Estrategia de diseño de aplicaciones Web enfocada a prevenir vulnerabilidades de seguridad para el Departamento de Sistemas de la Universidad Simón Bolívar Sede Cúcuta.

Facultad de Ingenierías y Arquitectura

Universidad de Pamplona N.S.

Tesis dirigida por Mg. Luis Alberto Esteban Villamizar, codirigida por Mg. Sandra Milena Vargas Angulo y desarrollada por el Ing. José Darío Rodríguez Leal, para optar por el título de Magíster en Gestión de Proyectos Informáticos.

Pamplona, Abril de 2018

Dedicatoria

A la memoria de mi padre. A mi madre y mis hermanos, por ser soporte, guía e inspiración para ser cada día una mejor persona y profesional.

Agradecimientos

Me gustaría expresar mi más sincero agradecimiento a las siguientes personas:

En primer lugar, a mis directores Luis Alberto Esteban y Sandra Milena Vargas por guiarme con mano experta en todos los aspectos de este trabajo de investigación. Les estaré siempre agradecido por su ilimitado talento y su firme dedicación.

A Raúl Eduardo Rodríguez Ibáñez y Yurley Karime Hernández Peña por aportar su conocimiento y experiencia en la validación del instrumento utilizado para recolectar la información requerida para el desarrollo de la investigación. Mis sinceras gracias a su dedicación y consejos brindados.

A Ramiro Alfonso Gómez, Cleiver Andrade y Miguel Eduardo Ropero por aportar su conocimiento en el desarrollo de software Web para la validación de la política de desarrollo seguro del departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta. Mi más sincero agradecimiento por su tiempo, su pericia y dedicación.

A Carlos René Angarita Sanguino, Jefe de Sistemas de la Universidad Simón Bolívar sede Cúcuta, mi gratitud por el apoyo brindado en la elaboración del documento y el tiempo dedicado a despejar mis dudas con respecto al funcionamiento del departamento que lidera.

Y también a todos aquellos que me ofrecieron generosamente su ayuda mientras llevaba a cabo la investigación del presente proyecto. Gracias por su generosidad y franqueza al compartir conmigo sus conocimientos.

Tabla de Contenidos

Lista de figuras

Lista de tablas

Resumen

Este trabajo diseña un conjunto de prácticas y lineamientos como estrategia para reducir el riesgo de los aplicativos web desarrollados en el Departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta, a partir de un diagnóstico del estado actual de la política de seguridad en la institución frente a otras instituciones de la región, un análisis de la normativa y el estado del arte de la seguridad en aplicaciones web como objeto de estudio. Esta estrategia fue validada mediante el juicio de expertos en el desarrollo de aplicaciones web y logro reflejar la importancia de aplicar dichas políticas para prevenir las vulnerabilidades en el software web institucional.

Palabras clave: Aplicación web, Estrategia desarrollo seguro, Política de desarrollo web, Seguridad informática, Vulnerabilidades de software.

Abstract

This work design a set of practices and guidelines as a strategy to reduce the risk of web applications developed in the Systems Department of the Simon Bolivar University from Cucuta, based on a diagnosis of the current state of the security policy in the institution in front of other institutions in the region, an analysis of the regulations and the state of the art of security in web applications as an object of study. This strategy was validated through the judgment of experts in the development of web applications and it was possible to reflect the importance of applying these policies to prevent vulnerabilities in the institutional web software.

Keywords: Web Application, Safe development strategy, Web development policy, Informatic security, Software vulnerabilities.

1 Introducción

El presente trabajo de investigación parte de la necesidad de dotar al Departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta de buenas prácticas de seguridad al momento de realizar los desarrollos de software web, esto debido a que la Institución está migrando sus sistemas hacia ambientes web los cuales estarán más vulnerables a ataques de piratas informáticos. Para esto se toma como base el conocimiento de algunas instituciones que a nivel mundial dictan parámetros para ayudar a prevenir los problemas de seguridad que los desarrolladores puedan dejar en las aplicaciones al momento de realizar el desarrollo y también el ciclo de vida del desarrollo de software seguro ya que ayudará a tener en cuenta los aspectos de seguridad desde la toma de requerimientos, pasando por las pruebas e implementación. La Institución en la actualidad no tiene políticas de seguridad establecidas para el desarrollo de software, se toman controles y se cuenta con una arquitectura de desarrollo propia pero los parámetros de seguridad los determina el desarrollador por cuenta propia al momento de realizar los desarrollos; es por esto que en la presente investigación se comparó con algunos parámetros de seguridad que se están aplicando en otras Instituciones de Educación Superior de la región de Norte de Santander y teniendo en cuenta la información recolectada en el estado del arte y marco teórico se diseñó una política de desarrollo de software web seguro la cual dicta los parámetros que deben tener en cuenta los desarrolladores al momento de iniciar la construcción de nuevos sistemas o las empresas en caso que la institución decida tercerizar el software institucional.

