sin fines de lucro, privada o pública, pequeña o grande. Está redactada por los mejores especialistas del mundo en el tema y proporciona una metodología para implementar la gestión de la seguridad de la información en una organización. También permite que una empresa sea certificada; esto significa que una entidad de certificación independiente confirma que la seguridad de la información ha sido implementada en esa organización en cumplimiento con la norma ISO 27001. [KOOLE15]

ISO 27001 se ha convertido en la principal norma a nivel mundial para la seguridad de la información y muchas empresas han certificado su cumplimiento; aquí se puede ver en la figura 2-1 la cantidad de certificados emitidos en los últimos años en Latinoamérica:

Figura 4. Cantidad de certificados emitidos en Latinoamérica

Fuente: Encuesta ISO sobre certificaciones de la norma para sistemas de gestión.

Establece un sistema gerencial que permite minimizar el riesgo y proteger la información de amenazas externas o internas. No está orientada a despliegues tecnológicos o de infraestructura, sino a aspectos netamente organizativos, es decir, la frase que podría definir su propósito es organizar la seguridad de la información.

Actualmente es el único estándar aceptado internacionalmente para la administración de la Seguridad de la Información y se aplica a todo tipo de organizaciones, independientemente de su tamaño o actividad. [KOOLE15]

Su objetivo principal es el establecimiento e implementación de un Sistema de Gestión de la Seguridad de la Información.

La seguridad de la información debe preservar la:

- Confidencialidad: Aseguramiento de que la información es accesible sólo para aquellos autorizados a tener acceso.
- Integridad: Garantía de la exactitud y completitud de la información y de los métodos de su procesamiento.
- Disponibilidad: Aseguramiento de que los usuarios autorizados tienen acceso cuando lo requieran a la información y sus activos asociados.

Entre otros objetivos de la norma son los siguientes:

- La definición clara y transmitida a toda la organización de los objetivos y directrices de seguridad.
- La sistematización, objetividad y consistencia a lo largo del tiempo en las actuaciones de seguridad.
- El análisis y prevención de los riesgos en los Sistemas de Información.
- La mejora de los procesos y procedimientos de gestión de la información.
- La motivación del personal en cuanto a valoración de la información.
- El cumplimiento con la legislación vigente.
- Una imagen de calidad frente a clientes y proveedores. Propone secuencias de acciones tendientes al:
    - Establecimiento-Implementación.
    - Operación.
    - Monitorización.

- Revisión-Mantenimiento.
- Mejora SGSI Sistema de Gestión de la Seguridad dela Información.

II.3.1.1 Cómo funciona la ISO 27001

El eje central de ISO 27001 es proteger la confidencialidad, integridad y disponibilidad de la información en una empresa. Esto lo hace investigando cuáles son los potenciales problemas que podrían afectar la información (es decir, la evaluación de riesgos) y luego definiendo lo que es necesario hacer para evitar que estos problemas se produzcan (es decir, mitigación o tratamiento del riesgo). [KOOLE15]

Por lo tanto, la filosofía principal de la norma ISO 27001 se basa en la gestión de riesgos: investigar dónde están los riesgos y luego tratarlos sistemáticamente, como se ve en la figura 5.

Figura 5. ***Estructura de ISO 27001***

Fuente: [KOOLE15]

Las medidas de seguridad (o controles) que se van a implementar se presentan, por lo general, bajo la forma de políticas, procedimientos e implementación técnica (por ejemplo, software y equipos). Sin embargo, en la mayoría de los casos, las empresas ya tienen todo el hardware y software, pero utilizan de una forma no segura; por lo tanto, la mayor parte de la implementación de ISO 27001 estará relacionada con determinar las reglas organizacionales (por ejemplo, redacción de documentos) necesarias para prevenir violaciones de la seguridad.

[KOOLE15]

Como este tipo de implementación demandará la gestión de múltiples políticas, procedimientos, personas, bienes, etc., ISO 27001 ha detallado cómo amalgamar todos estos elementos dentro del sistema de gestión de seguridad de la información (SGSI).

Por eso, la gestión de la seguridad de la información no se acota solamente a la seguridad de TI (por ejemplo, cortafuegos, anti-virus, etc.), sino que también tiene que ver con la gestión de procesos, de los recursos humanos, con la protección jurídica, la protección física, etc. [KOOLE15]

II.3.1.2 Beneficios de ISO 27001

Hay 4 ventajas comerciales esenciales que una empresa puede obtener con la implementación de esta norma para la seguridad de la información:

Cumplir con los requerimientos legales, cada vez hay más y más leyes, normativas y requerimientos contractuales relacionados con la seguridad de la información. La buena noticia es que la mayoría de ellos se pueden resolver implementando ISO 27001 ya que esta norma le proporciona una metodología perfecta para cumplir con todos ellos.

Obtener una ventaja comercial, si su empresa obtiene la certificación y sus competidores no, es posible que usted obtenga una ventaja sobre ellos ante los ojos de los clientes a los que les interesa mantener en forma segura su información.

Menores costos, la filosofía principal de ISO 27001 es evitar que se produzcan incidentes de seguridad, y cada incidente, ya sea grande o pequeño, cuesta dinero; por lo tanto, evitándolos su empresa va a ahorrar mucho dinero. Y lo mejor de todo es que la inversión en ISO 27001 es mucho menor que el ahorro que obtendrá.

Una mejor organización, en general, las empresas de rápido crecimiento no tienen tiempo para hacer una pausa y definir sus procesos y procedimientos; como consecuencia, muchas veces los empleados no saben qué hay que hacer, cuándo y quién debe hacerlo. La implementación de ISO 27001 ayuda a resolver este tipo de situaciones ya que alienta a las empresas a escribir sus principales procesos (incluso los que no están relacionados con la seguridad), lo que les permite reducir el tiempo perdido de sus empleados. [KOOLE15]

II.3.1.3 Aplicabilidad de la norma ISO 27001 para auditoria

ISO 27001 (Ver acápite 2.1.3.) es la única norma internacional auditable que define los requisitos para un sistema de gestión de seguridad de la información (SGSI). Donde un SGCI es una parte del sistema de gestión de una organización, basado en una aproximación de los riesgos del negocio (actividad) para establecer, implementar, operar, monitorizar, revisar, mantener y mejorar la seguridad de la información. [KOOLE15]

La creación de un SGSI es una decisión estratégica en una organización y como tal, debe ser apoyada y supervisada por la dirección. El hecho de certificar un SGSI según la norma ISO 27001 puede aportar las siguientes ventajas a la empresa que siga un Modelo de Implementación:

- Demuestra la garantía independiente de los controles internos y cumple los requisitos de gestión corporativa y de continuidad de la actividad comercial.
- Demuestra independientemente que se respetan las leyes y normativas que sean de aplicación.
- Proporciona una ventaja competitiva al cumplir los requisitos contractuales y demostrar a los clientes que la seguridad de su información es primordial.
- Verifica independientemente que los riesgos de la organización estén correctamente identificados, evaluados y gestionados al tiempo que formaliza los procesos procedimientos y documentación de protección de la información.
- Demuestra el compromiso de la cúpula directiva de su organización con la seguridad de la información.
- Los procesos de evaluaciones periódicas ayudan a supervisar continuamente el rendimiento y la mejora.

En la norma ISO 27001, se menciona lo siguiente:

"La organización, establecerá, implementará, operará, monitorizará, revisará, mantendrá y mejorará un documentado SGSI en su contexto para las actividades globales de su negocio y de cara a los riesgos". [KOOLE15]

El mismo documento debe contener:

- La política de seguridad;
- Las normas o estándares de funcionamiento;
- Los procedimientos detallados;

- Guías y recomendaciones.

Para implementar el SGSI se debe utilizar el ciclo continuo PDCA10 cuyo objetivo final es asegurar la Integridad, Confidencialidad y Disponibilidad dela Información.

La metodología PDCA es la médula de la instrumentación básica de la Gestión de la Calidad Total, pero vale la pena insistir en que su extraordinaria potencialidad técnica solo podrá ser bien aprovechada si hay adecuada motivación, participación y valorización de los técnicos y funcionarios. La metodología PDCA está integrada por cuatro pasos. [KOOLE15]

- PLANEAR - PLAN (ESTABLECER EL SGSI): Establecer política, objetivos, procesos y procedimientos SGSI relevantes para manejar el riesgo y mejorar la seguridad de la información para entregar resultados en concordancia con las políticas y objetivos generales de la organización;
- HACER - DO (IMPLEMENTAR Y OPERAR EL SGSI): Implementar y operar la política, controles, procesos y procedimientos SGSI;
- CHEQUEAR- CHECK (MONITOREAR Y REVISAR EL SGSI): Evaluar y, donde sea aplicable, medir el desempeño del proceso en comparación con la política, objetivos y experiencias prácticas SGSI y reportar los resultados a la gerencia para su revisión;
- ACTUAR-ACT (MANTENER Y MEJORAR EL SGSI): Tomar acciones correctivas y preventivas, basadas en los resultados de la auditoría interna SGSI y la revisión gerencial u otra información relevante, para lograr el mejoramiento continuo del SGSI.

El uso eficiente de estos recursos, aplicando rigurosamente los procedimientos correspondientes, permitirá obtener resultados cada vez mejores, alcanzándose las metas establecidas, o por lo menos, acercándose cada vez más a ellas. Se trata de una metodología relativamente simple, pero que, si bien aplicada gerencialmente y con funcionarios motivados, será extraordinariamente efectiva. [KOOLE15]

II.4 Aplicación web

La aplicación web se define como un software dirigido a mostrar información a los usuarios

de la red internet en la mayoría de los casos. Esto puede contener elementos que permiten una comunicación activa entre el usuario y la información. Esto permite que el usuario acceda a los datos de modo interactivo, gracias a que la aplicación responde a cada una de sus acciones, como por ejemplo rellenar y enviar formularios, participar en juegos diversos y acceder a gestores de base de datos de todo tipo con los que cuenta una aplicación web. [THEOPEN09]

II.4.1 Estructura de una aplicación web

Está estructurada como una aplicación en tres capas en su forma más común. La primera capa consiste en el navegador web, la segunda capa y central u motor capaz de usar la tecnología web dinámica, y la base de datos que constituye la tercera capa. El navegador web manda peticiones a la capa central que ofrece servicio de interfaz gráfica, permitiendo interactuar con la base de datos valiéndose de las consultas y actualizaciones. [THEOPEN09]

II.4.2 Arquitectura de una aplicación web

Se utiliza el término arquitectura web, para definir una tarea que requiere conocimientos técnicos de construcción, funcionales y de diseño para sitios o páginas web. La construcción de páginas web requiere una compleja conjunción de diferentes sistemas integrados entre sí: servidores, bases de datos, organización de la información, etcétera.

Tal como en la arquitectura tradicional, actualmente el foco para el diseño y construcción de páginas web se centra en el usuario y sus requerimientos.

La arquitectura de un sitio web tiene tres componentes principales:

- Un servidor Web
- Una conexión de red
- Uno o más clientes

El servidor web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor web, éste localiza y envía al navegador par su despliegue.

Una importante característica de aplicaciones web es que fue proyectado para funcionar en topología de Internet. De esta forma el web incorpora naturalmente la característica de ser un

ambiente distribuido y multiplataforma. [JAV10]

II.4.3 Servicios web

Es un conjunto de protocolos y estándares que se utilizan para intercambiar datos entre aplicaciones. Las distintas aplicaciones desarrolladas con diferentes tecnologías de implementación y ejecutas sobre cualquier plataforma permitiendo la utilización de los servicios web para intercambiar datos en redes internet. [THEOPEN09]

II.5 Seguridad de aplicaciones web

La seguridad es un aspecto importante para proteger la integridad y privacidad de los datos y recursos de las aplicaciones web. Debiendo designar una estrategia de seguridad para la aplicación web, que use soluciones de seguridad de eficacia probada, e implementar métodos de autenticación, autorización y validación de datos, para proteger la aplicación de una serie de amenazas. [ROM14]

II.5.1 Validación de datos

La validación de datos es un proceso que consiste en realizar un filtro de los datos proporcionados por el usuario, si estos son correctos permite el acceso caso contrario regresa mensajes de erros basándose en procedimientos definidos en la implementación.
Es una de las áreas más importantes a tener en cuenta en las aplicaciones orientadas a la web, para la no existencia de código oculto de por medio. [THEOPEN09]

II.5.2 Manejo de sesiones

Es un método ampliamente utilizado en muchos tipos de aplicaciones web, es la secuencia de páginas que usuario visita en un sitio web desde que entra hasta que lo abandona.
Tiene como objetivo primordial asegurar que solo los usuarios autenticados posean una amplia navegación y acceso a la información requerida, haciendo cumplir los controles de autorización previniendo los ataques. [THEOPEN09]

II.5.3 Ataque a la aplicación web

Se refiere a una acción o un método por el cual individuo haciendo uso de un sistema informático intenta dañar o alterar una aplicación web.

Es un intento organizado y deliberado de una o más personas para infiltrarse en las aplicaciones web de grandes organizaciones preferentemente por la confidencialidad de la información que manejan. [VVA13]

II.5.4 Riesgos en una aplicación web

El uso masivo de las aplicaciones web en la actualidad generado un riesgo, por tal motivo se debe emplear políticas de seguridad en el desarrollo, desde la fase inicial de su diseño hasta su puesta en funcionamiento, sin embargo, muchas veces la seguridad no es tomada en cuenta hasta que los fallos se presentan. En el trayecto del desarrollo no solo debe dar importancia a los usuarios y sus requerimientos, sino también en los eventos que puedan inferir con la seguridad y el contenido de la información. [VVA13]

II.5.5 Vulnerabilidad

La vulnerabilidad es la debilidad de un recurso o grupo de recursos que son aprovechados por una o varias amenazas generadas a partir de la falta de seguridad en el sistema esto ocurren en dos factores Hardware o Software.

La vulnerabilidad del hardware es la facilidad de acceso a los dispositivos y la vulnerabilidad del software son las fallas o debilidades del sistema. [ROM14]

Las aplicaciones web pueden presentar diversas vulnerabilidades que van de acuerdo a los servicios que prestan. De acuerdo con la OWASP (Open Web Application Security Project) las vulnerabilidades en aplicaciones web más explotadas, fueron las siguientes:

- Cross Site Scripting (XSS).
- Ataques de inyección de código.
- Ejecución de archivos maliciosos.
- Insecure Direct Object Reference.
- Ataques Cross Site Request Forgery (CSRF).
- Pérdidas de información y errores al procesar mensajes de error.
- Robo de identidad de autentificación.
- Almacenamiento criptográfico inseguro.
- Comunicaciones inseguras.
- Acceso a URLs ocultas no restringidas de manera adecuada.

Estas vulnerabilidades son aprovechados para realizar ataques, es por eso que es necesario tomar en cuenta durante el proceso de implementación de las aplicaciones web. [ROM14]

II.6 Sistema de detección de intrusos

Uno de los mecanismos de defensa más usados para reducir el riesgo de las compañías ante ataques dirigidos hacia los bienes informáticos han sido los sistemas de detección de Intrusos o IDS (Intrusion Detection Systems). [KIO14]

Un IDS es un elemento que escucha y analiza toda la información que circula por una red de datos e identifica posibles ataques. Cuando aparece un ataque, el sistema reaccionará informando al administrador y cerrará las puertas al posible intruso reconfigurando elementos de la red como firewalls y routers.

Los IDS han sido usados ampliamente a lo largo de estos últimos años por muchas compañías porque han proporcionado una capa adicional de seguridad. Sin embargo, se ha encontrado que estos elementos proporcionan seguridad reactiva, es decir para que haya protección debe existir primero un ataque. Si un ataque es pequeño y efectivo el IDS reaccionará demasiado tarde y el ataque logrará su objetivo. [KIO14]

Los IDS permiten, no solo la detección de ataques cubiertos por otros componentes de seguridad, sino la detección de intrusiones que pasan desapercibidas a otros componentes del dispositivo de seguridad. Así pues, realizan dos tareas fundamentales: la prevención y la reacción, como se observa en la figura 6.

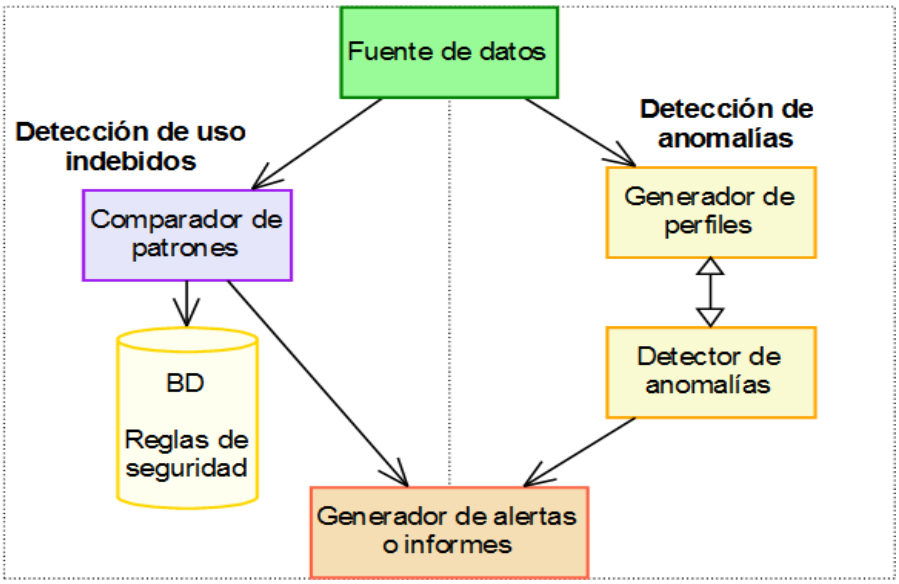


Figura 6. ***Esquema general de un IDS***

Fuente: [GON10]

II.6.1 Funciones de un IDS

Estos sistemas introducen métodos de trabajo que permiten complementar y completar el trabajo realizado por otras herramientas de seguridad como los cortafuegos. Las funciones de un IDS se pueden resumir de la siguiente forma: [PFC11]

- Detección de ataques en el momento que están ocurriendo o poco después.
- Automatización de la búsqueda de nuevos patrones de ataque, gracias a herramientas estadísticas de búsqueda, y al análisis de tráfico anómalo.
- Monitorización y análisis de las actividades de los usuarios. De este modo se pueden conocer los servicios que usan los usuarios, y estudiar el contenido del tráfico, en busca de elementos anómalos.
- Auditoría de configuraciones y vulnerabilidades de determinados sistemas;
- Descubrir sistemas con servicios habilitados que no deberían de tener, mediante el análisis del tráfico y de los logs.
- Análisis de comportamiento anormal. Si se detecta una conexión fuera de hora, reintentos de conexión fallidos y otros, existe la posibilidad de que se esté en presencia de una intrusión. Un análisis detallado del tráfico y los logs puede revelar una máquina comprometida o un usuario con su contraseña al descubierto.

- Automatizar tareas como la actualización de reglas, la obtención y análisis de logs, la configuración de cortafuegos y otros.
- Logs que nos permite visualizar los acontecimientos y la funcionalidad de una aplicación.

Un IDS puede compartir u obtener información de otros sistemas como firewalls, routers y switches, lo que permite reconfigurar las características de la red de acuerdo a los eventos que se generan. También permite que se utilicen protocolos como SNMP (Simple Network Management Protocol) para enviar notificaciones y alertas a otras máquinas de la red. Esta característica de los IDS recibe el nombre de interoperabilidad.

Otra característica a destacar en los IDS, es la correlación, que consiste en la capacidad de establecer relaciones lógicas entre eventos diferentes e independientes, lo que permite manejar eventos de seguridad complejos que individualmente no son muy significativos, pero que analizados como un todo pueden representar un riesgo alto en la seguridad del sistema. [PFC11]

La utilidad de un sistema de detección de intrusos debe ser evaluada teniendo en cuenta la probabilidad que tiene el sistema de detectar un ataque y la probabilidad de emitir falsas alarmas. Teniendo en cuenta el gran número de alertas que se generan y la gran cantidad de falsas alarmas producidas, la gestión o revisión de éstas se convierte en una tarea muy complicada y la carga de trabajo se multiplica para los administradores de sistemas. Son pues, sistemas que requieren de un mantenimiento y una supervisión constante. [PFC11]

Actualmente, hay herramientas gráficas que permiten visualizar los datos almacenados por los sistemas de detección y presentan los datos en distintas interfaces permitiendo encontrar y analizar detalles, tendencias y problemas que a simple vista no encontraríamos.