El objetivo de la estrategia propuesta, es dotar a los desarrolladores de la Institución de los patrones que deben seguir al momento de desarrollar aplicaciones web, que garanticen la seguridad, evitando efectos de vulnerabilidades como los ataques de inyección sobre las aplicaciones, fallas XSS, sesiones rotas, referencias inseguras a objetos, ataques CRFS, criptografía insegura, entre otras fallas que se pudieran presentar. Con este planteamiento, se busca garantizar la integridad de la información y a su vez, fortalecer el correcto funcionamiento de los diferentes aplicativos que son desarrollados y puestos en marcha en las diferentes dependencias de la Universidad para uso de la comunidad educativa.

La validación de la estrategia planteada, se realizará por medio del juicio y prueba de expertos, quienes finalmente determinarán si la propuesta es aplicable a los procesos de desarrollo de software web que se generan al interior del Departamento de Sistemas de la institución.

Este documento está organizado en cinco capítulos en los cuales se podrá encontrar los siguientes temas. El primer capítulo describe los alcances del trabajo en términos de planteamiento del problema, objetivos y metodología de investigación utilizada. El segundo resume el marco teórico fundamental para la comprensión de la estrategia diseñada y el estado del arte de la seguridad de las aplicaciones web. El tercer capítulo describe la estrategia como producto principal del proceso de investigación, junto con el proceso de validación el cual se realizó por medio de juicio de expertos. El cuarto capítulo presenta las conclusiones del trabajo. El quinto capítulo presenta las referencias bibliográficas utilizadas durante la investigación y finalmente se presentan los anexos entre los cuales se destaca el instrumento con el cual se recolecto información del estado actual de las políticas de seguridad de las instituciones de educación superior de la región, el formulario de validación de la estrategia y el artículo enviado para publicación.

1.1 Planteamiento del problema

En la actualidad se ve como el desarrollo de software se está volcando cada vez más hacia ambientes web dejando de lado el software que se desarrolla para ambientes de escritorio. Esto se debe a las múltiples ventajas que este tipo de aplicaciones presentan como por ejemplo el hecho de ser multiplataforma ya que son ejecutadas en los navegadores web lo que garantiza la inmediatez de acceso, no necesitan ser descargadas, instaladas y configuradas, las actualizaciones que se le realizan a la aplicación quedan disponibles de forma transparente al usuario y consumen menos recursos. La Universidad Simón Bolívar sede Cúcuta, consiente de esta realidad, ha iniciado la implementación de un programa institucional de migración de sus sistemas convencionales hacia nuevas tecnologías enfocadas a este tipo de ambientes. Con este nuevo enfoque de desarrollo, es importante establecer mecanismos robustos que prevengan las vulnerabilidades de seguridad a los que se encuentran expuestos los aplicativos, ya que al estar disponibles al público en general, puede ocasionar que la información se vea expuesta y sea más vulnerable a los ataques, perjudicando directamente la integridad y disponibilidad de la

información, tanto académica como financiera, poniendo en riesgo la continuidad de las actividades administrativas y académicas que hacen parte de la cotidianidad y de la dinámica de la Institución. Es una realidad, que las aplicaciones tienen un nivel alto de vulnerabilidad a ataques, indistintamente de la tecnología web sobre la cual se desarrolle, se enfrentan hoy a algunos de los mayores desafíos de seguridad debido a diferentes tipos de riesgos: troyanos, robo de identidad, ataques de inyección, spyware, guerra electrónica y muchas más formas de ataques cibernéticos. Sin embargo es posible disminuir estos riesgos si se cuenta con una serie de procedimientos y métodos de seguridad durante su proceso de desarrollo.

De acuerdo con (Mark Dowd, 2006) en su libro The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities los tipos de vulnerabilidades se pueden definir en términos de cada fase del ciclo de vida de desarrollo de software. Estos incluyen el diseño comúnmente aceptado, la implementación y vulnerabilidades operacionales. Generalmente las vulnerabilidades se deben a defectos en la arquitectura seleccionada y a especificaciones del software. Se pueden abordar en la fase de análisis de requisitos o en las especificaciones de la fase de diseño. Las vulnerabilidades de implementación hacen referencia a los errores de seguridad que comenten los desarrolladores al construir módulos u objetos del sistema para cumplir con las especificaciones propuestas. Las vulnerabilidades operacionales se refieren a los defectos de seguridad que surgen en el despliegue y la configuración del sistema desarrollado en un entorno particular.

Frente a este planteamiento, y evidenciada la necesidad del entorno en cuestión, surge la siguiente interrogante: ¿Es posible prevenir vulnerabilidades de seguridad en aplicaciones web, mediante la aplicación de una estrategia para el desarrollo de aplicaciones web enfocada a prevenir vulnerabilidades de seguridad para el Departamento de Sistemas de la Universidad Simón Bolívar Sede Cúcuta?

1.2 Justificación

La implementación de una estrategia de desarrollo web, enfocada a prevenir las vulnerabilidades de seguridad sobre los procesos de desarrollo de aplicaciones, permitirá disminuir los riesgos de pérdida de productividad de los sistemas de información, ya que dentro

de la dinámica de calidad concerniente al mejoramiento continuo y fortalecimiento de los sistemas de información, es inminente que la Institución migre sus sistemas a ambientes web, esto frente a la creciente competitividad que exige el mundo moderno, el cual tiene una tendencia clara hacia el mejoramiento y facilidad de acceso a la información. De igual forma, se busca generar el procedimiento adecuado para que los desarrolladores al iniciar la construcción de nuevos productos de software, tengan en cuenta cuales serían los posibles riesgos de seguridad sobre las aplicaciones, y de esta forma generar los mecanismos de protección sobre las mismas, entre las cuales se debe tener en cuenta los ataques de inyección, las fallas XSS, el manejo inadecuado de las sesiones, los ataques CRSF, criptografía insegura, fallas de restricción URL, entre otras vulnerabilidades.

Según la OWASP (Open Web Application Security Project), los desarrolladores de aplicaciones web necesitan una estrategia o guía para producir aplicaciones seguras desde su construcción en la cual se debe tener en cuenta los principios de seguridad tales como son la confidencialidad la cual debe permitir el acceso únicamente a los datos a los cuales el usuario está permitido, la integridad que se encarga de asegurar que los datos no se falsifican o alteran por usuarios no autorizados y la disponibilidad que debe garantizar que los sistemas y datos estén disponibles para los usuarios autorizados cuando lo necesiten. Para realizar la selección de los controles que se deben tener en cuenta, sólo es posible después de clasificar los datos a proteger. Por ejemplo, controles aplicables a sistemas de bajo valor tales como blogs y foros son diferentes al nivel y número de controles adecuados para sistemas de alto valor como lo son en el caso de la institución los sistemas académicos y financieros los cuales al estar en un ambiente web estaría en riesgo de ser manipulada por criminales informáticos que logren acceder a las plataformas institucionales.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar una estrategia para el desarrollo de aplicaciones web enfocada a prevenir vulnerabilidades de seguridad para el Departamento de Sistemas de la Universidad Simón Bolívar Sede Cúcuta.

1.3.2 Objetivos Específicos

- Evaluar el estado actual de las políticas de seguridad aplicadas a los procesos de desarrollo de software web del Departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta.
- Analizar comparativamente las políticas de seguridad establecidas y/o implementadas en Instituciones de Educación Superior de la región del Norte de Santander.
- Diseñar una estrategia para el desarrollo de aplicaciones web, enfocada a prevenir vulnerabilidades de seguridad para el Departamento de Sistemas de la Universidad Simón Bolívar Sede Cúcuta.
- Validar la estrategia de diseño de aplicaciones web enfocada a prevenir vulnerabilidades de seguridad para el Departamento de Sistemas de la Universidad Simón Bolívar Sede Cúcuta.

1.4 Metodología de la investigación

A través del siguiente apartado, se presenta el andamiaje metodológico que sustenta el presente proyecto de investigación. En este sentido, se parte de la definición del paradigma, método y tipo de estudio, los cuales en coherencia con el planteamiento de la propuesta, evidencian la solidez de la estrategia sustentada desde un enfoque científico e investigativo, proceso necesario para construir con claridad una ruta de acción que brinde la rigurosidad metodológica necesaria para el desarrollo de todo proyecto informático.

1.4.1 Paradigma de la Investigación

La palabra Paradigma proviene del griego “paradeigma”, cuyo significado es ejemplo o modelo. Este término se emplea para indicar un patrón, modelo, ejemplo o arquetipo. Este concepto fue utilizado en teoría de la ciencia por primera vez por Lichtenberg (1742-1799). A finales de los 60, el filósofo Thomas Kuhn dio a la palabra el significado que tiene en la actualidad al emplearla para referirse al conjunto de prácticas que definen una disciplina científica durante un período específico de tiempo.