Estas herramientas es un aporte de beneficioso para detectar los ataques y prevenir de alguna manera, y realizar acciones necesarias. [PFC11]

II.6.2 Tipos de sistemas de detección de intrusos

Existen distintos tipos de IDS, atendiendo a distintas clasificaciones establecidas de acuerdo a las características que se usen para establecer dicha clasificación. Cada uno de ellos se caracteriza por diferentes aproximaciones de monitoreo y análisis y presenta distintas

ventajas y desventajas como se observa en la figura 7. [PFC11]

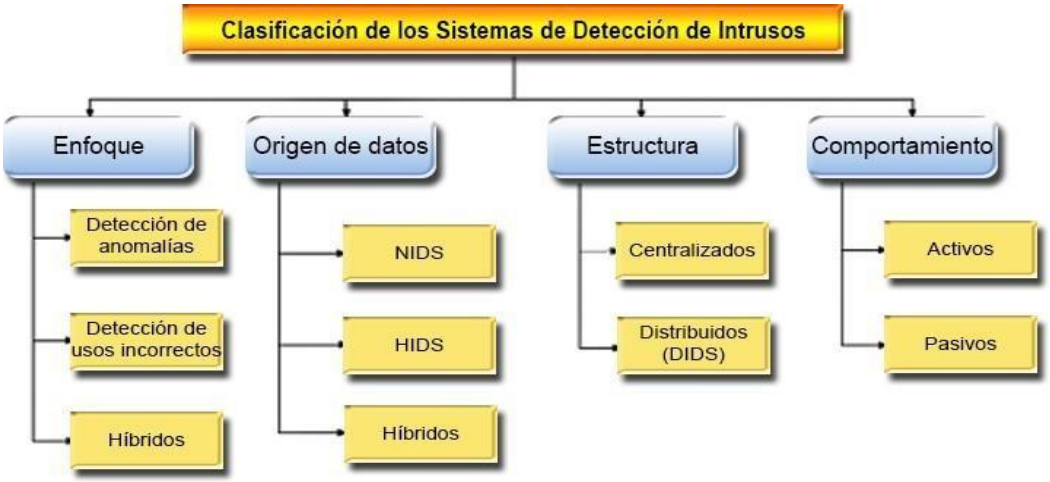


Figura 7. Clasificación del IDS

Fuente: [GON10]

II.6.2.1 Tipos de IDS en función del origen de los datos

Los IDS pueden clasificarse atendiendo a las fuentes de información que se utilicen.
Así tenemos tres tipos: IDS basados en host, en Red y los IDS híbridos. [PFC11]

II.6.2.1.1 HIDS: Host-based Intrusion Detection Systems

Los HIDS están diseñados para monitorizar, detectar y responder a los datos generados por un usuario o un sistema en un determinado host, fueron los primeros IDS que surgieron. Estos IDS son útiles para identificar amenazas e intrusiones a nivel del host local. Monitorizan gran cantidad de eventos, analizando actividades con una gran precisión, determinando de esta manera qué procesos y usuarios se involucran en una determinada acción. Recaban información del sistema como ficheros, logs, recursos, para su posterior análisis en busca de posibles incidencias.

Sin embargo, cuando un sistema comprende cientos de hosts, los HIDS, por sí solos, no son viables para realizar una monitorización adecuada. Requieren confianza en el sistema donde se van a instalar (un sistema de detección de intrusos de host no será muy útil en un sistema que ha sido comprometido anteriormente). Impactan directamente sobre el sistema que protegen; dado que comparten los mismos recursos que el sistema y aplicaciones que protegen y son vulnerables ante un ataque directo. [PFC11]

Los HIDS utilizan las bitácoras del sistema, las cuales se generan de forma automática por diferentes aplicaciones o por el propio núcleo del sistema operativo. Como se muestra en la Figura 8, se hace un análisis de las bitácoras prestando especial atención a los registros relativos a demonios de red, como un servidor web o el propio inetd. Por otra parte, se usan también verificadores de integridad de determinados ficheros de importancia vital para el sistema, como el de contraseñas. [PFC11]

Figura 8. ***Sistema de detección de intrusos basado en host***

Fuente: [PFC11]

II.6.2.1.2 NIDS: Network Intrusion Detection Systems

Los NIDS son muy parecidos a los HIDS en tanto que monitorizan, detectan y responden ante los datos generados, pero en vez de proteger un determinado host, reciben los datos de la red local donde estén instalados. Tienen la ventaja de ser, normalmente, pasivos, de forma que no interfieren en el correcto uso de la red. Actúan mediante la utilización de un dispositivo de red configurado en modo promiscuo, capturando todos los paquetes que pasan por él y almacenando la información de estos paquetes en un repositorio para su posterior análisis en busca de patrones indicativos de un ataque. Véase la figura 9.

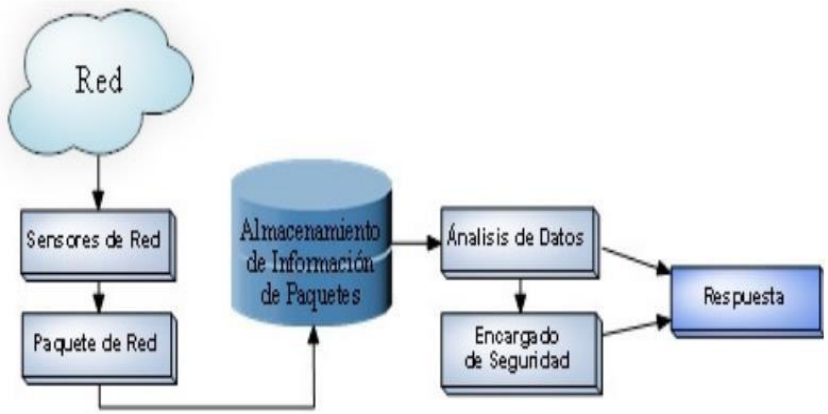


Figura 9. ***Sistema de detección de intrusos basado en red***

Fuente: [PFC11]

Analizan el tráfico de red, normalmente, en tiempo real. No sólo trabajan a nivel TCP/IP, también lo pueden hacer a nivel de aplicación. Permiten la detección de ataques a un mayor nivel de abstracción, puesto que ahora no solo tienen información de un solo host, sino de múltiples. No obstante, los NIDS solo perciben información que pasa por la red, siendo inútil ante los ataques locales de los usuarios de un determinado host. [PFC11] Un NIDS, puede protegernos contra los ataques que entran a través de los cortafuegos hasta la LAN Interna. Los cortafuegos pueden estar mal configurados, permitiendo la introducción de tráfico no deseado en nuestra red. Incluso cuando funcionan

correctamente, los cortafuegos normalmente dejan pasar algún tráfico de aplicación que puede ser peligroso, ya que sólo puede ver el tráfico que lo atraviesa desde el exterior y no respecto a la actividad de la LAN.

Podemos pensar en los NIDS y en los cortafuegos como dispositivos de seguridad complementarios. Lo que llega a los puertos de cortafuegos, permitido por las reglas, se envía a los servidores internos provocando un tráfico potencialmente dañino. Un NIDS puede comprobar dicho tráfico y marcar los paquetes sospechosos. Configurado correctamente puede hacer una doble comprobación de las reglas de nuestro cortafuegos y proporcionarnos una protección adicional para los servidores de aplicación. Bien ubicados, pueden analizar grandes redes y su impacto en el tráfico suele ser pequeño.

Los NIDS, requieren emular el comportamiento de los sistemas que protege para mantener una sincronía entre su procesamiento y el de estos equipos; si existe asincronía, el sistema puede ser evadido. También necesita capacidad de procesamiento suficiente para evitar la

pérdida de paquetes y visibilidad de los equipos que vigila (configurar puertos de vigilancia en switches para permitir a este tipo de sistemas vigilar el tráfico de los sistemas que protege puede tener consecuencias de desempeño en todo el segmento de red). [PFC11]

II.7 Detección de riesgos en las aplicaciones web

En 1996 Dan Farmer realizo un estudio comparativo acerca de la detección de intruso, en los sistemas informáticos y sitios web de comercio haciendo uso de técnicas sencillas, catalogando los tipos de problema en dos grupos: [SEIN15]

- Rojo: En este grupo se encuentran los sistemas potencialmente sensibles a ataques, es decir los problemas de seguridad son conocidos por los atacantes. Podemos mencionar el servicio FTP mal configurado;
- Amarillo: Los ataques de este grupo son menos serios ya que el problema detectado no implica un daño serio e inmediato al sistema, sin embargo, esto no significa que no exista el riesgo y que no cause daños como la perdida de información, modificación, o bien robo de la misma.

En la siguiente ***tabla 4*** se muestran los sistemas que fueron evaluados y la categoría en la cual se asignaron, y el porcentaje de vulnerabilidad. Se observa que los sitios de noticias son mucho más vulnerables, seguidos de los sitios gubernamentales.

Tipos	Cantidad	Vulnerabilidad (%)	Amarilla (%)	Roja (%)
Bancos	660	68.34	32.73	35.61
Financieros	274	51.1	30.66	20.44
Gobiernos	47	61.1	23.4	38.30
Informativas	312	69.55	30.77	38.78
Otros	469	33.05	15.78	17.27
Total	**1762**	**56,628**	**26,668**	**30,08**

Tabla 4: ***Porcentaje de vulnerabilidad por tipo de sitio***

Fuente: [SEIN15]

Haciendo un énfasis, se mención a tres principales puntos de ataque hacia una aplicación web, independientemente de la tecnología aplicada en su desarrollo:

- Ataque hacia el cliente.
- Ataque hacia el canal de comunicación.
- Ataque hacia servidores.

Estos medios al ser los principales medios de comunicación de una aplicación web son los que más sufren ataques. [SEIN15]

II.7.1 Detección de riesgos del lado del cliente

En la tabla 5 observa la clasificación de riesgos encontrados de lado del cliente, con la descripción respectiva.

RIESGO	CAUSAS	IMPACTO
Phishing Es la capacidad de duplicar una página web para hacer creer al visitante que se encuentra en la página original en lugar de la apropiada. Normalmente se utiliza con fines delictivos duplicando	• Falta de cultura informática; • Políticas deficientes; • Inadecuada propaganda.	• Desprestigio del sitio web; • Fraudes.

páginas web de bancos conocidos y enviando indiscriminadamente correos para que se acceda a esta página para actualizar las páginas de acceso al banco.		
Falta de control en las "Transacciones" Las transacciones son operaciones entre componentes, un ejemplo sería una transacción cuando se modifica el estado de una ID, por esto es que en una aplicación web se pone empeño en la operación de las transacciones, ya que estas conllevan a la falla de las aplicaciones.	• Transaccion es limitadas.	• Robo o pérdida de la información; • Acceso a zonas no autorizadas del sistema.
Fraudes por ausencia de "Roles" Un rol se define como un perfil que designa el comportamiento de los usuarios que tienen acceso a la aplicación, estos se distinguen entre sí por el nivel de privilegios que poseen respecto a la Aplicación Web.	• Ausencia de Roles	• Difícil administración de la aplicación; • Cambios sin autorización en cuentas de otros usuarios.

Cross Site Scripting (CSS) Una aplicación web puede ser usada como un mecanismo para transportar un ataque al navegador del usuario final. Un ataque exitoso puede comprometer el Token de sesión del usuario final, atacar la maquina local o enmascarar contenido para engañar al usuario.	• Validadores de entradas de datos deficientes.	• Desprestigio de la aplicación web.
Ataques Unicode El ataque Unicode es utilizado para forzar a los servidores web a salir de la raíz de la aplicación y acceder a archivos residentes en otras partes de la ruta de la aplicación Esto es logrado provocando un error en el cliente.	• Validadores de entradas de datos deficientes.	• Divulgación de información privada de aplicaciones web tales como rutas, direcciones, puertos.
Inyección de código Técnica consistente en aprovechar las debilidades que ofrece la aplicación en sus	• Validaciones de entradas de datos deficientes.	• Alteración del comportamiento de la aplicación, provocando:

validaciones e ingresar código malicioso en sus campos de entrada permitiendo que estos se ejecuten en la aplicación web.		• Acceso a zonas no autorizadas de la aplicación web; • Alteración del comportamiento de la aplicación.

Tabla 5: ***Detección de riesgos al lado del cliente***

II.7.2 Detección de riesgos del lado del servidor

En la siguiente tabla 6 se presentan los riesgos y las técnicas de ataque al lado del servidor.

RIESGO	CAUSAS	IMPACTO
Desbordamiento de buffer Debilidad ligada al servidor, ya que esta tiene el control de los procesos que ejecuta la aplicación, esta condición ocurre cuando los datos escritos en la memoria exceden el tamaño reservado en el buffer y direcciones de memoria adyacentes son sobrescritas causando que la aplicación falle o termine de manera inesperada. Los atacantes suelen corromper los buffer de memoria para interrumpir el funcionamiento de la	• Validaciones de entradas de los datos deficientes; • Mala programación.	• La parte que afecta directamente este tipo de ataques es el Back end de la aplicación Por ejemplo puede causar problemas en el funcionamiento de la ID.

aplicación en el servidor y así denegar los servicios. Esto lo logran introduciendo grandes cantidades de información al sistema web, por ejemplo, en una aplicación de Comercio Electrónico un carrito de compras con un excesivo número de productos puede ser causa inmediata para un desbordamiento de Memoria.		
Caída del servidor por mal manejo de errores. Condiciones de error que ocurren durante la operación normal que no son manejadas adecuadamente.	• Manejo inadecuado de errores.	• Si un agresor puede causar que ocurran errores que la aplicación web no maneja, este puede obtener información detallada del sistema, denegar servicios, causar que mecanismos de seguridad fallen.
Administración de autenticación y sesión de servicio interrumpida	• Mala programación.	• Fraudes en la aplicación.
Acceso y modificación a configuraciones del Servidor	• Administración de configuración Insegura.	• Fallas de seguridad en el software del servidor; • Alteraciones al

		listado de directorio o acceso no autorizado de directorio.
Generación de basura dentro del servidor	• Administración de configuración Insegura.	• Generación de archivos innecesarios, por ejemplo, de respaldo o de ejemplo, incluyendo Scripts, Logs, aplicaciones archivos de configuración y páginas web.
Vulnerabilidad en los accesos a las propiedades de configuración del servidor	• Administración de configuración Insegura.	• Permisos no adecuados en archivos y directorios.
Infiltración de los usuarios malintencionados o no autorizados	• Cuentas de usuario definidas por defecto en la instalación.	• Acceso no autorizado a propiedades del servidor final.
Perdida de información del servidor web	• Funciones administrativas o de depuración que son habilitadas o accesibles	• Alto costo en recuperación de información de sistema.

Tabla 6: ***Detección de riesgos del lado del servidor***

II.7.3 Detección de riesgos en el canal de comunicación

En la tabla 7 se muestra riesgos en el trayecto, es decir en el recorrido de la información por la red, la información solicitada puede ser interceptada y ser modificado con datos ajenos.

RIESGO	CAUSAS	IMPACTO
Robo de información en los canales de información	• Certificados SSL y opciones de encristalaron mal configurados o no habilitados.	• Alto costo en recuperación de información sensible de la aplicación que corre sobre el servidor web.

Interceptación de información sensible Las aplicaciones web frecuentemente utilizan funciones de criptografía para proteger información y credenciales. Estas funciones y el código que integran a ellas han sido difíciles de codificar adecuadamente, lo cual frecuentemente redunda en una protección débil.	• Almacenamiento de información insegura.	• Robo de información sensible.

Tabla 7: ***Detección de riegos en el canal de comunicación***

II.8 Seguridad en Bases de Datos

La gran mayoría de los datos sensibles del mundo están almacenados en sistemas gestores de bases de datos comerciales tales como Oracle, Microsoft SQL Server entre otros, y atacar una base de datos es uno de los objetivos favoritos para los criminales. Esto puede explicar por qué los ataques externos, tales como inyección de SQL, subieron 56.2% en 2012, "Esta tendencia es prueba adicional de que los agresores tienen éxito en hospedar páginas Web maliciosas, y de que las vulnerabilidades y explotación en relación a los navegadores Web están conformando un beneficio importante para ellos. [VIL12]

Mientras que la atención generalmente se ha centrado en asegurar los perímetros de las redes por medio de, firewalls, IDS / IPS y antivirus, cada vez más las organizaciones se están enfocando en la seguridad de las bases de datos con datos críticos, protegiéndolos de intrusiones y cambios no autorizados.

En las siguientes secciones daremos las siete recomendaciones para proteger una base de datos en instalaciones tradicionales. [VIL12]

II.8.1 Inyección de código

La inyección de código es común entre los individuos que gustan por atacar las páginas web, generalmente usan esta técnica para obtener otra información que les ayude a atacar en mayor grado los sistemas.

Para ello hace uso de los errores al procesar información errónea, esto es usado por un atacante para cambiar la ejecución o funcionalidad normal, más aún puede ser usado para la propagación de algunos códigos maléficos con fines de dañar o alterar la información.

Una inyección de código puede ser de tipo SQL o Fichero, el de SQL se aprovecha de la sintaxis en la instrucción y el de fichero hace referencia a un archivo externo. [VIL12]

II.8.2 Inyección de código maliciosa

Este tipo de inyección de código malicioso se realiza de la siguiente manera.

- Instalación de malware en alguna computadora por medio de la inyección de código a un navegador web o en sus plugins.
- Instalación de malware inyectando código en aplicaciones web desarrollados en cualquiera de las plataformas disponibles.
- La inyección de código puede ser usada por medio del shell del sistema operativo, para obtener mayores privilegios de los permitidos.

Robo de sesiones desde el navegador Web usando inyección HTML/Script (Cross-site scripting). [SEIN15]

II.8.3 Inyección de código benéfico

Así como hay inyección de código maliciosa, también hay inyección de código benéfica para el programador, por ejemplo, se puede modificar una tabla de la base de datos de un sistema

existente usando la inyección de datos. Básicamente la inyección de código benéfica es útil para modificar el sistema de alguna manera eficiente y con menores costos.
La inyección de código beneficioso como puede ser útil para modificar algunos componentes de una aplicación, también puede ser perjudicial en algunos casos. [SEIN15]

II.8.4 Inyección de código inesperado

Es cuando el usuario ingresa caracteres inválidos en el sistema, lo cual puede ocasionar que funcione indebidamente con comportamientos inesperados, es por eso que se recomienda que se tenga un control de caracteres en los campos donde se requiere que el usuario ingrese datos al sistema.
Atacando a través de campos de entrada incorrectamente validados, o explotando alguna vulnerabilidad presente. Con el propósito de infectar el más grande número posible de sitios web con un sencillo mecanismo, los atacantes intentarán atacar un tipo específico de clase buscando vulnerabilidades comunes en los sitios y generalmente automatizando el descubrimiento y exploración. Esto permite a los atacantes infectar sitios web con la eficiencia comúnmente encontrada en Gusanos de Internet. [SEIN15]

II.8.5 Inyección SQL

Es la inserción de código SQL por medio de entradas desde la parte del cliente hacia la aplicación en una petición. Permitiendo al atacante modificar las consultas programadas en la aplicación y ejecutar otras acciones totalmente distintas con la intención de acceder a la información privilegiada o confidencial, borrar los datos almacenados, entre otras muchas cosas. [VIL12]
Como consecuencias de estos ataques y dependiendo de los privilegios que tenga el usuario de la base de datos bajo el que se ejecutan las consultas, se podría acceder no sólo a las tablas relacionadas con la aplicación, sino también a otras tablas pertenecientes a otras bases de datos alojadas en ese mismo servidor web.
Esto ocurre a causa ciertos caracteres en los campos de entrada de información por parte del usuario, ya sea mediante el uso de los campos de formularios que son enviados al servidor mediante POST o GET en las urls de las páginas web, lo mismos posibilitan coordinar varias consultas SQL o ignorar el resto de la consulta, para ello una alternativa es utilizar

validaciones más seguras y cambios en la sintaxis de instrucción. [VIL12]

II.8.6 Encriptación de base de datos

En la actualidad la mayor parte de bases de datos contienen la información sensible, propia, o privada. Esto puede incluir la información de confidencial. La llave al mantenimiento de esta información en una manera segura es la confidencialidad. [CRIP06]

La seguridad de los datos implica protegerlos de operaciones indebidas que pongan en peligro su definición, existencia, consistencia e integridad independientemente de la persona que los accede. Esto se logra mediante mecanismos que permiten estructurar y controlar el acceso y actualización de los mismos sin necesidad de modificar o alterar el diseño del modelo de datos; definido de acuerdo a los requisitos del sistema o aplicación software. [CRIP06]

Toda encriptación se encuentra basada en un Algoritmo, la función de este Algoritmo es básicamente codificar la información para que sea indescifrable a simple vista, por ejemplo, de manera que una letra "A" pueda equivaler a:"5x5mBwE" o bien a "xQE9fq", el trabajo del algoritmo es precisamente determinar cómo será transformada la información de su estado original a otro que sea muy difícil de descifrar. [CRIP06]

Una vez que la información arribe a su destino final, se aplica el algoritmo al contenido codificado "5x5mBwE" o bien a "xQE9fq" y resulta en la letra "A" o según sea el caso, en otra letra. Hoy en día los algoritmos de encriptación son ampliamente conocidos, es por esto que para prevenir a otro usuario "no autorizado" descifrar información encriptado, el algoritmo utiliza lo que es denominado llave ("key") para controlar la encriptación y desencriptación de información. Algunos algoritmos son DES (algoritmo simétrico) AES que posiblemente suplantará a DES y uno de los más conocidos RSA (algoritmo asimétrico).