El paradigma de la presente investigación es el Histórico Hermenéutico. El término hermenéutico proviene del griego "hermeneuein" que quiere decir interpretar. Según (Martínez, 2009) el método básico de toda ciencia es la observación de los datos o hechos y la interpretación (hermenéutica) de su significado. Toda ciencia trata de desarrollar técnicas especiales para efectuar observaciones sistemáticas y garantizar la interpretación. De esta forma, la calidad de los resultados de una investigación dependerá del nivel de precisión terminología, su rigor metodológico, de la sistematización con que se presenta todo el proceso y de la actitud crítica que la acompañe. La hermenéutica es el arte de interpretar, es por esto que se podría decir que la actividad mental del ser humano se reduce a recibir estímulos visuales, auditivos, olfativos, etc., que por su naturaleza, son ambiguos y amorfos, y a ubicarlos en un contexto que le dé un posible sentido o significado.

Para el caso de la presente investigación, se establecieron previamente unas categorías de análisis, de las cuales se indaga su incidencia y relaciones con la finalidad de llegar al planteamiento de una estrategia de diseño de aplicaciones web enfocada a la prevención de vulnerabilidades de seguridad para el departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta, proceso en el cual, se identificó, comprendió e interpretó un fenómeno, que es una realidad cotidiana de la dependencia en cuestión, dentro del contexto de los Sistemas y de la seguridad informática.

Para (Hurtado, 2010), la hermenéutica abarca el análisis de la temática, corresponde a la fase analítica del proceso operativo. La fase analítica le permite al investigador, criticar y juzgar acerca de la pertinencia de los conceptos e informaciones encontrados, con respecto a su trabajo; valorar la amplitud, el alcance y la capacidad explicativa de las diversas teorías y conceptos en torno a su pregunta de investigación. El investigador asume una visión crítica de los planteamientos de diferentes autores, identifica los aportes y limitaciones de la teoría en cuestión, se centra en los significados y comprende los planteamientos del autor y sus implicaciones.

De acuerdo con esto, se busca valorar el estado de las políticas de seguridad aplicadas a los desarrollos web en la institución y compararlas con las de otras instituciones de la región para así determinar el estado en el cual se encuentran los desarrollos institucionales.

La investigación planteada tiene un enfoque epistemológico basado en la fenomenología, el cual es un método que consiste en describir lo inmediatamente dado en la conciencia. Los fenómenos, cuyo estudio es el objeto de la fenomenología, no debe entenderse en el sentido subjetivista de Kant ni en sentido positivista de Comte y Taine, sino en el sentido de lo inmediatamente dado en la conciencia. La fenomenología quiere dejar la palabra a las cosas mismas (Morales, 2011).

La fenomenología entonces, da cuenta de la realidad. En este sentido, la investigación realizada, inició con la indagación del estado actual de la seguridad en el desarrollo de aplicaciones web en la Institución, una revisión exhaustiva sobre las condiciones y lineamientos vigentes que permitieron develar el panorama presente en la dependencia y de esta manera, ser punto de partida para el planteamiento de la presente propuesta. De igual manera, esta indagación incluyó el análisis de dichas condiciones de seguridad informática en otras Instituciones de la región, permitiendo conocer esas realidades externas que de alguna manera incidieron para la comprensión global del entorno y para develar aspectos claves que fueron tenidos en cuenta para el diseño de la propuesta presentada.

De acuerdo con Martínez (2009), las realidades cuya naturaleza y estructura peculiar solo pueden ser captadas desde el marco de referencia del sujeto que las vive y experimenta, exigen ser estudiadas mediante métodos fenomenológicos. Para estos casos se está estudiando una realidad cuya esencia depende del modo en que es vivida y percibida por el sujeto, una realidad interna y personal, única y propia de cada ser humano. Es por esto que la fenomenología nació y se desarrolló para estudiar estas realidades como son en sí, dejándolas que se manifiesten por si mismas sin constreñir su estructura desde afuera, sino respetándola en su totalidad.

Para Husserl, la fenomenología es el estudio de los fenómenos tal como son experimentados, vividos y percibidos por el hombre. Es la ciencia que trata de descubrir las estructuras esenciales de la conciencia; debido a ellos, el fin de la fenomenología no es tanto describir un fenómeno singular sino descubrir en él, la esencia válida universalmente, y útil científicamente (Taylor y Bodgan, 1992).

El método fenomenológico aporta la inter-subjetividad y la intuición en la comprensión de los fenómenos. En este método los investigadores tratan de describir la realidad vivida por otras personas, trata con significados y describe el mundo de la vida para comprenderlo, comprende lo objetivo en términos de los actos subjetivos, busca comprender el mundo como es experimentado personalmente por cada quien y siempre empieza con la experiencia concreta de las personas. (Gurdian-Fernández, 2007)

Gracias a las experiencias de los desarrolladores de software de la institución se abordó el problema de investigación y se planteó la estrategia de seguridad, ya que el conocimiento en el área por parte del personal como de las instituciones de educación superior a las que se les aplico el instrumento permitió desarrollar un mejor producto para el Departamento de Sistemas de la institución.