II.8.6.1 Encriptación de datos en transito

La mayoría de los ambientes de base de datos utilizan TCP/IP y el servidor de base de datos escucha algunos puertos y acepta las conexiones iniciadas por los clientes de la base de datos. Mientras que los puertos son configurables, la mayoría de las veces se utiliza los puertos por defecto del servidor que está usando, por ejemplo: puerto 1433 para el servidor de Microsoft SQL, puerto 1521 para Oracle, puerto 4100 para Sybase, puerto 50000 para DB2, y puerto

3306 para MySQL. [CRIP06]

Los clientes de la base de datos se conectan con el servidor sobre estos puertos predefinidos para iniciar una comunicación, dependiendo del tipo de la base de datos y la configuración del servidor, redireccionando a otro puerto o terminando la comunicación entera sobre el mismo puerto del servidor. [CRIP06]

En un alto nivel, esto significa que, con las herramientas derechas y el acceso correcto a la red, cualquiera puede golpear ligeramente en sus conversaciones de la base de datos y escuchar detrás de las puertas encendido el acceso de base de datos capturando y robando las declaraciones que se publica también los datos enviados por el servidor de base de datos. [CRIP06]

Meterse en tu base de datos de comunicaciones es relativamente fácil, porque las bases de datos de comunicaciones son en su mayoría en texto legible, mediante el uso de los servicios públicos y en su mayoría simples herramientas libres, que un hacker pueda escuchar y robar información. La manera de evitar que esto suceda es encriptar las comunicaciones entre la base de datos de clientes y servidores de bases de datos. Este tipo de cifrado es llamado cifrado de los datos en tránsito porque todos, las comunicaciones entre el cliente y el servidor están encriptadas. La encriptación se produce en los extremos. [CRIP06]

Aunque el cifrado de los datos en tránsito se está convirtiendo en gran importancia, porque la mayoría no encripta los datos en tránsito, y para muchos ambientes en que está perfectamente bien. Si usted cree que un potencial espía, es algo que no puede vivir con él, entonces se debería cifrar los datos en tránsito.

II.8.7 Cross site scripting (XSS)

Es un agujero de seguridad común en las aplicaciones web que permite inyectar código JavaScript evitando medidas de control de seguridad.

No solamente se limita a sitios web, también en aplicaciones locales o en los mismos navegadores, porque no se validan de manera correcta los datos de entrada. [OWA14]

II.9 Ataques DDOS

Para poder identificar las amenazas que puede correr una aplicación web es necesario primero conocer los tipos de ataques, las formas de acceso, el objetivo del mismo y la forma en la que

opera. La forma en la que afecta el ataque DDoS (Denegación de Servicios) es en bajar los servicios que deberían de estar disponibles, el ataque corrupción de los datos consiste en modificar la información afectando así la estabilidad del negocio. El ataque DDoS afecta a todo tipo de plataformas, una forma de evitar este tipo de ataques es monitoreando las IPs que realizan la petición a nuestra aplicación si estas sobrepasan de un número de conexiones HTTP en rango de tiempo corto es considerado
un ataque. [SEIN15]

El código malicioso es un ataque informático que requiere de la intervención del usuario para propagar; Últimamente Latinoamérica es una de las regiones más afectadas por Gaobot, Spybot y Netsky P.

La forma de propagación de este último consiste en un envío masivo de sí mismo usando una extensión .zip ya que de esta forma puede evadir los filtros de seguridad, estos archivos con esta extensión .zip generalmente son confiables, por tal motivo el usuario final abre el archivo provocando la propagación del virus. [SEIN15]

II.10 Mejores prácticas en el desarrollo

Existen varias mejores prácticas que se pueden implementar para las aplicaciones web desarrolladas por las empresas dedicadas al área del desarrollado, dichas prácticas pueden ser ITIL, CMMi, ISO27001, Six Sigma, entre otros. A continuación, se presenta una breve descripción de cómo apoyarse en ITIL y COBIT que son las más utilizadas en la actualidad.

II.10.1 ITIL

Las empresas de desarrollo de aplicaciones necesitan concentrarse en la calidad de los servicios que brindan, y asegurarse que los mismos estén alineados a los objetivos de la organización que las contrata.

Cuando los servicios son críticos, cada una de las actividades que se realizan deben de estar ejecutadas con un orden determinado para asegurar que la empresa encargada o el desarrollador proporcione la entrega de los servicios de forma consistente, esto brinda un mejor rendimiento en la calidad. [QUE11]

ITIL tiene que ver con todos aquellos procesos que se requieren ejecutar dentro de las organizaciones para la administración y operación de la infraestructura de las empresas

encargadas de la implementación, de tal forma que se tenga una óptima provisión de servicios a los clientes bajo un esquema de costos congruentes con las estrategias del negocio.

Desarrollada su primera versión a finales de 1980, la Biblioteca de Infraestructura de Tecnologías de la Información (ITIL) se ha convertido en el estándar mundial de facto en la Gestión de Servicios Informáticos. Uno de los conceptos esenciales de ITIL es que establece que para una adecuada Gestión de Servicios en las Tecnologías de Información es necesaria una mezcla entre tres factores: Personas, Procesos y Tecnología, son las principales implicancias con las tecnologías de información. [QUE11]

Como las características que tiene ITIL podemos mencionar lo siguiente:

- Es un Framework de procesos de desarrollo no propietario;
- Es independiente de los proveedores;
- Es independiente de la tecnología;
- Está basado en "Best Practices". Y además provee:
- Una terminología estándar;
- Las interdependencias entre los procesos;
- Los lineamientos para la implementación;
- Los lineamientos para la definición de roles y responsabilidades de los procesos;
- Las bases para la comparación de la empresa frente a las “mejores prácticas”.

En la administración o la gestión de servicios informáticos es abarcada por dos publicaciones: Entrega de Servicios y Soporte de Servicios. [QUE11]

Entrega de Servicios: Cubre los procesos necesarios para la planeación y entrega de la calidad de los servicios de tecnologías de información, estos procesos son:

- Administración de niveles de servicio;
- Administración financiera;
- Administración de capacidad;
- Administración de la continuidad de servicios;
- Administración de la Disponibilidad.

Soporte de Servicios: Proporciona los detalles de la función de mesa de servicio y los procesos necesarios para el soporte y mantenimiento de los servicios, estos procesos son los siguientes:

- Administración de incidentes;

- Administración de problemas;
- Administración de configuraciones;
- Administración de cambios;
- Administración de releases.

La Gestión de servicios organiza las actividades necesarias para administrar la entrega y soporte de servicios en procesos.

Un proceso es una serie de actividades que a partir de una entrada obtienen una salida. El flujo de la información dentro y fuera de cada área de proceso indicará la calidad del proceso en particular. [QUE11]

Existen puntos de monitoreo en el proceso para medir la calidad de los productos y provisión de los servicios.

Los procesos pueden ser medidos por su efectividad y eficiencia, es decir, si el proceso alcanzó su objetivo y si se hizo un óptimo uso de los recursos para lograr el objetivo.

Por lo que si el resultado de un proceso cumple con el estándar definido, entonces el proceso es efectivo, y si las actividades en el proceso están cumpliendo con el mínimo esfuerzo requerido y costo, entonces el proceso es eficiente. [QUE11]

II.10.2 COBIT

El estándar COBIT (Control Objectives for Information and related Technology) ofrece un conjunto de mejores prácticas para la gestión de los Sistemas de Información de las organizaciones. [MEA13]

El objetivo principal de COBIT consiste en proporcionar una guía a alto nivel sobre puntos en los que establecer controles internos con tal de:

- Asegurar el buen gobierno, protegiendo los intereses de los clientes, accionistas, empleados;
- Garantizar el cumplimiento normativo del sector al que pertenezca la organización;
- Mejorar la eficacia y eficiencia de los procesos y actividades de la organización;
- Garantizar la confidencialidad, integridad y disponibilidad de la información.

El estándar define el término control como: “Políticas, procedimientos, prácticas y estructuras organizacionales diseñadas para proveer aseguramiento razonable de que se lograrán los objetivos del negocio y se prevendrán, detectarán y corregirán los eventos no

deseables" [MEA13]

Por tanto, la definición abarca desde aspectos organizativos (flujo para pedir autorización a determinada información, procedimiento para reportar incidencias, selección de proveedores) hasta aspectos más tecnológicos y automáticos (control de acceso, monitorización de los sistemas mediante herramientas automatizadas).

Por otra parte, todo control tiene por naturaleza un objetivo. Es decir, un objetivo de control es un propósito o resultado deseable como, por ejemplo: garantizar la continuidad de las operaciones ante situaciones de contingencias.

En consecuencia, para cada objetivo de control de nuestra organización puede implementar uno o varios controles (ejecución de copias de seguridad periódicas, traslado de copias de seguridad a otras instalaciones) que nos garanticen la obtención del resultado deseable (continuidad de las operaciones en caso de contingencias). [MEA13]

COBIT clasifica los procesos de negocio relacionados con las tecnologías de la información en cuatro dominios:

- Planificación y organización;
- Adquisición e implementación;
- Entrega y soporte;
- Supervisión y evaluación.

En la planificación y organización, la dirección de la organización debe implicarse en la definición de la estrategia a seguir en el ámbito de los sistemas de información, de forma quesea posible proporcionar los servicios que requieran las diferentes áreas de negocio. Para ello, COBIT presenta 10 procesos:

- P01: Definición de un plan estratégico: gestión del valor, alineación con las necesidades del negocio, planes estratégicos y tácticos;
- P02: Definición de la arquitectura de información: modelo de arquitectura, diccionario de datos, clasificación de la información, gestión de la integridad;
- P03: Determinar las directrices tecnológicas: análisis de tecnologías emergentes, monitorizar tendencias y regulaciones;
- P04: Definición de procesos, organización y relaciones: análisis de los procesos, comités, estructura organizativa, responsabilidades, propietarios de la información, supervisión, segregación de funciones, políticas de contratación;

- P05: Gestión de la inversión en tecnología: gestión financiera, priorización de proyectos, presupuestos, gestión de los costos y beneficios;
- P06: Gestión de la comunicación: políticas y concienciación de usuarios;
- P07: Gestión de los recursos humanos de las tecnologías de la información: contratación, competencias del personal, roles, planes de formación, evaluación del desempeño de los empleados;
- P08: Gestión de la calidad: mejora continua, orientación al cliente, sistemas de medición y monitorización de la calidad, estándares de desarrollo y adquisición;
- P09: Validación y gestión del riesgo de las tecnologías de la información;
- P10: Gestión de proyectos: planificación, definición y asignación de recursos.

II.10.2.1 Adquisición e implementación

Con el objeto de garantizar que las adquisiciones de aplicaciones comerciales, el desarrollo de herramientas a medida y su posterior mantenimiento se encuentren alineado con las necesidades del negocio, el estándar COBIT define los siguientes 7 procesos:

AI1: Identificación de soluciones: análisis funcional y técnico, análisis del riesgo, estudio de la viabilidad.

AI2: Adquisición y mantenimiento de aplicaciones: Diseño, controles sobre la seguridad, desarrollo, configuración, verificación de la calidad, mantenimiento.

Objetivo de control: Adquisición y mantenimiento de aplicaciones software.

Requisito de negocio: Suministrar funciones automáticas que soporten de forma efectiva los procesos de negocio. [MEA13]

Se debe comprobar si existe:

- La definición de estados específicos de los requisitos funcionales y operativos;
- Una implementación estructurada con dictámenes claros;
- Las actividades de desarrollo y pruebas están separadas. Indicadores:
- Cantidad de aplicaciones entregadas puntualmente de acuerdo al plan;
- Cantidad de pedidos de cambios relacionados con defecto del sistema;

- Tiempo en que tardan en analizar y resolver problemas.

AI3: Adquisición y mantenimiento de la infraestructura tecnológica: Plan de infraestructuras, controles de protección y disponibilidad, mantenimiento.

AI4: Facilidad de uso: Formación a gerencia, usuarios, operadores y personal de soporte.

AI5: Obtención de recursos tecnológicos: control y asignación los recursos disponibles, gestión de contratos con proveedores, procedimientos de selección de proveedores.

AI6: Gestión de cambios: Procedimientos de solicitud o autorización de cambios, verificación del impacto y priorización, cambios de emergencia, seguimiento de los cambios, actualización de documentos.

AI7: Instalación y acreditación de soluciones y cambios: Formación, pruebas técnicas y de usuario, conversiones de datos, test de aceptación por el cliente, traspaso a producción. [MEA13]

II.10.2.2 Entrega y soporte

La entrega y soporte de servicios se encuentran constituidos por diversos procesos orientados a asegurar la eficacia y eficiencia de los sistemas de información.

DS1: Definición y gestión de los niveles de servicio: SLA con usuarios/clientes.

DS2: Gestión de servicios de terceros: gestión de las relaciones con proveedores, valoración del riesgo (non-disclousure agreements NDA), monitorización del servicio.

Objetivo de control: Gestionar los servicios prestados por las empresas que trabajan con las tecnologías de información y encargadas desarrollo de aplicaciones y de los usuarios dedicadas a esa área. [MEA13]

Requisito de negocio: Asegurar que las reglas y las responsabilidades de terceras partes están definidas de forma clara, adheridas y continuar satisfaciendo los requisitos.

Se tomará en consideración:

- Monitorización de contratos existentes;
- Acuerdos de servicio con terceras partes;
- Requisitos legales y regulados. Indicadores:
- Cantidad de contratos de servicio actualizados.

DS3: Gestión del rendimiento y la capacidad: planes de capacidad, monitorización del rendimiento, disponibilidad de recursos.

DS4: Asegurar la continuidad del servicio: plan de continuidad, recursos críticos, recuperación de servicios, copias de seguridad.
DS5: Garantizar la seguridad de los sistemas: gestión de identidades, gestión de usuarios, monitorización y tests de seguridad, protecciones de seguridad, prevención y corrección de software malicioso, seguridad de la red, intercambio de datos sensibles.
DS6: Identificar y asignar costos
DS7: Formación a usuarios: identificar necesidades, planes de formación.
DS8: Gestión de incidentes y Help Desk: registro y escalado de incidencias, análisis de tendencias.
DS9: Gestión de configuraciones: definición de configuraciones base, análisis de integridad de configuraciones.
DS10: Gestión de problemas: identificación y clasificación, seguimiento, integración con la gestión de incidentes y configuraciones.
DS11: Gestión de los datos: acuerdos para la retención y almacenaje de los datos, copias de seguridad, pruebas de recuperación.
DS12: Gestión del entorno físico: acceso físico, medidas de seguridad, medidas de protección medioambientales.
DS13: Gestión de las operaciones: planificación de tareas, mantenimiento preventivo.

II.10.2.3 Supervisión y evaluación

Se tiene lo siguiente:

- Garantizar la alineación con la estratégica del negocio;
- Verificar las desviaciones en base a los acuerdos del nivel de servicio;
- Validar el cumplimiento regulatorio;
- Esta supervisión implica paralelamente la verificación de los controles por parte de auditores (internos o externos), ofreciendo una visión objetiva de la situación y con independencia del responsable del proceso.

ME1: Monitorización y evaluación del rendimiento ME2: Monitorización y evaluación del control interno
ME3: Asegurar el cumplimiento con requerimientos externos ME4: Buen gobierno. [MEA13]

II.11 Modelo 4+1 vistas

Kruchten propone el modelado de arquitecturas utilizando cuatro diferentes vistas y una vista de casos de uso para ilustrar y validar las otras vistas. Cada vista aborda un enfoque específico de la arquitectura para un conjunto particular de actores. En 4+1 View Model of Architecture en figura 2-7, Kruchten define las siguientes vistas: [MOR14]

- La vista lógica, representa los requerimientos funcionales del sistema, usando abstracciones elaboradas desde el dominio del problema. Esta vista es utilizada por el usuario final para asegurarse que todos los requerimientos funcionales han sido considerados en la implementación del sistema.
- La vista de procesos, describe los mecanismos de concurrencia y sincronización usados en el sistema. Esta vista es utilizada por los integradores del sistema para el análisis del rendimiento y escalabilidad del sistema.
- La vista de desarrollo, muestra el diseño del código en el ambiente de desarrollo. Esta vista es utilizada por los líderes de proyecto y programadores, y tiene como fin ayudar en la planeación y la evaluación del progreso del proyecto.
- La vista física, describe el mapeo del software en el hardware y refleja los aspectos de la distribución. Los Ingenieros en Sistemas desarrollan esta vista para determinar la topología y los requerimientos de comunicación entre los distintos componentes.
- Los escenarios son el termino +1 dentro del 4+1. Los escenarios ilustran las distintas decisiones que son tomadas a lo largo de las cuatro vistas. En los escenarios se capturan las características generales del sistema, como se observa en la figura 10. [MOR14]

Figura 10.Figura 2-7: Modelo 4+1 vistas de la arquitectura

Fuente: [MOR14]

Cada una de las vistas describe de forma general al sistema desde una perspectiva particular. Cada una de las vistas tiene una notación particular de representación. En nuestro caso usaremos el Lenguaje de Modelado Unificado para describir las vistas de forma ejemplificada.

En la Tabla 8 se muestra el mapeo de las vistas, propuestas por Kruchten, de la Arquitectura a diagramas UML. [MOR14]

4+1 Vistas	Arquitectura	Diagramas UML
Vista lógica	Lógica	Clases, de estados y colaboración
Vista de procesos	Procesos	Actividad, estados y de secuencia
Vista de desarrollo	Implementación y datos	Componentes
Vista física	Despliegue	Despliegue
Escenarios	Requerimientos	Casos de uso

Tabla 8: ***Mapeo de vistas a diagramas UML***

II.12 Lenguaje de modelado unificado (UML)

Uno de los principales problemas en el desarrollo de software es la especificación de la arquitectura. Sin embargo, no se ha resuelto del todo el problema debido a que no existe un

estándar para este tipo de leguajes. Por lo cual el problema que ahora surge es cómo representar la arquitectura de un sistema de software, de tal manera que sea entendible por la mayoría de las personas que están involucradas en el desarrollo. [DEVA14]

Para hacer frente a este problema se propone el uso del lenguaje de modelado unificado (UML) para la representación de las arquitecturas. UML es un lenguaje estándar que nos permite modelar los componentes de un sistema de forma que sea entendible para todos los integrantes del equipo de desarrollo. [DEVA14]

El trabajo de Philippe Kruchten sobre el modelo de 4+1 vistas (Ver acápite 2.11.) (the 4+1 model view) para describir la arquitectura de software nos ayuda a analizar de una manera más comprensiva. Este enfoque utiliza diferentes vistas para separar los conceptos de cada Stakeholder. El enfoque de 4+1 vistas ha sido ampliamente usado por la comunidad de la industria de software para representar el diseño de la arquitectura de software de las aplicaciones.

En la figura 11 se muestra los diagramas de UML que corresponden a cada vista del modelo de 4+1 vistas.

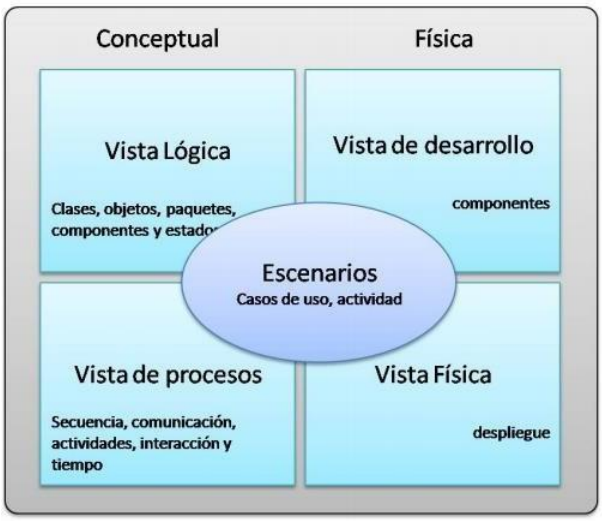


Figura 11. Modelo 4+1 vistas con UML

Fuente: [DEVA14]

UML es un lenguaje de modelado que utiliza conceptos orientados a objetos y tiene una sintaxis y semántica bien definidas lo que nos permite utilizarlo en todas las etapas de desarrollo y no solo en el proceso de diseño arquitectónico. De acuerdo con la especificación

de UML 2 se encuentra divido en 13 tipos de diagramas básicos dentro de dos categorías clave:

- Diagramas estructurales: estos diagramas son utilizados para definir la arquitectura estática. Estos comprenden la construcción estática como las clases, objetos y componentes, y la relación entre estos elementos. Existen seis diagramas estructurales: Diagrama de paquetes, diagrama de clases, diagrama de objetos, diagrama de componentes, diagrama de despliegue y diagrama de estructura compuesta.
- Diagramas de comportamiento: estos diagramas son usados para representar la arquitectura dinámica. Estos comprenden la construcción dinámica como actividades, estados, líneas de tiempo, y los mensajes que ocurren entre diferentes objetos. Estos diagramas son usados para representar la interacción entre varios elementos del modelo y estados instantáneos sobre un periodo de tiempo. Existen siete diagramas de comportamiento: Diagramas de casos de uso, diagramas de actividades, diagramas de estados, diagramas de comunicación, diagramas de secuencia, diagramas de tiempo y diagramas de interacción.

La organización fundamental de un sistema de software puede ser representada por:

- Elementos: estructurales y sus interfaces que comprenden o forman un sistema.
- Comportamiento: representado por la colaboración entre los elementos estructurales.
- Composición: de elementos estructurales y de comportamiento dentro de los grandes sistemas.

Tales composiciones son guiadas por las capacidades deseadas (requerimientos no funcionales) como la usabilidad, resistencia, rendimiento, reusabilidad, limitaciones económicas y tecnológicas etc. Además, existen temas transversales (como la seguridad y el manejo de transacciones) que aplican para todos los elementos funcionales.

Cada uno de los elementos y relaciones tiene una representación gráfica que, a su vez, se puede complementar con la especificación del elemento o relación; la especificación no es visible del todo ya que corresponde a los datos o propiedades adicionales que complementan la semántica del elemento o relación.

Por lo tanto, necesitamos múltiples puntos de vista para las distintas necesidades de los actores involucrados en el desarrollo de un sistema de software.

II.13 Modelo de seguridad

Un modelo de seguridad es la presentación formal de una política de seguridad. El modelo debe identificar el conjunto de reglas y prácticas que regulan cómo un sistema maneja, protege y distribuye información delicada. [LORE06]

Los modelos se clasifican en:

- Modelo abstracto: se ocupa de las entidades abstractas como sujetos y objetos.
- Modelo concreto: traduce las entidades abstractas en entidades de un sistema real como procesos y archivos.

Además, los modelos sirven a tres propósitos en la seguridad informática:

- Proveer un sistema que ayude a comprender los diferentes conceptos. Los modelos diseñados para este propósito usan diagramas, analogías u otros aspectos que puedan ayudar a un mejor entendimiento del objeto de estudio.
- Proveer una representación de una política general de seguridad formal clara.
- Expresar la política exigida por un sistema de cómputo específico.

II.14 Metodologías de desarrollo

Una metodología es aquella guía que se sigue a fin realizar las acciones propias de una investigación. En términos más sencillos se trata de la guía que nos va indicando qué hacer y cómo actuar cuando se quiere obtener algún tipo de investigación. Es posible definir una metodología como aquel enfoque que permite observar un problema de una forma total, sistemática, disciplinada y con cierta disciplina.

Al intentar comprender la definición que se hace de lo que es una metodología, resulta de suma importancia tener en cuenta que una metodología no es lo mismo que la técnica de investigación. Las técnicas son parte de una metodología, y se define como aquellos procedimientos que se utilizan para llevar a cabo la metodología, por lo tanto, como es posible intuir, es uno de los muchos elementos que incluye.

II.14.1 RUP (Proceso Unificado de Racional)

Es un proceso de ingeniería de software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta y de mayor calidad para satisfacer las necesidades de los

usuarios que tienen un cumplimiento al final dentro de un límite de tiempo y presupuesto previsible. Es una metodología de desarrollo iterativo que es enfocada hacia "diagramas de los casos de uso, y manejo de los riesgos y el manejo de la arquitectura" como tal.
El RUP mejora la productividad del equipo ya que permite que cada miembro del grupo sin importar su responsabilidad específica pueda acceder a la misma base de datos incluyendo sus conocimientos. Esto hace que todos compartan el mismo lenguaje, la misma visión y el mismo proceso acerca de cómo desarrollar un software.

II.14.2 XP (Programación Extrema)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes.

II.14.3 MSF (Microsoft Solution Framework)

Es una metodología realizada por Microsoft que puede aplicarse a otros proyectos de tecnologías de información y no solo al desarrollo de software, es flexible, se basa en modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

II.14.4 SCRUM

Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.
En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado

para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

II.15 Tecnologías de desarrollo

Algunas de las tecnologías de desarrollo para aplicaciones web. Ver figura 2-9

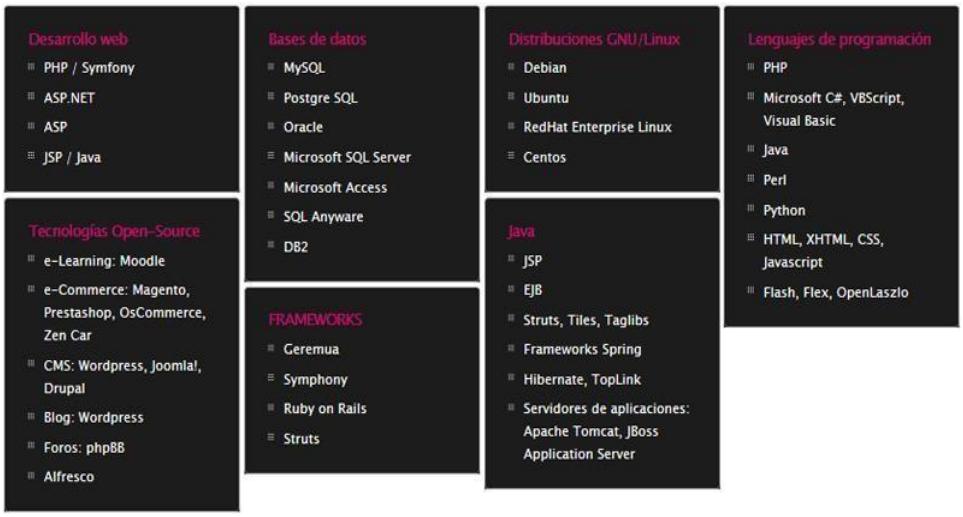


Figura 12. Tecnologías más usados para el desarrollo web

CAPITULO III

ESTUDIO DE VULNERABILIDADES EN APLICACIONES WEB

III. CAPITULO III ESTUDIO DE VULNERABILIDADES EN APLICACIONES WEB

Componente I: Realizar un estudio de vulnerabilidades para identificar las principales vulnerabilidades que afectan a las aplicaciones web durante la etapa de desarrollo.

III.1 Introducción

En el presente capitulo citaremos algunas de las principales vulnerabilidades que afectan a las aplicaciones web durante su desarrollo, para cada vulnerabilidad se establecerá algunas recomendaciones para evitar dichas vulnerabilidades, también se utilizará algunos ejemplos para una mejor comprensión.

III.2 Propósito

El propósito de este componente es brindar recomendaciones para evitar las vulnerabilidades más comunes que afectan a las aplicaciones web durante la etapa de desarrollo.

III.3 Objetivo general

Identificar las vulnerabilidades más comunes que afectan a las aplicaciones web durante la etapa de desarrollo.

III.4 Objetivos específicos

- Mostrar ejemplos de vulnerabilidades en las aplicaciones web.
- Citar recomendaciones para evitar ataques a las aplicaciones web.

III.5 Web.

Las aplicaciones web pueden presentar diversas vulnerabilidades que van de acuerdo a los servicios que prestan. De acuerdo con la OWASP (Open Web Application Security Project), las vulnerabilidades en aplicaciones web más explotadas a finales del año 2019, fueron las siguientes:

- Cross Site Scripting (XSS).
- Ataques de inyección de código.

- Ejecución de archivos maliciosos.
- Insecure Direct Object Reference.
- Ataques Cross Site Request Forgery (CSRF).
- Pérdidas de información y errores al procesar mensajes de error.
- Robo de identidad de autentificación.
- Almacenamiento criptográfico inseguro.
- Comunicaciones inseguras.
- Acceso a URLs ocultas no restringidas de manera adecuada.

El sitio www.opensecurity.es indica que los 5 principales tipos de vulnerabilidades en aplicaciones web son:

- Ejecución remota de código.
- SQL.
- Vulnerabilidades en formato de cadenas.
- Cross Site Scripting (XSS).
- Problemas atribuidos a los usuarios.

Otro sitio (www.vsantivirus.com) publica que las vulnerabilidades más comunes son:

- SQL Injection.
- Cross Site Scripting (XSS).

Podemos apreciar que, en las anteriores listas, la vulnerabilidad en aplicación web en común es el ataque Cross Site Scripting (XSS), por lo que se debe poner atención en ella, así como las que se explicarán más adelante.

Para este trabajo, se analizaron las vulnerabilidades en aplicaciones web más comunes porque pueden representar un peligro para los servidores web y de bases de datos.

III.5.1 XSS (Cross Site Scripting).

El XSS es un fallo de seguridad en sistemas de información basados en web, que más que comprometer la seguridad del servidor web compromete la seguridad del cliente.

El XSS es un ataque, que consiste en inyectar código, HTML y/o JavaScript en una aplicación web, con el objetivo de que el cliente ejecute el código inyectado al momento de ejecutar la aplicación.

El XSS se da cuando una aplicación web permite inyectar código en la página; esto se puede lograr por medio de campos de texto de formularios o por medio de la URL.

Por lo regular, el código inyectado se ejecuta de manera que el cliente no nota algún comportamiento fuera de lo normal, ya que la ejecución de este código se hace de manera simultánea con el código original de la aplicación. Dependiendo de otros factores, el XSS puede hacer que el navegador funcione de manera indebida y en algunos casos, llegar a provocar un fallo en el servidor. Aunque esto último es muy difícil porque el código HTML y JavaScript se ejecutan en el navegador y no en el servidor web.

Para poder evitar este tipo de ataque, en las aplicaciones web se debe:

- Evitar que llegue código HTML y/o JavaScript por medio del método Get o Post, es decir, hay que filtrar el código. Algunos lenguajes de programación del lado del servidor proveen funciones para evitar que el código HTML llegue como tal a nuestras aplicaciones. Un ejemplo de esto es PHP que cuenta con funciones que convierten el código HTML en texto o simplemente lo suprimen, tal es el caso de las funciones htmlspecialchars() y htmlentities().
- Procurar que los datos viajen por Post en lugar de Get para evitar ataques de XSS por URL.
- También es recomendable hacer uso de navegadores modernos, ya que algunos tienen la capacidad de detectar casos en donde se podría presentar la vulnerabilidad y lo invalidan.

Aunque algunos expertos en seguridad informática opinan que las vulnerabilidades a XSS son ya obsoletas, otros opinan lo contrario, pues el XSS es muy utilizado para realizar ataques como el *phishing*, robo de identidad y otro tipo de ataques, por lo que es importante conocer cómo funciona para poder evitarlo.

III.5.2 CSRF (Cross Site Request Forgery).

El CSRF es prácticamente igual que XSS, salvo que este ataque se basa en explotar la confianza que un usuario tiene en un sitio web.

Se trata de una vulnerabilidad en aplicaciones web, en donde el usuario es forzado a ejecutar acciones no deseadas en una aplicación web en la cual se encuentra autenticado. Con un poco de ayuda de ingeniería social (como el envío de una liga vía correo electrónico o chat), una persona malintencionada podría forzar al usuario de la aplicación web a ejecutar acciones no deseadas por el mismo usuario, por ejemplo, la ejecución de código de manera remota.

De ser exitoso, el CSRF puede comprometer la información del usuario y el comportamiento normal del usuario en la aplicación web. En caso de que el usuario sea el administrador de la aplicación web, este tipo de vulnerabilidad puede llegar a comprometer por completo al sistema en cuestión.

Existen numerosas formas en las que el usuario puede ser engañado cuando intercambia información con un sitio web. Con el fin de llevar a cabo un ataque de este tipo, primero se debe saber cómo generar una petición maliciosa para que nuestra víctima la ejecute.

Ejemplo:

El usuario B desea hacer una transferencia bancaria de $1,000 al usuario A, a través del portal de Internet de un banco (http://banco.com), la cabecera http de la petición al servidor generada por el usuario B podría ser de la siguiente manera:

```
GET http://banco.com/transfer.do HTTP/1.1
...
...
...
Content-Length: 19;

acct=USUARIOA&amount=1000
```

Pero el usuario C nota que la aplicación web realiza la transferencia de parámetros de la URL de la siguiente manera:

```
GET http://banco.com/transfer.do?acct=USUARIOA&amount=1000 HTTP/1.1
```

Y como consecuencia de lo anterior, el usuario C decide explotar esta vulnerabilidad en la aplicación web del banco tomando como su víctima al usuario B. De esta forma, el usuario C construye una URL que transferirá $100,000 de la cuenta del usuario B a su propia cuenta, como sigue:

```
http://banco.com/transfer.do?acct=USUARIOC&amount=100000
```

Ahora que el código de la petición maliciosa se ha generado, el usuario C debe engañar al usuario B para que este envíe la petición al servidor del banco, ya que el usuario B está autenticado en el sistema. El método más sencillo sería que el usuario C le envíe al usuario B el link con la petición y que este lo abra, ya sea por correo electrónico o vía chat. La etiqueta HTML sería la siguiente:

```
<a href="http://banco.com/transfer.do?acct=USUARIOC&amount=100000">Gane
$10,000.00 Ahora</a>
```

Asumiendo que el usuario B se encuentra autenticado (ha iniciado sesión) en la aplicación web del banco, cuando da clic al link que le envió al usuario C, él inconscientemente transfiere a la cuenta del usuario C $100,000. Para que el usuario B no lo note (ya que el banco le notificará de la transferencia), el usuario C decide esconder el ataque en una imagen de cero bytes, como sigue:

```
<img src="http://banco.com/transfer.do?acct=USUARIOC&amount=100000" width="1"
height="1" border="0">
```

Si esta etiqueta HTML de la imagen, fuese incluida en un correo electrónico, el usuario B sólo vería una pequeña caja indicando que el navegador no puede mostrar la imagen. Sin embargo, el navegador, en cualquier caso, enviará la petición al sistema del banco sin ninguna indicación de que la transferencia se ha llevado a cabo.

Este tipo de ataque se puede prestar a fraudes como *phishing*.

Para evitar este tipo de vulnerabilidad en aplicaciones web, por el lado del programador, se debe:

- Hacer que las sesiones expiren en un tiempo corto. este tiempo tiene que ser el suficiente para que el usuario haga la transacción que requiere.
- Forzar a que el usuario termine su sesión para que de esta forma se evite que la sesión del usuario quede activa y se pueda hacer mal uso de ella.
- Hacer del conocimiento del usuarioque los problemas mencionados anteriormente se pueden presentar y concientizarlo de cómo prevenirlos.
- Ocultar la URL en navegadores y códigos fuente de aplicaciones para evitar su mal uso.
- Hacer un filtrado de datos que llegan al servidor, es decir, verificar que los tipos de datos sean los que se esperan.
- Hacer que la composición de la URL sea compleja, es decir, cifrar los datos que viajan a través de esta.

Para evitar este tipo de vulnerabilidad en aplicaciones web, por parte del usuario, se debe:

- Hacer el intercambio de la información a través de internet en un lugar seguro y no sitios públicos como escuelas, lugares de trabajo o cibercafés.
- Proteger el equipo con software *antiphishing*, sobre todo para personas que no tienen muchos conocimientos de informática.
- Advertir al usuario de no abrir ligas o correos electrónicos sospechosos o de dudosa procedencia.

III.5.3 Inyección de Código (Code Injection).

Code Injection es el nombre de un ataque, que consiste en insertar código que podría ser ejecutado por una aplicación. Un ejemplo de esto, es cuando se añade una cadena de caracteres en una cookie o los valores de un argumento en la URL. Este tipo de ataque hace uso de la falta de una validación correcta de los datos: tipos de caracteres permitidos, formato de datos, datos esperados, etcétera.

Code Injection y Command Injection, son ataques muy similares entre sí, por lo que no analizaremos a fondo este último, pero si diremos que la diferencia entre Code Injection y

Command Injection son las medidas distintas que se toman para lograr objetivos similares. Mientras que Code Injection tiene como objetivo añadir código malicioso en una aplicación, que luego se ejecutará como parte de ésta, Command Injection no es precisamente código que pertenece a la aplicación y no necesariamente se ejecutará simultáneamente con ésta. Este tipo de vulnerabilidad en el código, puede ser muchas veces peor que cualquier otra vulnerabilidad, ya que la seguridad del sitio web y posiblemente del servidor, se ve comprometida.

Un ejemplo muy sencillo de cómo opera este tipo de ataque es el siguiente:

Hagamos la suposición de que el siguiente código en PHP se va a ejecutar:

```
<html>

<body>

<?php

$pagina=$_GET[‘page’];
Include(‘$pagina’);
?>

</body>

</html>
```

Si en un servidor externo tenemos un script, es posible realizar un ataque con Code Injection que el código anterior presenta. Digamos que la URL que corresponde al código de la aplicación de arriba es la siguiente: http://ejemplo.net/index.php y que el script con el código que se inyectará está en el servidor cuya URL es http://codigo.net/code.php, entonces basta con construir una URL como sigue para realizar el ataque y llamarla desde el navegador:

```
http://ejemplo.net/index.php?page=http://codigo.net/code.php
```

Con esto se ejecutará todo el código que el atacante haya escrito en el archivo “code.php”.

Para evitar este tipo de ataques, basta con hacer una programación ordenada, con el filtrado del código y con la inicialización de todas las variables. El ejemplo aquí presentado es el más

utilizado para realizar este tipo de ataques, por lo que se debe evitar a toda costa incluir archivos por medio de variables.

III.5.4 Buffer Overflow.

Este tipo de vulnerabilidad, es quizá, la más conocida dentro de la seguridad en el software. La mayor parte de los desarrolladores de software saben lo que una vulnerabilidad del tipo Buffer Overflow es, sin embargo, este tipo de vulnerabilidad suele ser aún común hoy en día. Parte del problema se debe a la gran variedad de formas en las cuales el Buffer Overflow se puede dar.