1.4.2 Tipo de estudio

El presente estudio se basa en la investigación cualitativa con método descriptivo. La investigación cualitativa según Sampieri (2010), se enfoca en comprender y profundizar los fenómenos, explorándolos desde la perspectiva de los participantes en un ambiente natural y en relación con el contexto. El enfoque cualitativo se selecciona cuando se busca comprender la perspectiva de los participantes (individuos o grupos pequeños de personas a los que se investigará) acerca de los fenómenos que los rodean, profundizar en sus experiencias, perspectivas, opiniones y significados, es decir, la forma en que los participantes perciben subjetivamente su realidad.

Para Noguero (2002), el modelo cualitativo surge como alternativa al paradigma racionalista, puesto que en las disciplinas de ámbito social existen diferentes problemáticas, cuestiones y restricciones que no se pueden explicar ni comprender en toda su extensión desde la metodología cuantitativa. Estos planteamientos proceden fundamentalmente de la antropología, la estenografía y el interaccionismo simbólico. Se propone el método descriptivo ya que ayuda a describir de modo sistemático las características de una población, situación o área de interés. Entre las características principales del método descriptivo se cuentan que busca únicamente

describir situaciones o acontecimientos, con frecuencia las descripciones se hacen con encuestas, no está interesada en comprobar explicaciones.

El presente proyecto es un estudio de tipo cualitativo, dado que independientemente de la naturaleza técnica de su temática, las vulnerabilidades de seguridad hacen parte de una realidad social de la que no se encuentra exento el mundo informático. Este proceso comienza con la definición clara de un problema, el cual fue identificado como una afectación importante y latente para la integridad de la información institucional, y que, a partir de su conocimiento y análisis interno y del entorno, permitió la definición de la naturaleza primaria de su realidad, de sus comportamientos y posibles soluciones plasmadas en el desarrollo de la estrategia de seguridad propuesta.

Para Martínez (2009), el término cualitativo se usa bajo dos acepciones. La primera, para referirse a la cualidad y la segunda, más integral y compresiva, como cuando se refiere al "control de calidad" donde la calidad representa, primordialmente, la naturaleza y la esencia completa, total, de un producto. Es por esto que la investigación cualitativa trata de identificar, básicamente, la naturaleza profunda de las realidades, su estructura dinámica, aquella que da razón plena de su comportamiento y manifestaciones. De aquí que lo cualitativo no se opone de ninguna forma a lo cuantitativo, sino que lo implica e integra, especialmente donde sea importante.

La presente investigación se plantea con una metodología cualitativa ya que se trabajará en una estrategia con la cual los desarrolladores de la institución puedan construir aplicaciones web seguras, para esto se utilizara un método híbrido donde se aplicará un instrumento cuantitativo como lo es la encuesta, que citando a Álvarez y Jurgenson (2003), estos métodos híbridos son valiosos para el desarrollo de estudios cualitativos donde el procedimiento de la aplicación no varía, la única diferencia se encuentra en la forma cómo se interpretan los datos.

Teniendo en cuenta los anteriores planteamientos, se parte de trabajos previamente publicados como artículos científicos, tesis de grado y estándares en relación con la seguridad en el desarrollo de software web. A partir de ello se construirá la estrategia de seguridad para el desarrollo de aplicaciones web.

1.4.3 Población y Muestra

La muestra es un subconjunto representativo y finito que se extrae de la población accesible. En este sentido, una muestra representativa es aquella que por su tamaño y características similares a las del conjunto, permite hacer inferencias o generalizar los resultados al resto de la población con un margen de error conocido (Arias, 2012).

Para la presente investigación, la muestra fueron 6 jefes de sistemas de instituciones de educación superior de Norte de Santander, los cuales diligenciaron una encuesta para determinar el estado de las políticas de seguridad aplicadas a los desarrollos de software web. Con la información recolectada, se evaluará el estado actual de las políticas aplicadas a nivel institucional y de las demás instituciones para realizar un análisis comparativo y así, desarrollar la estrategia que se debe aplicar en la institución.

1.4.4 Instrumentos y Fuentes de Consulta

Para determinar el estado actual de los estándares de seguridad en el desarrollo de software web, se hizo necesario el uso de instrumentos de recolección de información tales como:

1.4.4.1 Encuesta

Para determinar el estado actual de las políticas de seguridad aplicadas a los desarrollos de software web implementados tanto en la institución como en algunas de las instituciones de educación superior de Norte de Santander, se construyó un instrumento teniendo como base la operacionalización de variables (ver anexo 7) que consta de 38 preguntas que fueron aplicadas a los jefes de sistemas o encargados de TI, ver anexo 1.