Este tipo de vulnerabilidades no son tan fáciles de descubrir y cuando se llegan a descubrir, son por lo general, difíciles de explotar. A pesar de lo anterior, los atacantes han logrado identificar vulnerabilidades de este tipo en un sin fin de sistemas de todo tipo.

Es clásico que, en esta vulnerabilidad, el hacker envíe datos a un programa, los cuales, son almacenados en una pila de tamaño inferior al de los datos. El resultado de esto es que la información en la pila es sobrescrita incluyendo las funciones de punto de retorno. Los datos establecen el valor del punto de retorno, así que, si la función regresa, transfiere el control al código malicioso contenido en los datos del atacante.

Aunque este tipo de Buffer Overflow es aún común en algunas plataformas y en algunas comunidades de desarrollo, existen otras variedades de tipos de Buffer Overflow. Otra clase de vulnerabilidad muy similar y conocida es la llamada Format String Attack. Existe un gran número de información acerca de esta vulnerabilidad en Internet y en libros, que provee muchos detalles de cómo trabaja esta vulnerabilidad, por lo que sólo la mencionamos, por ser conocida al igual que las vulnerabilidades Heap Buffer Overflow y Off-By-One Error, que entran en esta categoría.

A nivel de código las vulnerabilidades de tipo Buffer Overflow envuelven comúnmente la violación a las sentencias del programador. Muchas funciones de manipulación de memoria en lenguajes como C/C++ y sus derivados no ejecutan chequeos de límites y es posible sobrescribir fácilmente los límites asignados para su operación. La combinación de la manipulación de memoria y sentencias equivocadas referentes al tamaño son la principal causa de este tipo de vulnerabilidades.

A nivel de código, una vulnerabilidad de este tipo ocurre cuando una aplicación se basa en datos externos para controlar su comportamiento o depende de las propiedades de los datos que se ejecutan fuera del ámbito inmediato del código, o bien si es tan compleja que un programador no puede predecir con exactitud su comportamiento.

En las aplicaciones web, los atacantes explotan vulnerabilidades de este tipo para corromper la pila de ejecución de las aplicaciones web. Al enviar cuidadosamente datos de entrada a una aplicación web, el atacante puede causar que dicha aplicación ejecute código de una manera arbitraria y así hacerse de una manera efectiva del control del sistema.

Las vulnerabilidades Buffer Overflow pueden estar presentes tanto en el servidor web como en el servidor de aplicaciones, que da soporte de contenido estático o dinámico al sitio o a la aplicación web y pueden plantear un riesgo significativo para los usuarios del servidor web. Cuando las aplicaciones web hacen uso de librerías, se abre la posibilidad a ataques de Buffer Overflow.

La vulnerabilidad también puede encontrarse a nivel de código en las aplicaciones web y suele ser más probable dada la falta de control que se tiene en estas aplicaciones y lo difícil que resulta detectar dichas vulnerabilidades. Aunque son más probables las vulnerabilidades de este tipo en este caso, ya habíamos mencionado que son mucho más difíciles de encontrar y por lo tanto el número de atacantes que intentará encontrar y explotar estas vulnerabilidades será menor.

Casi todos los servidores web, servidores de aplicaciones y entornos de aplicaciones Web son susceptibles a vulnerabilidades de Buffer Overflow, sin embargo, se tiene una notable excepción en los entornos desarrollados con lenguajes como Java, que son inmunes a este tipo de ataques (a excepción de desbordamientos en el intérprete mismo), ya que Java tiene una máquina virtual que cuenta con ciertos niveles de seguridad que impiden que esta vulnerabilidad se presente.

Para terminar con esta vulnerabilidad, revisaremos un ejemplo de esta para comprender su funcionamiento.

El siguiente fragmento de código de ejemplo (en lenguaje C) demuestra que existe una vulnerabilidad de Buffer Overflow que se da a causa de que el código se basa en datos

externos para controlar su comportamiento. El código hace uso de la función gets(), que quienes tienen algunos conocimientos del lenguaje de programación C/C++ sabrán que sirve

```
char buf[BUFSIZE];
cin >> (buf);
```

para leer una cantidad arbitraria de datos y guardarlos en una pila. Como no hay forma de limitar la cantidad de datos que esta función lee, la seguridad del código depende de que el usuario siempre ingrese una cantidad menor a la capacidad en tamaño de la pila.

Para prevenir este tipo de vulnerabilidades es importante estar informado y al día con los últimos reportes de fallos para nuestro servidor web, para nuestras aplicaciones y para otros productos de nuestra infraestructura de Internet. Realizar una programación segura de nuestras aplicaciones es también muy importante. Hay que aplicar siempre los parches y las actualizaciones de seguridad de nuestros productos de software que por lo regular se encuentran disponibles en la web de nuestros proveedores, pero siempre hay que estar informado si son seguros o no. También es importante escanear y monitorear de manera periódica nuestros servidores y nuestras aplicaciones con las diversas herramientas existentes para la búsqueda de estos fallos de seguridad y en el caso de las aplicaciones, revisar todo el código que acepta datos de usuarios a través de peticiones HTTP y asegurarse que se dispone del tamaño adecuado de buffer para almacenarlos. Esto debe hacerse incluso en el caso de entornos que no son sensibles a este tipo de vulnerabilidades.

III.6 Bases de datos.

Las bases de datos son vulnerables si no se tiene una buena configuración de seguridad y están propensas a experimentar el ataque llamado SQL Injection si no se tiene una buena validación en la aplicación con la que interactúa, por lo cual en este subcapítulo únicamente hablaremos acerca de este ataque que explota vulnerabilidades de una programación deficiente en los sistemas.

III.6.1 SQL Injection.

Un ataque del tipo SQL Injection consiste en insertar o inyectar una consulta SQL a través del intercambio de datos entre el cliente y la aplicación. Un ataque de SQL Injection, es capaz de leer datos sensibles de la base de datos, modificar los datos de dicha base de datos (Insert, Delete, Update), ejecutar operaciones como administrador, recuperar el contenido de un archivo dado que se encuentra en el Sistema de directorios del Sistema Manejador de Bases de Datos (DBMS) y en algunos casos ejecutar comandos en el sistema operativo. Los ataques de SQL Injection son del tipo de ataques de inyección (como Code Injection y Command Injection).

Como ejemplo de SQL Injection tenemos el siguiente: supongamos que tenemos un formulario de autenticación de usuarios que requiere un nombre de usuario y una contraseña. Veamos la siguiente consulta, haciendo uso de PHP:

```
SELECT * FROM usuarios WHERE usuario=$_POST['usuario'] AND
contrasena=$_POST['contrasena'];
```

Ahora bien, si en el formulario de autenticación introducimos como usuario tom y como contraseña ‘’ OR 1 = 1, de tal forma que la consulta sería:

```
SELECT * FROM usuarios WHERE usuario= 'tom' && contrasena = ‘’ OR 1 = 1;
```

Con la consulta anterior, cualquier usuario quedaría autenticado, porque esta consulta siempre regresaría algo diferente de nulo.

Para evitar que nuestras aplicaciones sean vulnerables a un ataque de SQL Injection, nunca debemos confiar en la información que el usuario introduce en los formularios, toda esa información debe ser verificada y validada, se recomienda también no construir consultas de SQL de forma dinámica, ya que son susceptibles a ser cambiadas de forma externa, también hay que evitar el uso de cuentas con privilegios administrativos, al igual que se debe evitar proporcionar información innecesaria (mensajes como “contraseña incorrecta” o “usuario incorrecto” no deben ser utilizados, mejor utilizar “usuario y/o contraseña incorrectos”).

El conocer las vulnerabilidades en las aplicaciones web es de suma importancia por el riesgo que implica al servidor si no se encuentra protegido correctamente.

N°	VULNERABILIDAD	ATAQUE	DESCRIPCIÓN	RECOMENDACIÓN	EJEMPLOS
1	**XSS (Cross Site Scripting).**	Inyectar código HTML y/o JavaScript.	El XSS es un ataque, que consiste en inyectar código, HTML y/o JavaScript en una aplicación web, con el objetivo de que el cliente ejecute el código inyectado al momento de ejecutar la aplicación.	Evitar que llegue código HTML y/o JavaScript por medio del método Get o Post. Procurar que los datos viajen por Post en lugar de Get para evitar ataques de XSS por URL. También es recomendable hacer uso de navegadores modernos, ya que algunos tienen la capacidad de detectar casos en donde se podría presentar la vulnerabilidad y lo invalidan.	http://example.com/search.php?query=<script>alert("hackeado")</script>

2	**CSRF (Cross Site Request Forgery).**	Ataque se basa en explotar la confianza que un usuario tiene en un sitio web.	Se trata de una vulnerabilidad en aplicaciones web, en donde el usuario es forzado a ejecutar acciones no deseadas en una aplicación web en la cual se encuentra autenticado.	• Hacer que las sesiones expiren en un tiempo corto. este tiempo tiene que ser el suficiente para que el usuario haga la transacción que requiere. • Forzar a que el usuario termine su sesión para que de esta forma se evite que la sesión del usuario quede activa y se pueda hacer mal uso de ella. • Hacer del conocimiento del usuario que los problemas mencionados anteriormente se pueden presentar y concientizarlo de cómo prevenirlos. • Ocultar la URL en navegadores y códigos fuente de aplicaciones para evitar su mal uso. • Hacer un filtrado de datos que llegan al servidor, es decir, verificar que los tipos de datos sean los que se esperan. • Hacer que la composición de la URL sea compleja, es decir, cifrar los datos que viajan a través de esta.	Gane $10,000.00 Ahora

3	**Inyección de Código (Code Injection).**	Ataque que consiste en insertar código que podría ser ejecutado por una aplicación.	Este tipo de ataque hace uso de la falta de una validación correcta de los datos: tipos de caracteres permitidos, formato de datos, datos esperados, etcétera.	Validar cada campo de texto que el usuario de una aplicación ingresa. Hacer una programación ordenada, con el filtrado del código y con la inicialización de todas las variables.	http://ejemplo.net/index.php?page=http://codigo.net/code.php

4	**SQL Injection.**	Ataque del tipo SQL	Consiste en insertar o inyectar una consulta SQL a través del intercambio de datos entre el cliente y la aplicación.	Nunca debemos confiar en la información que el usuario introduce en los formularios, toda esa información debe ser verificada y validada, se recomienda también no construir consultas de SQL de forma dinámica, ya que son susceptibles a ser cambiadas de forma externa, también hay que evitar el uso de cuentas con privilegios administrativos, al igual que se debe evitar proporcionar información innecesaria (mensajes como "contraseña incorrecta" o "usuario incorrecto" no deben ser utilizados, mejor utilizar "usuario y/o contraseña incorrectos").	Como ejemplo de SQL Injection tenemos el siguiente: supongamos que tenemos un formulario de autenticación de usuarios que requiere un nombre de usuario y una contraseña. Veamos la siguiente consulta, haciendo uso de PHP: SELECT * FROM usuarios WHERE usuario=$_POST['usuario'] AND contrasena=$_POST['contrasena']; Ahora bien, si en el formulario de autenticación introducimos como usuario tom y como contraseña '' OR 1 = 1, de tal forma que la consulta sería: SELECT * FROM usuarios WHERE usuario= 'tom' && contrasena = '' OR 1 = 1; Con la consulta anterior, cualquier usuario quedaría autenticado, porque esta

					consulta siempre regresaría algo diferente de nulo.

5	**Buffer Overflow.**	Este tipo de vulnerabilidad, es quizá, la más conocida dentro de la seguridad en el software	El hacker envía datos a un programa, los cuales, son almacenados en una pila de tamaño inferior al de los datos. El resultado de esto es que la información en la pila es sobrescrita incluyendo las funciones de punto de retorno.	Para prevenir este tipo de vulnerabilidades es importante estar informado y al día con los últimos reportes de fallos para nuestro servidor web, para nuestras aplicaciones y para otros productos de nuestra infraestructura de Internet. Realizar una programación segura de nuestras aplicaciones es también muy importante. Hay que aplicar siempre los parches y las actualizaciones de seguridad de nuestros productos de software que por lo regular se encuentran disponibles en la web de nuestros proveedores.	El siguiente fragmento de código de ejemplo (en lenguaje C) demuestra que existe una vulnerabilidad de Buffer Overflow que se da a causa de que el código se basa en datos externos para controlar su comportamiento char buf[BUFSIZE]; cin >> (buf); El código hace uso de la función gets(), que quienes tienen algunos conocimientos del lenguaje de programación C/C++ sabrán que sirve para leer una cantidad arbitraria de datos y guardarlos en una pila. Como no hay forma de limitar la cantidad de datos que esta función lee, la seguridad del código depende de que el usuario siempre ingrese una cantidad menor a la capacidad en tamaño de la pila.

Tabla 9: ***Vulnerabilidad, ataque, descripción, recomendación y ejemplos***

CAPÍTULO IV

CASO DE ESTUDIO Y APLICACIÓN PRÁCTICA

IV. CAPÍTULO IV CASO DE ESTUDIO Y APLICACIÓN PRÁCTICA

Componente II: Realizar casos de estudio sobre las distintas vulnerabilidades que presentan las aplicaciones web durante su desarrollo, demostrando estas vulnerabilidades realizando una aplicación práctica sobre el prototipo de la aplicación web "Sistema de Gestión de Proyectos de Grado".

IV.1 Introducción

En el presente capítulo se da a conocer el caso de estudio y aplicación práctica el cual describe el uso de técnicas y herramientas para minimizar las vulnerabilidades que se presentan en el desarrollo de una aplicación web. La aplicación práctica planteada se adapta a cualquier tecnología que nos permita desarrollar una aplicación web. Proporcionando información general acerca de las características y servicios de seguridad, principalmente en 4 capas: Capa de presentación visual, Capa lógica de negocio, Capa servicios de datos web y Capa de alojamiento web. Para un mejor entendimiento usamos el modelo 4+1 vistas descritas en el capítulo II, para describir cada una de las capas.

Después de plantear el caso de estudio y la aplicación práctica en las aplicaciones web, se presentan arquitecturas de las aplicaciones web para una mejor comprensión y entendimiento, de esta manera observar las posibles vulnerabilidades que se pueden presentar en las aplicaciones web en base a la aplicación práctica que se realizara en el prototipo.

El modelo 4+1 vistas nos permiten realizar un estudio más detallado de los componentes que contempla cada una de las capas. Además, permite esquematizar con diagramas UML que nos facilita un mejor entendimiento de los componentes que interactúan en una aplicación web.

Después de observar las arquitecturas y vistas se describe las tareas y la interacción de las capas en las aplicaciones web, sugiriendo el uso adecuado de las técnicas y herramientas que permiten cubrir las vulnerabilidades de las aplicaciones web.

Además, se realiza un prototipo de una aplicación web que nos será de utilidad para el desarrollo de la aplicación práctica, para poder observar las distintas vulnerabilidades que presentan las aplicaciones web.

IV.2 Propósito

El propósito de este componente es dar a conocer y transferir conocimiento sobre las distintas vulnerabilidades que se presentan en las aplicaciones web mediante la aplicación práctica que se realizara sobre el prototipo de una aplicación web.

IV.3 Objetivo general

Identificar las distintas vulnerabilidades que presenta el prototipo de la aplicación web "Sistema de gestión de proyectos de grado".

IV.4 Objetivos específicos

- Realizar pruebas de vulnerabilidad al prototipo.
- Identificar vulnerabilidades que presenta el prototipo.

IV.5 Arquitectura de las aplicaciones web mediante capas

Para el desarrollo de la aplicación práctica vemos el procedimiento adecuado de la utilización de las técnicas y herramientas en las capas de las aplicaciones web.

Antes de realizar el desglose de las técnicas y herramientas se lista algunos complementos de OWASP referentes a la seguridad web, que permite desarrollar las aplicaciones webs más seguras en sus diferentes ámbitos.

- OWASP AntiSamy
- OWASP CAL9000
- OWASP CLASP
- OWASP Encoding
- OWASP Enterprise Security API
- OWASP Insecure Web App
- OWASP LAPSE
- OWASP Pantera Web Assessment Studio
- OWASP SQLiX

- OWASP Validation

IV.5.1 Capa de presentación visual

Es donde se despliega la interfaz de la aplicación web, y es la capa encargada de la recopilación de datos ingresados por el usuario que utiliza la aplicación web. Posteriormente se realiza un filtrado previo para comprobar que los datos a procesar son válidos y no erróneas.

Esta capa se encarga de pasar o enviar los datos filtrados a la capa lógica de negocio donde se realiza el procesado. A continuación, se detallan algunas de las técnicas y herramientas aplicados en esta capa.

IV.5.1.1 Validación de datos con JavaScript

La principal utilidad de JavaScript es la validación de formularios que permiten ingresar datos a los usuarios. Antes de enviar los datos de un formulario al servidor, se recomienda validar mediante JavaScript los datos ingresados por el usuario, para verificar la no existencia de errores, además se puede notificar de forma instantánea, sin necesidad de esperar la respuesta del servidor acortando el tiempo del procesamiento.

Notificar los errores de forma inmediata mediante JavaScript mejora la satisfacción del usuario que utiliza la aplicación web y ayuda a reducir la carga de procesamiento en el servidor.

Normalmente, la validación consiste en llamar a una función de validación que utiliza las expresiones regulares que comprueban si los datos ingresados cumplen con las restricciones impuestas por la aplicación.

IV.5.1.2 Validación AntiSamy

Es una API para asegurarse que las entradas HTML/CSS del usuario estén en cumplimiento con las reglas de la aplicación. Es decir, ayuda a asegurarse que los clientes no provean código malicioso en el HTML. El termino código malicioso en términos de aplicaciones web es generalmente relacionado solo con JavaScript. CSS, hojas de estilo en cascada son

consideradas maliciosas cuando invocan a JavaScript. Sin embargo, hay muchas situaciones donde HTML y CSS pueden ser usados de una forma maliciosa.

Por tanto, nos ayuda a reforzar los puntos débiles en un ataque de inyección de JavaScript y CSS basándose en el uso de expresiones regulares. Además, es aplicable en diferentes lenguajes de programación.

IV.5.1.3 Tecnología Ajax

Hay dos disponibilidades de Ajax, una es creando un objeto con JavaScript y la otra el de JQuery, con esto agilizamos él envió los datos bajo un formato, pero el empleo de esta tecnología puede ser de forma errónea. Por tanto, se debe tomar algunas medidas restrictivas:

- Implantar controles de seguridad como pueden ser autenticación, autorización, e incluso registro de operaciones, siempre en el lado del servidor.
- No almacenar datos sensibles o confidenciales en el lado del cliente, este hecho haría que obtenerlos por un atacante fuera algo potencialmente sencillo.
- Utilización de métodos criptográficos para transmitir datos sensibles o confidenciales entre el cliente y el servidor.
- Se recomienda usar Ajax en Angular.
- Toda petición realizada con AJAX y que acceda a recursos protegidos deberán encontrarse autenticadas.
- Utilizar métodos POST en vez de métodos GET.
- La lógica de negocio debe tener el principio de mínima exposición y encontrarse siempre en el lado del servidor.
- Ajax solo debe ser usado para enviar datos al servidor, para la validación u otro tipo de proceso.
- Utilizar el método adecuado para renderizar los tipos de datos recibidos del servidor por ejemplo JSON.
- No utilizar un Servidor Proxy para llamar a otros sistemas externos como servicios Web.

IV.5.1.4 Utilización del algoritmo de cifrado

Son algoritmos que se encargan de cifrar y descifrar datos en la aplicación web, para la utilización es necesario asignar una clave secreta para el cifrado tanto como para descifrado. La clave secreta se debe conocer en la capa de presentación visual y en la capa lógica de negocio.

Para aplicar un algoritmo de cifrado determinado junto con una clave, a una determinada información que se quiere transmitir confidencialmente, es necesario definir qué tipo de criptografía a utilizar, ya sea simétrica y asimétrica.

Algunos de los algoritmos comunes son MD5, SHA, SHA1, existe otros algoritmos que se pueden emplear. Pero lo más recomendable es utilizar un algoritmo propio y personalizado.

IV.5.1.5 Control de actividades

Es una técnica utilizada para el seguimiento de las acciones realizadas durante el uso de la aplicación web, consiste en generar una pila de control del lado del servidor para identificar al usuario autenticado.

Cada acción que el usuario realiza se valida del lado del servidor comparando la pila generado con anterioridad, si los datos son incorrectos se invalida la petición y la autenticación se anula.

IV.5.2 Capa lógica de negocio

Es la capa encargada de todo tipo de lógica de negocio o procesado de datos, es donde se ejecuta los procesos en su mayoría referentes a los datos recibidos de la capa de presentación visual. Las peticiones del usuario son recibidas y se envían las respectivas respuestas tras el proceso lógico. Es por lo que se denomina capa de negocio, porque es donde se establecen las reglas que deben cumplirse.

Inicialmente la capa lógica de negocio se comunica con la capa de presentación visual, para recibir las peticiones del usuario y presentar posteriormente los resultados. También se comunica con la capa servicios de datos web, para la interacción con uno o más gestores de bases de datos, con propósito de almacenar o recuperar los datos necesarios. A continuación, se detallan algunas de las técnicas y herramientas aplicados en esta capa.

IV.5.2.1 Validación de datos en el servidor

No es suficiente realizar la validación de datos en la capa de presentación visual es con objeto de disminuir la cantidad de peticiones y además pueden ser interceptados con facilidad y manipulados, lo cual lleva a utilizar un conjunto de librerías para validar los datos en el servidor, siempre será necesario realizar las validaciones de las entradas de formularios u otro tipo de entradas de datos.

Las validaciones generalmente son de verificar números, cadenas de textos, etc., utilizando expresiones regulares.

IV.5.2.2 Limitación de intentos durante la autenticación

Es una técnica utilizada para la disminución de ataque de fuerza bruta, es decir, cuando el usuario no autorizado intenta acceder a la aplicación web sin contar con los datos correctos, por ende, es necesario limitar el número de intentos para acceder.

Esto permite anular los intentos continuos para acceder a la aplicación web, disminuyendo la probabilidad de éxito de este tipo de ataques, por ejemplo, si el usuario falla tres o cinco veces en ingresar los datos de autenticación serán bloqueados automáticamente. Incluso es necesario tener registros de los intentos fallidos, cuyos registros pueden ser el identificador de usuario y la identificación de IP de la máquina.

IV.5.2.3 Cifrado de datos en servidor

Los algoritmos de cifrado como tal se pueden aplicar en ambos casos: cliente y servidor, pero siempre es recomendable realizarlo en el servidor, porque estas no pueden ser interceptados.

Es necesario cifrar los datos confidenciales ya sea de manera síncrona o asíncrona según al requerimiento con las claves respectivas.

IV.5.2.4 Transacción de los datos

Para garantizar la transferencia de datos entre la capa lógica de negocio y la capa servicio de datos web, es necesario contar con librerías que permitan manejar la transacción de datos a la base de datos y viceversa de forma segura.

Es recomendable utilizar la programación orientada a objetos para la respectiva conexión con la Base de Datos, de esta manera permitir la integridad de los datos sin corromper a la Base de Datos.

IV.5.2.5 Proceso de datos en SQL

Utilización de las librerías correspondientes al lenguaje de programación en el cual se utiliza para la inserción, modificación y consulta de datos. A esto se lo denomina binding para construir las sentencias SQL, estos componentes nos ayudan a evitar la inyección de SQL.

Mencionar que en algunas tecnologías de desarrollo web ya viene incorporado con el uso de Framework, y en caso de no ser así se recomienda construir de forma genérica.

IV.5.3 Capa servicios de datos web

Es la capa en donde persisten los datos y es la encargada de acceder a los mismos, está formada por uno o más gestores de Bases de Datos que realizan todo el almacenamiento de datos, reciben las peticiones de almacenamiento o recuperación de datos desde la capa lógica de negocio.

Cada Base de Datos debe tener asignados un usuario con los roles respectivos, independiente del administrador. A continuación, se observa algunas de las técnicas y herramientas referentes a esta capa.

IV.5.3.1 Servicios web

Un servicio web puede ser consumida por otras aplicaciones, y es la encargada de exponer los datos por medio de lógica de negocio a consumidores de este tipo de servicios. Básicamente sirve para intercambiar información entre dos aplicaciones independientemente al lenguaje en el que se encuentren implementados, cuyos servicios se encuentran desarrollados en un estándar llamado Web Services Description Language y utilizan el protocolo SOAP en su mayoría, no es el único protocolo utilizado para este tipo de servicios.

Se debe tener precaución respecto al formato de datos que maneja este tipo de servicios, porque puede ser tratadas y expuestas de manera errónea. Por lo general es usado el XML en su respuesta.

IV.5.3.2 Credenciales de acceso

La función de las credenciales de acceso o de autenticación es prevenir el robo de identidad de una manera fácil. Esta función trabaja de dos maneras. Primero, le permite verificar si la aplicación web es legítima, y no una aplicación falsa o copia, posteriormente pasa a la autenticación, si los pares de datos son correctos usuario y contraseña, la aplicación permite el acceso, caso contrario hace una solicitud de pregunta secreta.

La pregunta secreta es un mecanismo que permite verificar la veracidad del usuario autorizado para el acceso a la aplicación web. Ahora las preguntas pueden ser expresados mediante una imagen o frase. También se puede relacionar un usuario a una dirección IP, con esto se puede disminuir la suplantación de identidad.

IV.5.3.3 Limitar las peticiones

Esta técnica permite limitar la cantidad de peticiones al usuario, generalmente son usados para limitar las descargas. En caso de tener una cantidad elevada de peticiones puede causar que la aplicación web sea propensa a un ataque de tipo denegación de servicio.

Este tipo de ataques pueden provocar inestabilidad de las aplicaciones web, incluso la caída del servidor. Por tal motivo se debe considerar la incorporación de esta técnica de limitar las peticiones permitiendo el rendimiento y estabilidad.

IV.5.4 Capa de alojamiento web

En esta capa se encuentra alojada la aplicación web en su totalidad, por ello es necesario cumplir con las seguridades requeridas que puedan garantizar un buen funcionamiento de las aplicaciones web.

La primera medida de seguridad a realizar en contar con un proveedor de servicio hosting de renombre, estos proveedores en lo mínimo deben contar con un buen soporte y facilitar las configuraciones de seguridad necesarias. Para el propósito se presenta algunas de las técnicas y herramientas que son recomendables para esta capa, de esta manera garantizar aplicaciones web estables.

IV.5.4.1 Balanceo en la carga de datos

En las aplicaciones web donde hay un gran número de peticiones, es necesario equilibrar la carga de datos de manera que los servidores compartan el trabajo. Consiste realizar un cluster ya sea de tipo servidor de aplicaciones o base de datos, es decir repartir las peticiones entre servidores. En caso de que un servidor falle los demás siguen trabajando.

IV.5.4.2 Componentes ORM

Al utilizar los componentes ORM se permite disminuir el acceso constante a la base de datos debido a que se almacenan los registros de las últimas consultas. Las nuevas peticiones primero acuden al cache, si es que los datos consultados se encuentran será generado la respuesta respectiva a la petición, caso contrario se acude a la Base de Datos y almacenarlo en dicho cache, con esto se aligera el trabajo a la Bases de Datos.

IV.5.4.3 Servidor de cache

Un servidor de cache es de mucha utilidad cuando se trata de almacenar las ultimas peticiones realizadas por medio de un navegador web, la información que se guardan generalmente son las imágenes, códigos scripts entre otros.

El propósito de este servidor es la disminución de la carga a la aplicación web, por tanto, al momento de realizar alguna petición por primera vez, algunos de los elementos quedan almacenados, por tanto, para las siguientes peticiones en primera instancia acudo al servidor cache y despliega los resultados sin necesidad de que la petición acuda al servidor de aplicaciones.

El uso de los servidores de cache aligera el generando de una nueva respuesta por cada petición, acortando el tiempo de respuesta a las peticiones realizadas.

IV.5.4.4 Corta fuegos como seguridad

Un corta fuegos en una aplicación web es esencial, porque tiene por objetivo principal de proteger de los diversos ataques realizados por personas ajenas, además permite monitorear el tráfico de HTTP.

Algunos de los antivirus utilizan los cortafuegos que permiten analizar en tiempo real el tráfico de los datos. Existen una variedad de corta fuegos disponibles con diferentes propósitos, pero con el objetivo de no permitir el ingreso de elementos maliciosos, sin necesidad de hacer cambios a la aplicación.

IV.5.4.5 Implementación de SSL

Permite tener confidencialidad en las transmisiones de nuestra capa lógica de negocio a la capa de presentación visual y viceversa.

Proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico, típicamente el RC4 o IDEA, y cifrando la clave de sesión de RC4 o IDEA mediante un algoritmo de cifrado de clave pública, típicamente el RSA.

Un componente muy común para la generación de los certificados digitales es la utilización de OpenSSL.

IV.6 Seguridad mediante capas en las aplicaciones web

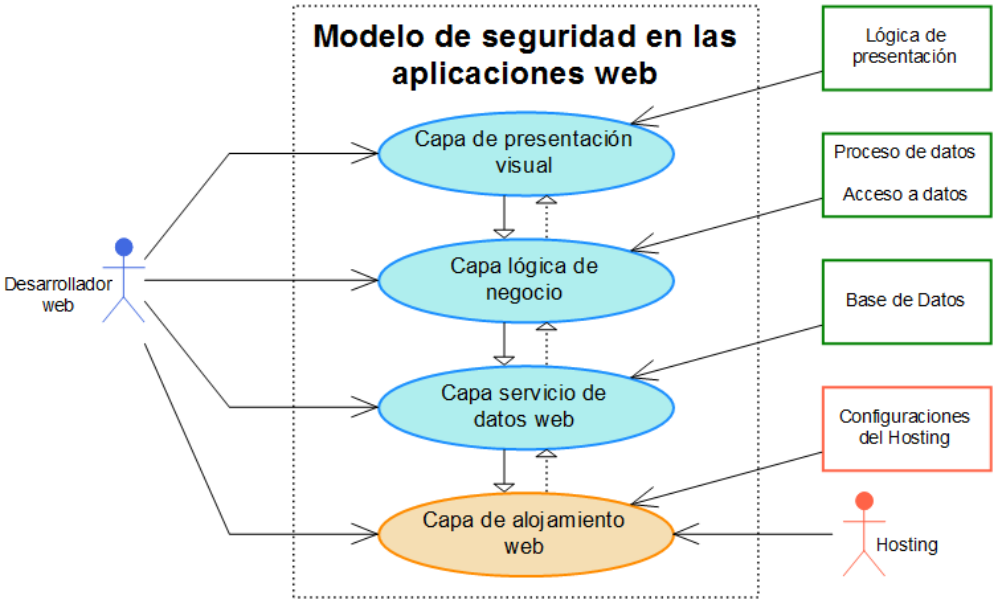


Figura 13. Modelo de seguridad en las aplicaciones web mediante capas

Fuente: Internet

IV.7 Prototipo

En el prototipo se abordan las vulnerabilidades más comunes que se encuentran en las aplicaciones web, como SQL inyección y Cross Site Scripting en sus tres casos, como también en el almacenamiento.

Las herramientas para el desarrollo del prototipo son las siguientes: HTML, CSS, JAVASCRIPT, ANGULAR 8, SpringBoot y PostgreSQL, estas herramientas son de software libre y son utilizadas generalmente para el desarrollo de las aplicaciones web en su mayoría. Existen una variedad de Framework como Laravel en PHP, Spring en JAVA y ASP.NET MVC Framework de Microsoft .NET.

En primera instancia se presenta el diseño de interfaz del prototipo.

Figura 14. ***Interfaz general del prototipo***

Fuente: Elaboración propia

IV.7.1 Caso de estudio 1

IV.7.1.1 Descripción

Con esta práctica vamos a realizar en un principio las pruebas de **SQL inyección** (Ver acápite II.8.5.), según a la propuesta de este tipo de vulnerabilidades.

Para ello se implementa un formulario de autenticación, el cual permite la identificación de los usuarios autorizados. Es así que la aplicación no debe admitir el acceso a los usuarios ajenos, por lo que el proceso de los datos ingresados se realiza de la siguiente manera.

La conexión de la base de datos debe ser de la siguiente manera:

```
#base de datos
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://localhost:5432/academico
db.username=postgres
db.password=5432

#port
server.port=8080

#thymeleaf
spring.thymeleaf.cache=false
spring.thymeleaf.enabled=true
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.vm

#path de la imagen
upload.path=C:/Users/HP-PERSONAL/eclipse-workspace/ejemplo3/src/main/resources/proyectos
```

Figura 15. Conexión de la base de datos

Fuente: Elaboración propia, application.propertie.

```
package taller1.ejemplo3.config;

import javax.sql.DataSource;

@Configuration
@EnableTransactionManagement
public class DataBaseConfig {
    @Autowired
    private Environment env;
    //@Autowired
    //private DataSource dataSource;

    @Bean
    public DataSource dataSource() {

        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setDriverClassName(this.env.getProperty("db.driver"));
        dataSource.setUrl(this.env.getProperty("db.url"));
        dataSource.setUsername(this.env.getProperty("db.username"));
        dataSource.setPassword(this.env.getProperty("db.password"));

        return dataSource;
    }
}
```

***Figura 16.* Configuración de la base de datos**

Fuente: Elaboración propia, DataBaseConfig.java

Después de ver la respectiva conexión que permite la fluidez de los datos (Ver acápite 4.5.2.4.) se hace el procesado de autenticación con el proceso de dato en SQL (Ver acápite 4.5.2.5.), como se observa a continuación:

```
54    public Personas authentification(String login, String password) {
55        sql = "select p.* from personas p, datos d where d.login = ? and d.password1 = ? and d.codper = p.codper ";
56        try {
57            return db.queryForObject(sql, new MapperPersonas(), new Object[] { login, password });
58        } catch (Exception e) {
59            e.printStackTrace();
60            System.out.println("\n No recibido autentificado... \n");
61            return null;
62        }
63    }
```

***Figura 17.* Autenticación**

Fuente: Elaboración propia, PersonasManager.java

En la figura 4-6 se observa el formulario de autenticación o inicio de sesión. Cuyo formulario es desplegado haciendo clic en el botón “Conectar” presentando dos entradas de datos; el primero el campo de “Usuario”, el segundo el campo de “Clave o Contraseña” y un botón de acción para ejecutar la instrucción de inicio de sesión “Iniciar Sesión”.

IV.7.1.2 Prueba

En dicho formulario se realizan las diferentes pruebas de vulnerabilidad de tipo SQL inyección.

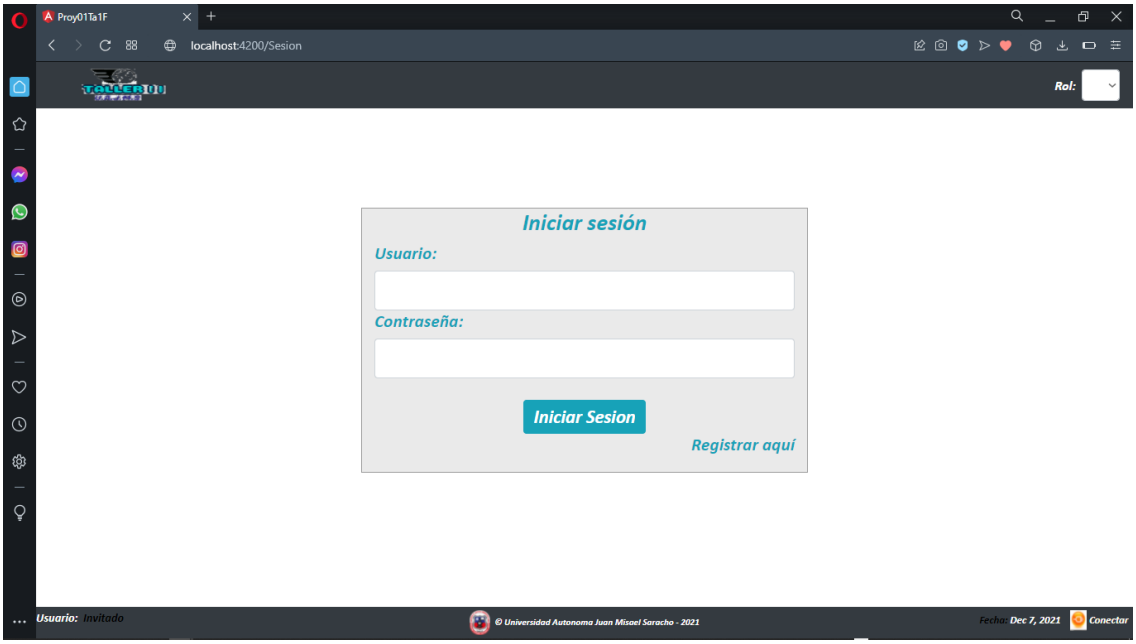


Figura 18. ***Formulario de inicio de sesión.***

Fuente: Elaboración propia

En la figura 4-7 se muestra el resultado de la autenticación correcto, sin intento de vulnerar con SQL inyección. Por lo tanto, no genera ninguna alerta de notificación, es así que la aplicación indica que la sesión ha sido iniciada correctamente y muestra el usuario autenticado.

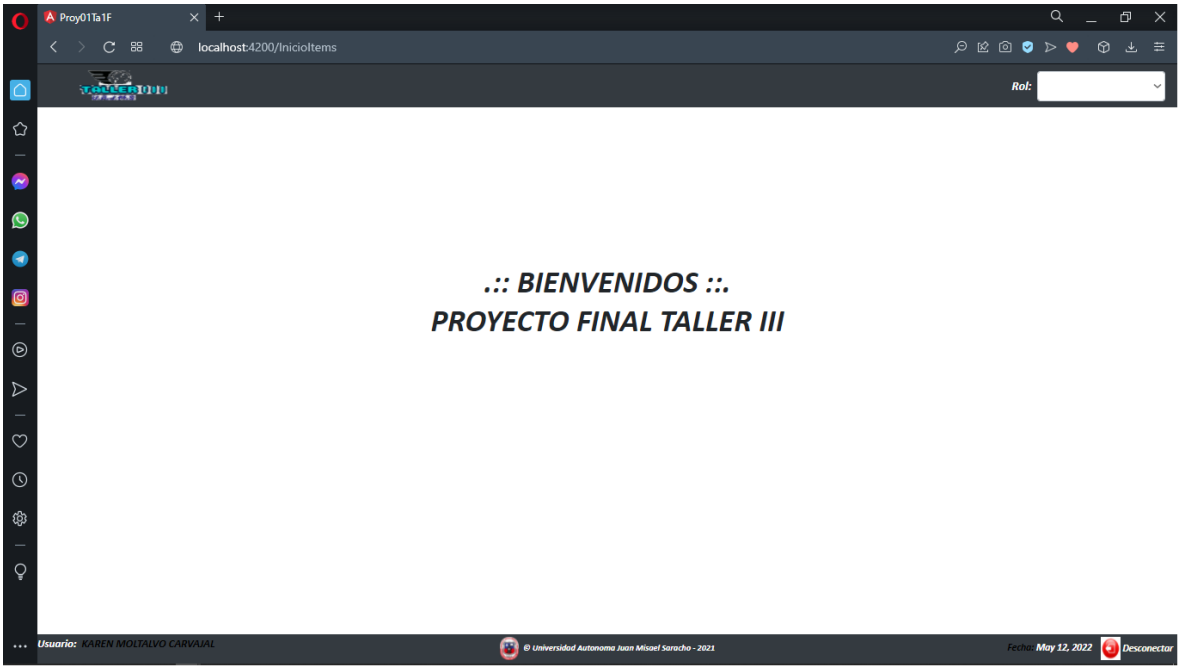


***Figura 19.*Inicio de sesión con datos correctos.**

Fuente: Elaboración propia

Función Autentificación

```
public Personas authentification(String login, String password) {
    sql = "select p.* from personas p, datos d where d.login = ? and d.password1 = ? and d.codper = p.codper ";
    try {
        return db.queryForObject(sql, new MapperPersonas(), new Object[] { login, password });
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("\n No recibido autentificado... \n");
        return null;
    }
}
```

Ahora se procede a intento de vulnerar con SQL inyección (Ver acápite II.8.5.) como se observa en la figura 4-8, las instrucciones que se usan son las siguientes.