1.4.4.2 Bases de Datos

Búsqueda bibliográfica en bases de datos científicas como ACM, Redalyc entre otras, con el objetivo de determinar el estado actual del objeto de estudio.

1.4.4.3 Entrevista no estructurada.

Se basaran en consultas técnicas no formales realizadas a desarrolladores de software web, jefes de sistemas o encargados de TI, diferentes profesionales con conocimiento en el área, entre otros, con el objetivo de identificar las pautas que se deben tener en la construcción de la estrategia de seguridad para el desarrollo de aplicaciones web.

1.4.5 Actividades desarrolladas

En la siguiente figura se muestra mediante la descomposición de actividades, cada una de las tareas ejecutadas al logro de cada uno de los objetivos de la investigación.

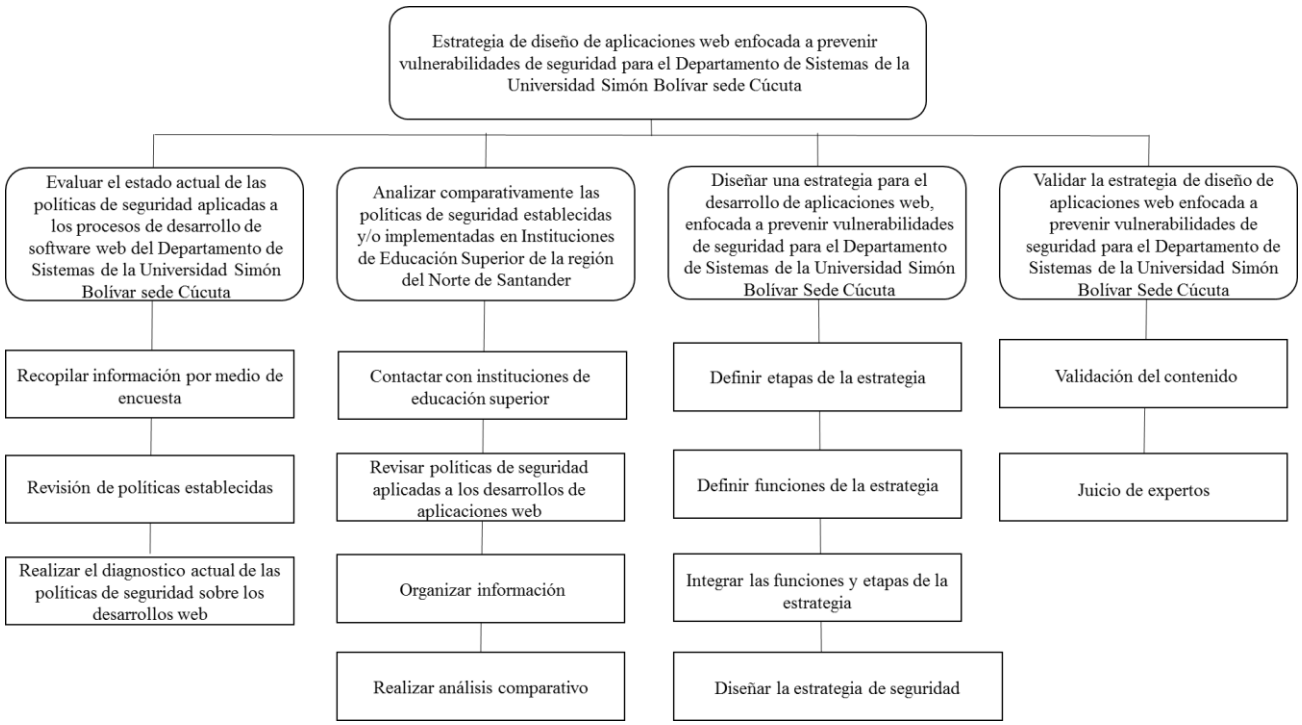

Figura 1 EDT estructura metodológica
Diseño propio

1.4.6 Métodos para la ejecución de actividades.

A continuación, se detalla el desarrollo de cada actividad de acuerdo a los métodos teóricos y empíricos existentes. De acuerdo a esto, cada actividad se relaciona con alguno de los métodos mencionados.

Objetivo 1: Evaluar el estado actual de las políticas de seguridad aplicadas a los procesos de desarrollo de software web del Departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta.

1.1 Recopilar información por medio de encuestas.

Métodos Teóricos	Métodos Empíricos
	Encuesta

1.2 Revisión de políticas establecidas.

Métodos Teóricos	Métodos Empíricos
Análisis y Síntesis	

1.3 Realizar el diagnostico actual de las políticas de seguridad sobre los desarrollos web.

Métodos Teóricos	Métodos Empíricos
Abstracción Científica.	