Entonces, ya que hemos descubierto como inyectar código que forme parte de la sentencia SQL ¿qué ocurriría si usamos como usuario y contraseña ***asdf' OR 'a'='a***? Veámoslo:

```
sql = "select p.* from personas p, datos d where d.login = asdf OR 'a' = 'a and
d.password1 = asdf OR 'a' = 'a and d.codper = p.codper ";
```

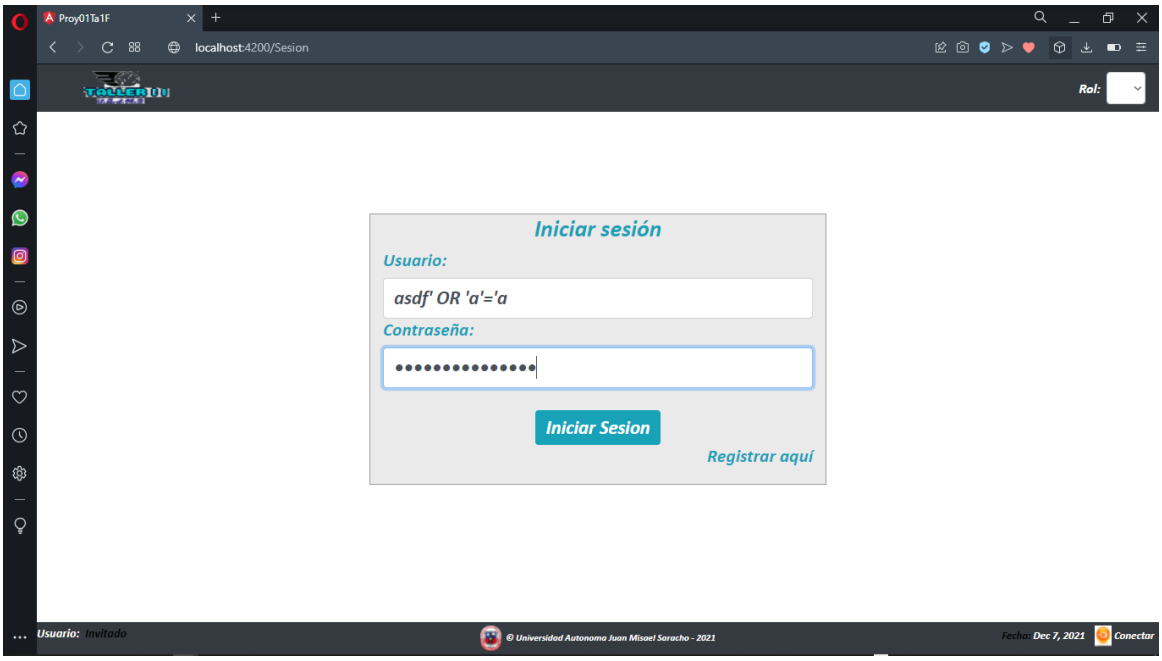


***Figura 20.* Inicio de sesión con inyección SQL.**

Fuente: Elaboración propia

IV.7.1.3 Resultado

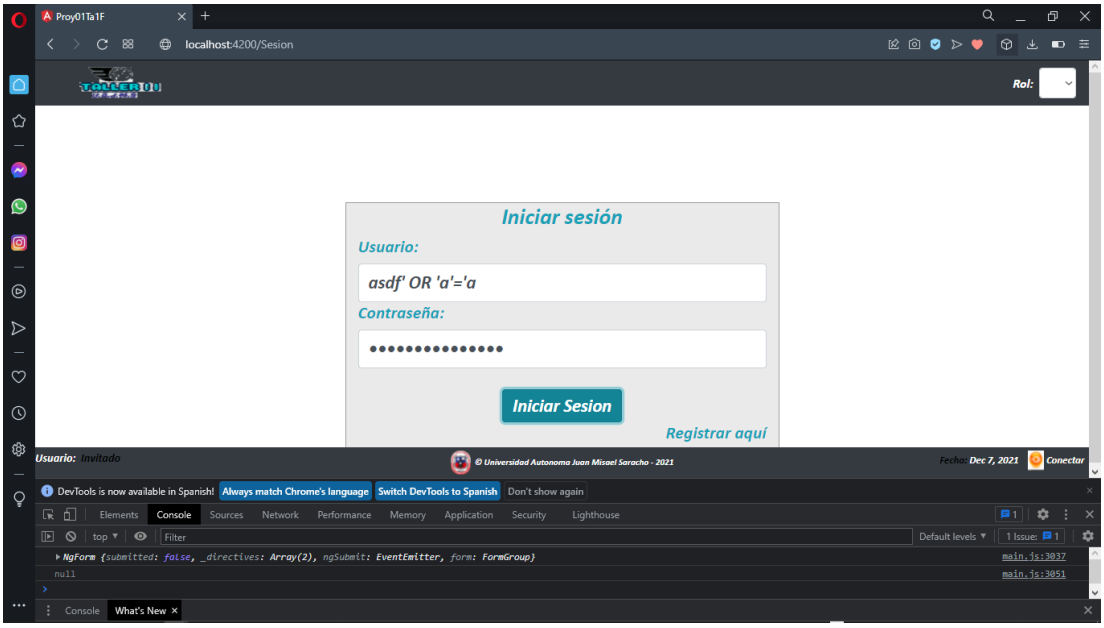


Figura 21. ***Intento de inicio de sesión con inyección SQL, Front-end.***

Fuente: Elaboración propia

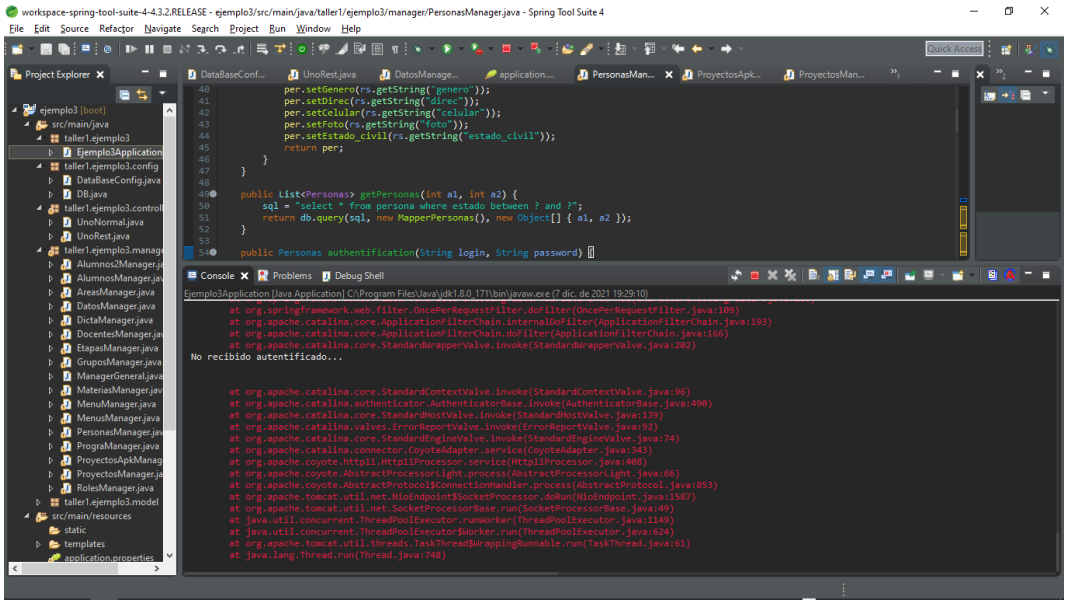


Figura 22. ***Intento de inicio de sesión con inyección SQL, Back-end.***

Fuente: Elaboración propia

Es así que se observa la funcionalidad de las técnicas y herramientas sugeridas contribuyendo a que las aplicaciones sean menos vulnerables. Además, mencionar que es recomendable guardar las actividades que realiza el usuario autenticado en las aplicaciones.

Para ver esquemáticamente la funcionalidad véase la figura 23 en la cual se observa el comportamiento durante el proceso de autenticación.

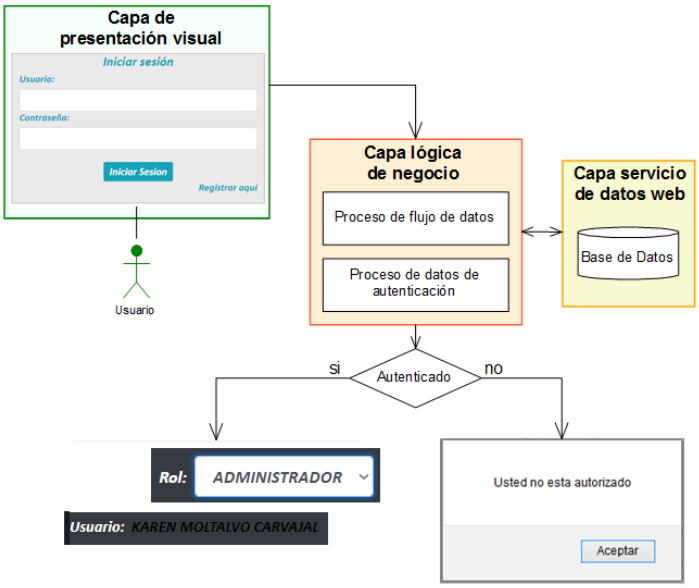


Figura 23. Proceso de autenticación

IV.7.1.4 Recomendación

No confiar en ninguna entrada de usuario

Cualquier entrada de usuario que utiliza autentificación SQL representa un potencial riesgo de inyección SQL.

Así que, es recomendable tratar a todas las entradas del usuario como si no fuesen de confianza.

Incluso si son usuarios autenticados, lo mejor es tratarlas todas de la misma manera.

IV.7.2 Caso de estudio 2

IV.7.2.1 Descripción

Ahora se realiza las pruebas de vulnerabilidad de tipo **Cross Site Scripting (XSS)**, para ello usaremos el código de JavaScript, tomando en cuenta tres formas de vulnerar listados a continuación:

- El Tipo-0, que se utiliza para ejecutar código remotamente o de forma local.
- El Tipo-1, ataque no-persistente o reflejado utilizado en páginas no estáticas.
- Y el Tipo-2, ataque persistente, donde se inyecta código en aplicaciones.

Suponiendo que la aplicación sea vulnerable al ataque XSS, hacemos la prueba de la siguiente manera, pasamos un valor como parámetro, en este caso es un código JavaScript:

https://prototipo.com/InicioItems/index.do?codper=<script>alert(“Pagina_vulnerable_a_ataques_XSS”);</script>

como se observa en la figura 24.

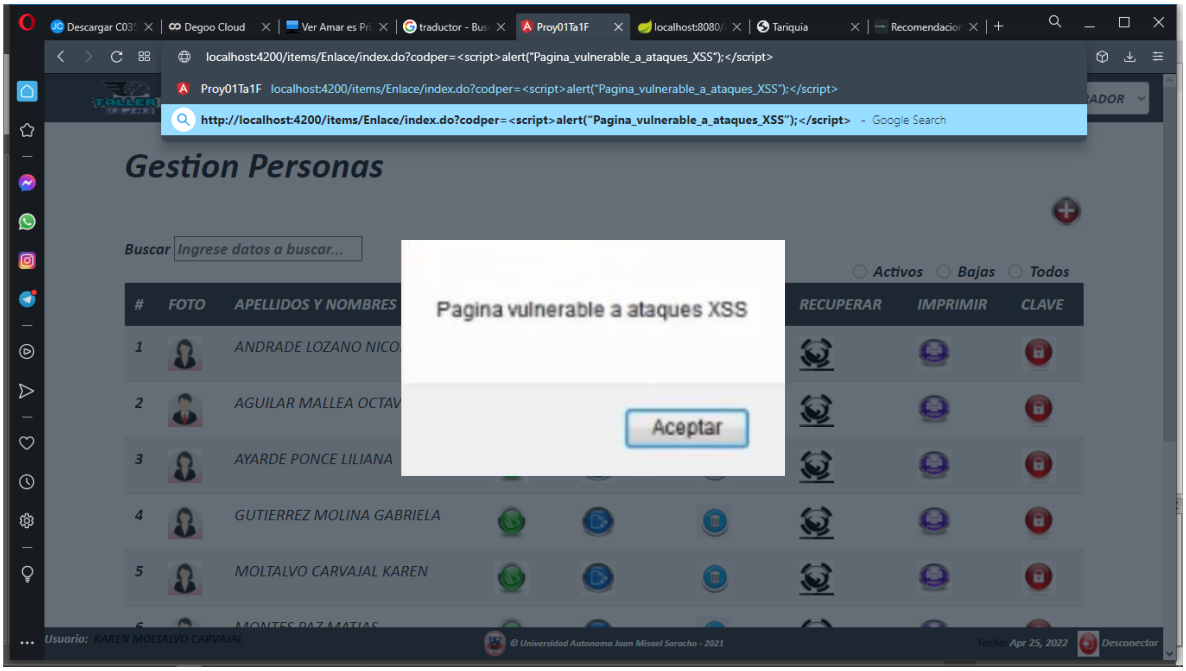


Figura 24. ***Aplicación vulnerable a ataques XSS***

La diferencia entre el Tipo-0 y los otros tipos de ataque, es que el código se inyecta a través de la URL, pero no se incluye en el código fuente de la aplicación. Es decir, el código nunca llegó a la aplicación web, solo se ejecutó en el navegador.

Se puede modificar la información a cualquier elemento HTML contenedor que tenga un identificador ya sea por CODPER, ID, NAME o CLASS. Estos identificadores pueden ser obtenidos con el código JavaScript.

El propósito es realizar la inserción de la información modificada con el modelo de objeto de documento **DOM** por ejemplo de la siguiente manera:

El propósito es realizar la inserción de la información modificada con el DOM por ejemplo de la siguiente manera:

```
document.getElementById('contenidoxss').innerHTML ='TEXTO MODIFICADO';

$('#contenidoxss').text('TEXTO MODIFICADO');
```

IV.7.2.2 Prueba

Se introduce la siguiente URL https://prototipo.com/InicioItems/index.do?codper=<script>document.getElementById('contenidoxss').innerHTML='TEXTO ANHADIDO POR URL APLICANCO EL ATAQUE XSS';</script>

IV.7.2.3 Resultado

Con cualquiera de los códigos el resultado es un texto modificado, resaltado con el color rojo. Como se observa en la figura 25.

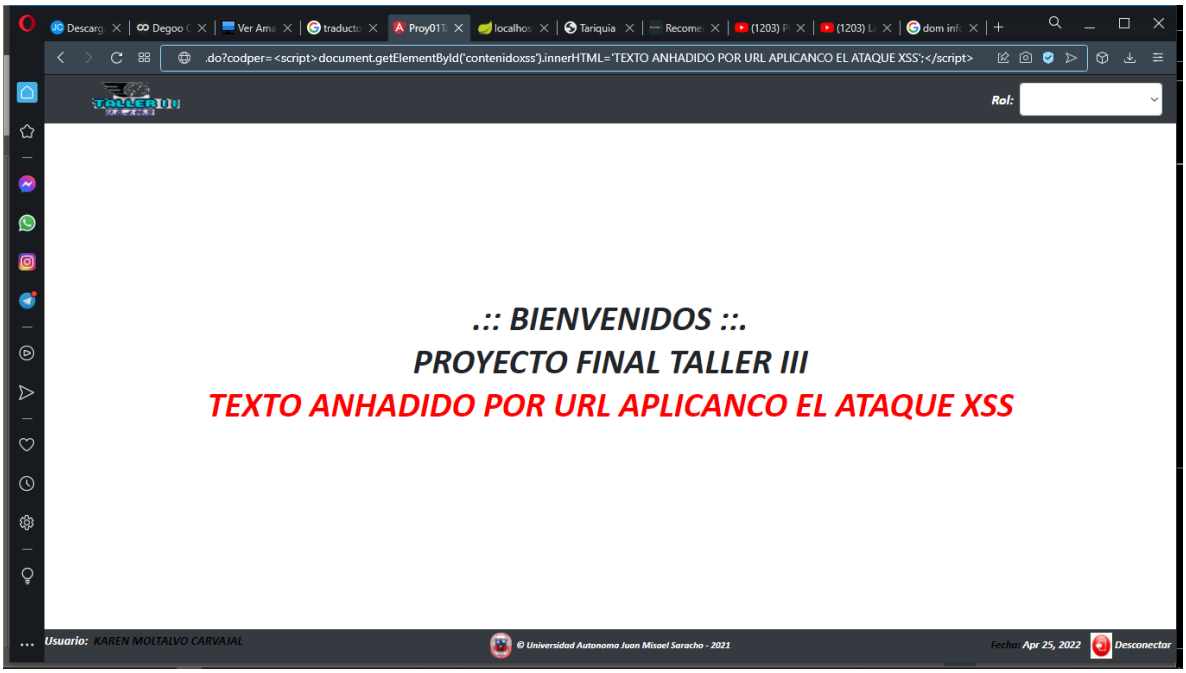


Figura 25. ***Vulnerabilidad XSS de Tipo-0***

IV.7.2.4 Recomendación

Mantener las aplicaciones y sistemas (navegadores web, antivirus, sistema operativo) actualizados. Los navegadores, por ejemplo, utilizan distintos filtros que analizan las solicitudes HTTP, el código HTML y las URLs para advertir o eliminar funciones sospechosas que se ejecutan en el navegador.

IV.7.3 Caso de estudio 3

IV.7.3.1 Descripción

El de **Tipo-1** probablemente es la vulnerabilidad más común que hay, por lo usual se encuentra en motores de búsqueda, es una vulnerabilidad no persistente o reflejada.

Utilizando otra aplicación falsa a la original, pero con el mismo contenido. El comportamiento es de la siguiente manera como se observa en la figura 26.

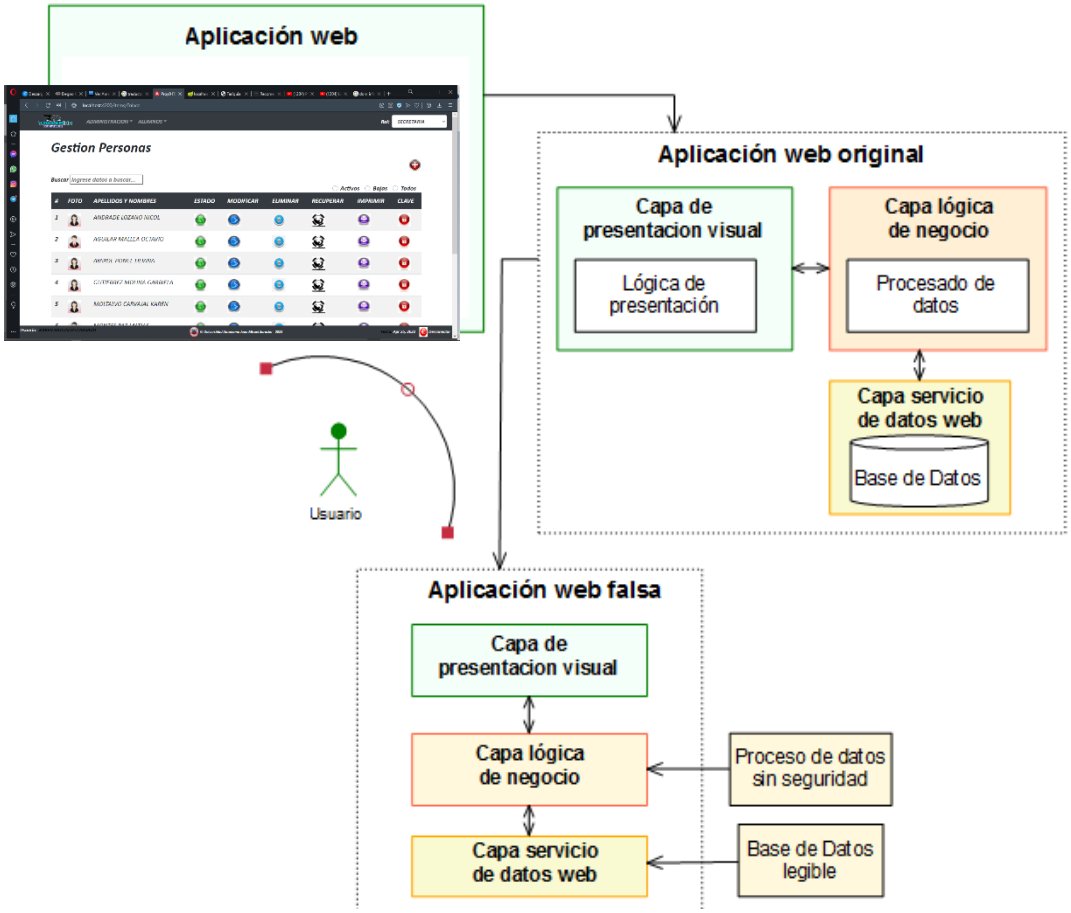


*Figura 26. **Comportamiento de la vulnerabilidad XSS de Tipo-1***

IV.7.3.2 Prueba

Para el desarrollo de esta prueba de vulnerabilidad el o los atacantes muestran una interfaz de una aplicación web falsa igual a la interfaz de la aplicación web original.