Objetivo 2: Analizar comparativamente las políticas de seguridad establecidas y/o implementadas en Instituciones de Educación Superior de la región del Norte de Santander.

2.1 Contactar con instituciones de educación superior.

Métodos Teóricos	Métodos Empíricos
	Entrevista , cuestionario

2.2 Revisar políticas de seguridad aplicada a los desarrollos de aplicaciones web.

Métodos Teóricos	Métodos Empíricos

Histórico	

2.3 Organizar la información.

Métodos Teóricos	Métodos Empíricos
Abstracción Científica.	

2.4 Realizar el análisis comparativo.

Métodos Teóricos	Métodos Empíricos
Abstracción Científica.	

Objetivo 3: Diseñar una estrategia para el desarrollo de aplicaciones web, enfocada a prevenir vulnerabilidades de seguridad para el Departamento de Sistemas de la Universidad Simón Bolívar Sede Cúcuta.

3.1 Definir etapas de la estrategia

Métodos Teóricos	Métodos Empíricos
sistémico	Abstracción Científica

3.2 Definir funciones de la estrategia

Métodos Teóricos	Métodos Empíricos
sistémico	Abstracción Científica

3.3 Integrar las funciones y etapas de la estrategia

Métodos Teóricos	Métodos Empíricos

sistémico	Abstracción Científica

3.4 Diseñar la estrategia de seguridad

Métodos Teóricos	Métodos Empíricos
Modelación	Abstracción Científica

Objetivo 4: Validar la estrategia de diseño de aplicaciones web enfocada a prevenir vulnerabilidades de seguridad para el Departamento de Sistemas de la Universidad Simón Bolívar Sede Cúcuta.

4.1 Validación del contenido

Métodos Teóricos	Métodos Empíricos
Análisis- síntesis Abstracción Científica	Cuestionario

4.2 Juicio de expertos

Métodos Teóricos	Métodos Empíricos
Análisis- síntesis Abstracción Científica	Cuestionario

2 Marco teórico y Estado del Arte

Teniendo en cuenta que los sistemas tienen su fundamento en la teoría del estructuralismo la cual se basa en la atención hacia la organización desde el punto de vista de su estructura, funcionamiento y los elementos que se utilizan para lograr los objetivos y de acuerdo con (Hurtado, 2010) quien afirma que, el estructuralismo intenta dilucidar la realidad al elaborar a través de la razón, una estructura que permita dar cuenta de los eventos estudiados y presenta en sí, un carácter de sistema, ya que una modificación en cualquiera de sus elementos implica una modificación en su totalidad. Con base en esto, para la elaboración del marco teórico se tendrá en cuenta lo referente a los estándares de desarrollo de software y seguridad con los que los programadores del departamento de sistemas de la institución deben estar familiarizados, para aplicarlos al momento de realizar la construcción de un software web seguro.

2.1 Desarrollo de Software

La estrategia a desarrollar, está enfocada en establecer las buenas prácticas de desarrollo de software web seguro, al interior del departamento de sistemas de la Universidad Simón Bolívar sede Cúcuta. Para esto, es importante tener claridad sobre ciertas definiciones de estándares y guías que ayudaran a tener una mejor idea de lo que es un producto de software. De acuerdo con (Booch, Jacobson y Rumbaugh, 2000) en la ingeniería de software el objetivo es construir un producto de software o mejorar uno existente. Esto se logra gracias a un proceso efectivo de desarrollo de software el cual debe proporcionar normas para el desarrollo eficiente, seguro y de calidad. El proceso de desarrollo de software debe ser capaz de evolucionar durante muchos años. Durante esta evolución debería limitar su alcance, en un momento del tiempo dado, a las realidades que permitan las tecnologías, herramientas, personas y patrones de organización.

2.1.1 Aspectos del Diseño de la Arquitectura de Seguridad

El diseño de la arquitectura de seguridad de una aplicación, se debe realizar desde el principio del ciclo de Desarrollo de Software. Después de la fase de análisis de requisitos de seguridad de una aplicación, una selección para su arquetipo y patrón de diseño, debe basarse además de la seguridad, en otros aspectos. Todos los tipos de software pueden tener vulnerabilidades en el código de todos los niveles de arquitectura y vulnerabilidades de diseño en sus plataformas y

componentes arquitectónicos. El número de ataques que una aplicación puede sufrir depende de las vulnerabilidades en: (Higuera, 2014)

- Componentes y niveles de arquitectura de software.
- Plataforma y sistema operativo.
- Seguridad del software del cliente incluyendo el sistema operativo.
- Servidores de aplicaciones.
- Red.
- Sistema de administración de base de datos.
- Tecnología de desarrollo
- Lenguajes de programación.
- Experiencia de seguridad y conocimiento de programadores.
- Protección en línea.