IV.7.3.3 Resultado

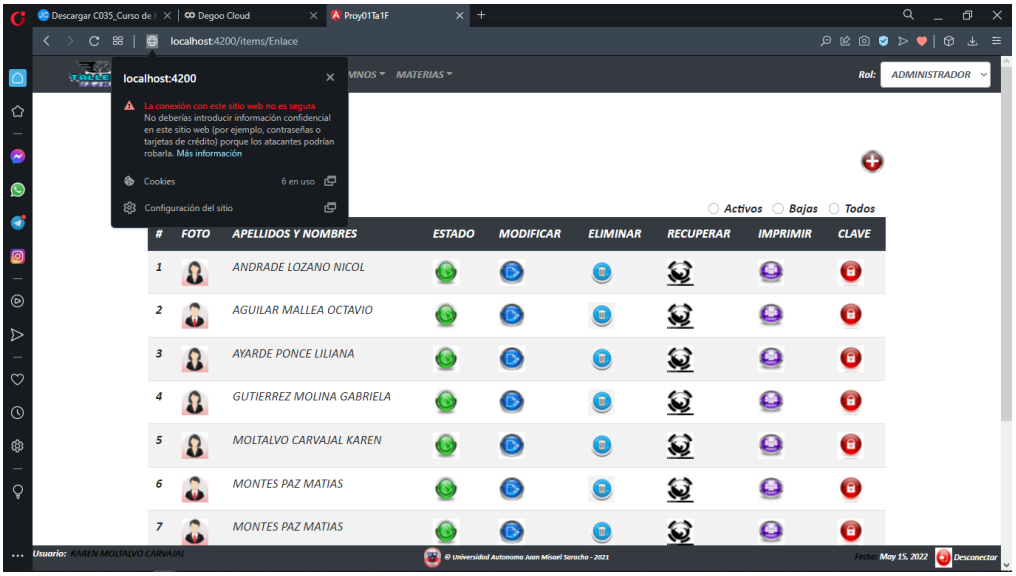


Figura 27. Interfaz aplicación falsa Vulnerabilidad XSS de Tipo-1

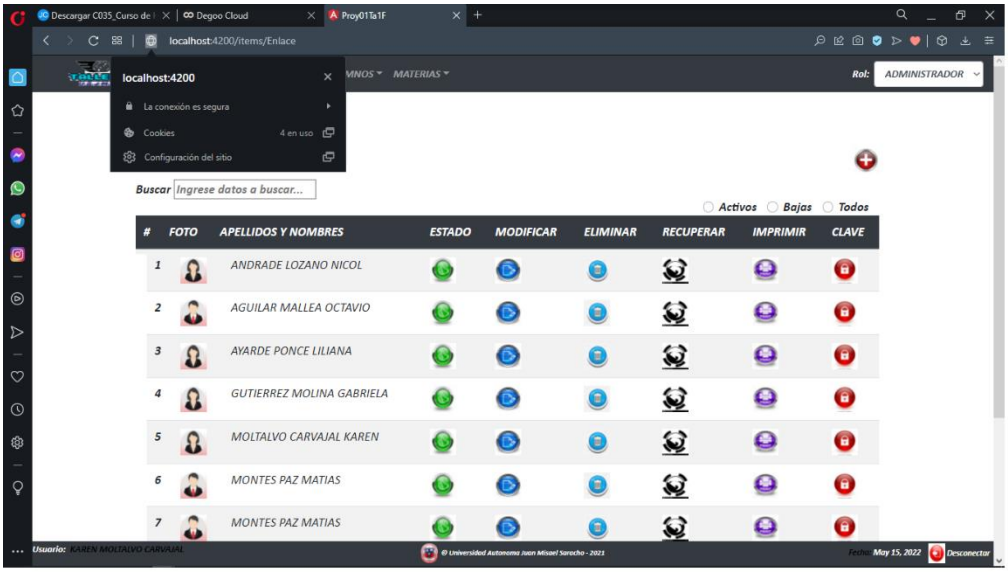
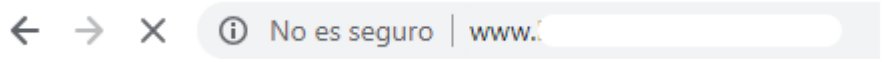


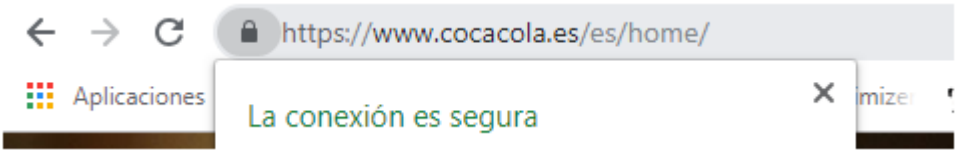
Figura 28. Interfaz aplicación Original Vulnerabilidad XSS de Tipo-1

IV.7.3.4 Recomendación

Cuando una aplicación web no es segura porque no tiene instalado un certificado SSL verás una alerta en el navegador. A continuación, te mostramos cómo se muestra actualmente la alerta en Google Chrome:



En cambio, si las conexiones a la web están cifradas con un certificado SSL, es decir, es seguro enviar datos personales o cualquier otra información, verás que el navegador sustituye esta alerta por un candado en el que podrás obtener más información sobre la página web.



Cuando una aplicación web es segura y tiene un SSL instalado para cifrar las conexiones, la página deja de funcionar bajo el protocolo HTTP y lo hace bajo HTTPS (Protocolo Seguro de Transferencia de Hipertexto o Hypertext Transport Protocol Secure, en inglés)

Precisamente esta «s» final marca la diferencia entre una web segura y otra que no lo es. En una página con HTTPS puedes navegar con tranquilidad; tus datos viajan encriptados o cifrados y no van a ser interceptados por terceros.

Nota: El protocolo HTTPS combina HTTP y TLS, lo que hace posible que todos los datos viajen seguros desde el navegador al servidor web.

Por lo tanto, se recomienda instalar un certificado SSL al prototipo de la aplicación web para encriptar los datos que viajan desde el navegador al servidor web.

Antes de introducir datos en una aplicación web se debe verificar la información de la página, esto para diferenciar si una aplicación web es segura o no, de esta manera se puede diferenciar una aplicación web falsa de una aplicación web real.

IV.7.4 Caso de estudio 4

IV.7.4.1 Descripción

La vulnerabilidad **XSS de Tipo-2** permite hacer los ataques más peligrosos y poderosos a las aplicaciones web. Es conocida como vulnerabilidad persistente o almacenada. La diferencia a los anteriores, es que la inyección la información proporcionada es almacenada en la base de datos, después es mostrada a otros usuarios que visiten la aplicación, por eso se dice que la vulnerabilidad persiste. Esto es lo poderoso, la inyección se quedará siempre en alguna parte de la web, no estará en una página que se genera cada vez que se pasa información al servidor.

También se puede utilizar para obtener información sobre las Base de Datos realizando de la siguiente forma:

IV.7.4.2 Prueba

https://prototipo.com/InicioItems/index.do?codper=select * from pg_catalog.pg_tables where schemaname != 'pg_catalog' and schemaname != 'information_schema'

IV.7.4.3 Resultado

El resultado que nos despliega la consulta sql introducida mediante la URL son los datos del catálogo de la base de datos de PostgreSQL, en esta consulta usamos una condición en la ***WHERE*** cláusula para filtrar las tablas del sistema. Si omite la ***WHERE*** clausula, se obtendrá muchas tablas, incluidas las tablas del sistema.

	Schemaname name	tablename name	tableowner name	tablespace name	hasindexes boolean	hasrules boolean	hastriggers boolean	rowsecurity boolean
1	public	alumnos2	postgres	[null]	true	false	true	false
2	public	areas	postgres	[null]	true	false	true	false
3	public	datos	postgres	[null]	true	false	true	false
4	public	dicta	postgres	[null]	true	false	true	false
5	public	docentes	postgres	[null]	true	false	true	false
6	public	etapas	postgres	[null]	true	false	true	false
7	public	grupos	postgres	[null]	true	false	true	false
8	public	menus	postgres	[null]	true	false	true	false
9	public	mepro	postgres	[null]	true	false	true	false
10	public	personas	postgres	[null]	true	false	true	false
11	public	procesos	postgres	[null]	true	false	true	false
12	public	progra	postgres	[null]	true	false	true	false
13	public	proyectos	postgres	[null]	true	false	true	false
14	public	roles	postgres	[null]	true	false	true	false
15	public	rolme	postgres	[null]	true	false	true	false
16	public	usurol	postgres	[null]	true	false	true	false

Tabla 10: ***Datos de la catalogo de la base de datos***

Si se quieren mostrar solo los nombres de las tablas de la base de datos de PostgreSQL utilizaremos la siguiente consulta:

```
SELECT table_name --seleccionamos solo la columna del nombre de la tabla
FROM information_schema.tables --seleccionamos la información del esquema
WHERE table_schema='public' --las tablas se encuentran en el esquema publico
AND table_type='BASE TABLE'; --tiene que ser del tipo table ya que aqui se listan tambien las vistas
```

Utilizando la consulta anterior la URL sería la siguiente:

https://prototipo.com/InicioItems/index.do?codper=SELECT table_name FROM information_schema.tables WHERE table_schema='public' AND table_type='BASE TABLE' El resultado que nos despliega, son los nombres de las tablas que tiene la Base de Datos, como se observa en la figura 29.

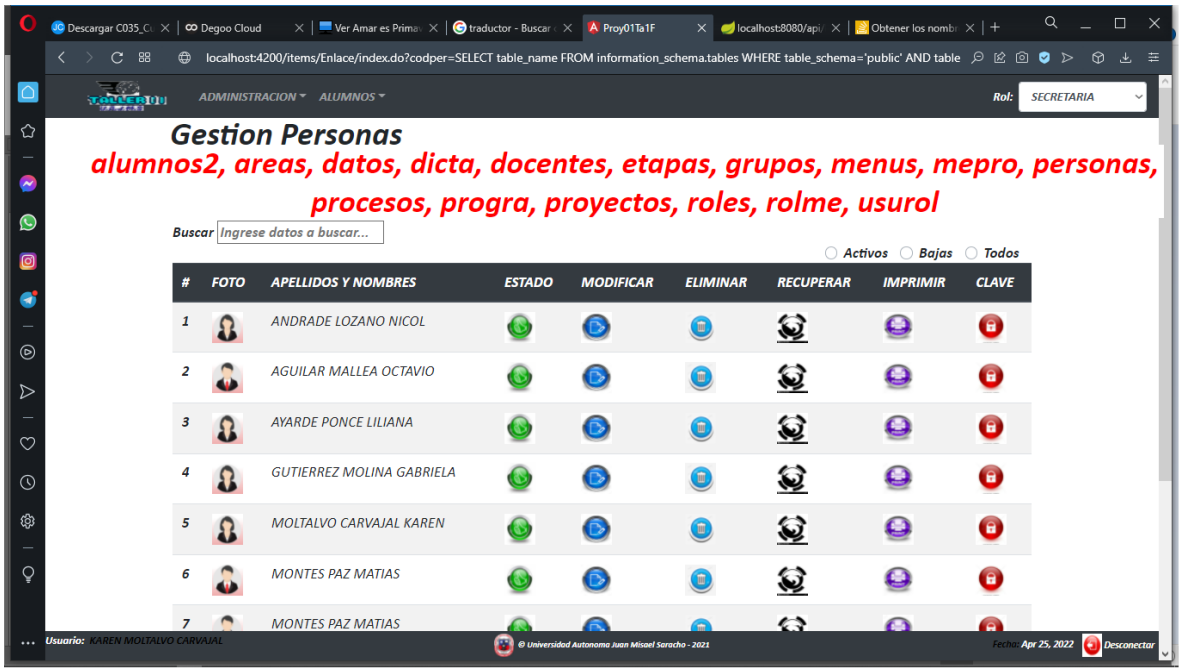


Figura 29. ***Vulnerabilidad XSS de Tipo-2***

Esquemáticamente su comportamiento es como se observa en la figura 30, el ataque del Tipo-2 aprovecha la vulnerabilidad en URL de las aplicaciones web. La instrucción que vulnera es por envió de parámetros con el método GET, para que el servidor procese estos datos y como resultado proporcione información referente a la Base de Datos, pudiendo extraer incluso los datos de las tablas.

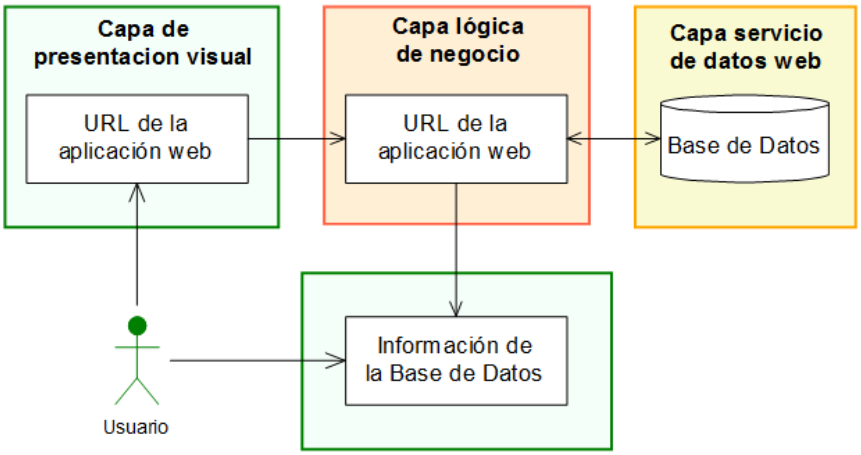


Figura 30. ***Comportamiento de la vulnerabilidad XSS de Tipo-2***

IV.7.4.4 Recomendación

Validar los datos es importante para una aplicación web, pero no es suficiente para crear una aplicación web, es más, para crear una aplicación web segura es más importante codificar correctamente los datos emitidos por la aplicación web. Los ataques XSS se producen precisamente por no codificar correctamente los datos emitidos provenientes de una fuente no confiable. Una fuente no confiable puede ser un parámetro en una aplicación web pero también puede ser cualquier otro dato que incluya una petición HTTP como una cabecera.

- Codificar o validar los datos para evitar ataques XSS en el prototipo de la aplicación web.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

V. CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES

V.1 Conclusiones

El presente proyecto de investigación se realizó de acuerdo a los objetivos propuestos en un principio y las conclusiones a las que se llegaron en relación a la seguridad en las aplicaciones web son las siguientes:

El objetivo general descrito en capítulo I, se cumplió con la realización de la aplicación práctica utilizando el prototipo de la aplicación web “Sistema de Gestión de Proyectos de Grado”.

Además, se ha elaborado recomendaciones para que las aplicaciones web que utilizan los usuarios cumplan con una serie de requisitos mínimos.

Durante el proceso de desarrollo de una aplicación web dentro de una empresa se debe dar mucha importancia a la seguridad, ya que existe una dependencia hacia la tecnología, si bien las amenazas, en cuanto a tecnologías de información, siempre van un paso más adelante que las soluciones, entonces se debe implementar las soluciones adecuadamente y aprovechar al máximo.

Existe la necesidad de crear una cultura sobre ello y así mismo, volverlo un hábito por medio de la auditoría en informática y la implementación de modelos de seguridad.

Durante el desarrollo del prototipo de la aplicación web “Sistema de Gestión de Proyectos de Grado” se realizaron varias pruebas de seguridad, esto con la finalidad de que al concluir el desarrollo de la aplicación web se tenga un mejor nivel de seguridad para evitar cualquier vulnerabilidad que pueda presentarse cuando la aplicación ingrese en funcionamiento. En conclusión, con el desarrollo del presente proyecto se busca concientizar a los desarrolladores de las aplicaciones web, esto para mejorar el nivel de seguridad de este tipo de aplicaciones.

Angular proporciona una serie de protocolos de seguridad que son muy importantes durante el desarrollo de la aplicación, angular por si solo está preparado para evitar ciertas vulnerabilidades que se presentan durante el desarrollo de las aplicaciones web.

El prototipo de la aplicación web “Sistema de gestión de proyectos de grado” cuenta con varias validaciones en cada campo de texto.

Se logró completar en un 80% las pruebas de vulnerabilidades al prototipo de la aplicación web “Sistema de gestión de proyectos de grado”.

V.2 Recomendaciones

Para evitar que nuestras aplicaciones web sean vulnerables a un ataque de SQL Injection, nunca debemos confiar en la información que el usuario introduce en los formularios, toda esa información debe ser verificada y validada

Hacer que las sesiones expiren en un tiempo corto, este tiempo tiene que ser el suficiente para que el usuario haga la transacción que requiere.

Forzar a que el usuario termine su sesión para que de esta forma se evite que la sesión del usuario quede activa y se pueda hacer mal uso de ella.

Hacer del conocimiento del usuario que los problemas mencionados anteriormente se pueden presentar y concientizarlo de cómo prevenirlos.

Ocultar la URL en navegadores y códigos fuente de aplicaciones para evitar su mal uso.

Hacer un filtrado de datos que llegan al servidor, es decir, verificar que los tipos de datos sean los que se esperan.

Hacer que la composición de la URL sea compleja, es decir, cifrar los datos que viajan a través de esta.

Evitar que llegue código HTML y/o JavaScript por medio del método Get o Post, es decir, hay que filtrar el código. Algunos lenguajes de programación del lado del servidor proveen funciones para evitar que el código HTML llegue como tal a nuestras aplicaciones. Un ejemplo de esto es PHP que cuenta con funciones que convierten el código HTML en texto o simplemente lo suprimen, tal es el caso de las funciones htmlspecialchars() y htmlentities().

Proteger las vistas con guards en angular, los guards pueden ser extensibles para que permitan acceder bajo las condiciones que nosotros solicitemos, podemos incluso hacer peticiones a un back-end antes de que el usuario entre en la página. Dentro de los guards hay 4 tipos principales: CanActivate, CanActivateChild, CanDeactivate y CanLoad.

Procurar que los datos viajen por Post en lugar de Get para evitar ataques de XSS por URL.

También es recomendable hacer uso de navegadores modernos, ya que algunos tienen la capacidad de detectar casos en donde se podría presentar la vulnerabilidad y lo invalidan.

Hacer el intercambio de la información a través de internet en un lugar seguro y no sitios públicos como escuelas, lugares de trabajo o cibercafés.

Proteger el equipo con software antiphishing, sobre todo para personas que no tienen muchos conocimientos de informática.

Advertir al usuario de no abrir ligas o correos electrónicos sospechosos o de dudosa procedencia.

Durante el desarrollo de una aplicación web se debe prestar mucho interés a la seguridad de la aplicación, esto para evitar la fuga de información o mal uso de la aplicación por parte de los criminales informáticos.

Al realizar los formularios de envió de datos por parte del usuario, se debe validar cada campo de texto, esto para evitar entradas como ser sentencias de sql para visualizar cierta información confidencial almacenada en la base de datos.

En el equipo de desarrollo de una aplicación web se debe contar con un especialista en seguridad con una dedicación completa.

Utilizar nuevas tecnologías de desarrollo de Sistemas Web ya que hacen más rápido la realización de dicho sistema.