Los objetivos de seguridad individuales se pueden utilizar para dividir la arquitectura de la aplicación para un análisis posterior y para ayudar a identificar las vulnerabilidades de la aplicación. Este enfoque conduce a un diseño que optimiza los siguientes objetivos de seguridad: (Goseva-Popstojanova, 2015)

Autenticación: La autenticación es el proceso en el que el software establece definitivamente la identidad del usuario que intenta iniciar sesión, normalmente con credenciales como nombre de usuario y contraseña.

Autorización: La autorización se refiere a cómo una aplicación controla el acceso a recursos y operaciones.

Gestión de la configuración: Ejecute el contexto de la aplicación, las bases de datos que conecta, la forma en que se administra la aplicación y cómo se protegen los recursos. Gestión de configuración hace referencia a cómo una aplicación se encarga de estas operaciones y problemas.

Confidencialidad: La aplicación debe proteger los secretos, los datos confidenciales del usuario y de la aplicación y manejar datos sensibles. Los datos sensibles se refieren a cómo la aplicación

maneja los datos que deben protegerse en la memoria, en la red o en almacenamiento persistente. Por lo general se realiza mediante algoritmos criptográficos.

Integridad: La aplicación debe comprobar los datos o bibliotecas para las alteraciones. Los valores aleatorios deben ser criptográficamente fuertes. También se hace mediante la criptografía.

Disponibilidad: Mantener la disponibilidad de información manejada por un sistema o sus recursos.

No repudio: Proporcione la prueba que una transmisión o recepción particular se ha hecho, el receptor / transmisor no puede negar que ocurrió.

Estos objetivos de seguridad se pueden utilizar para tomar decisiones clave de diseño de seguridad para una aplicación y documentarlos como parte de la arquitectura. Por ejemplo, la Figura 2 muestra los problemas de seguridad identificados en una arquitectura típica de aplicaciones Web.

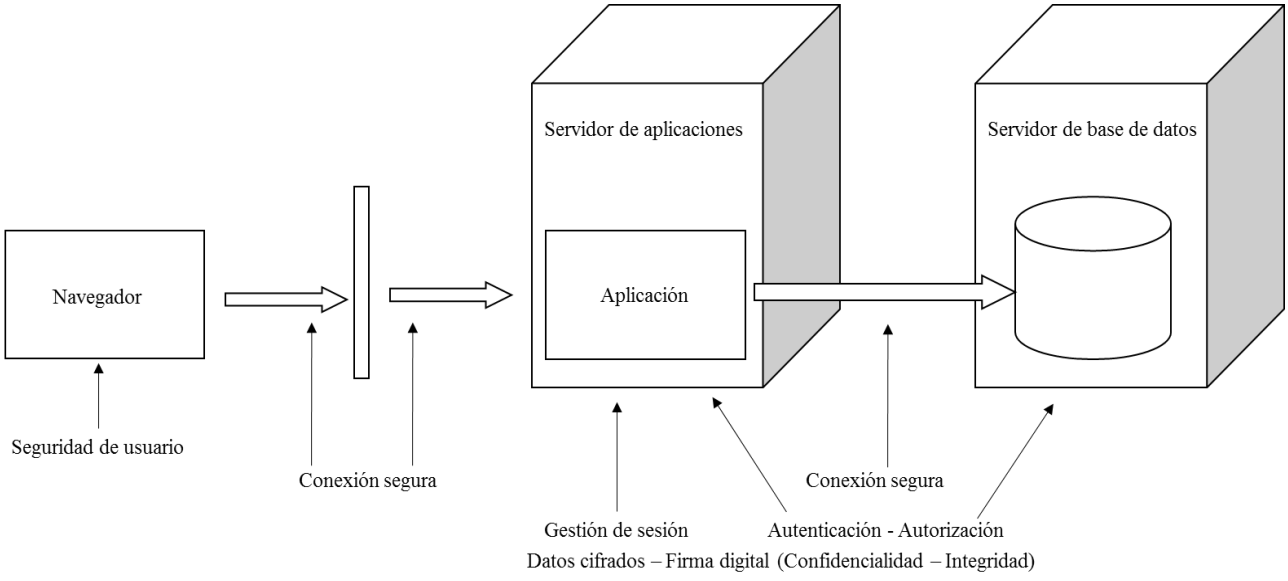

Figura 2 Problemas de seguridad en una arquitectura de aplicaciones web
Diseño propio

La política de seguridad de una organización o institución es un conjunto de reglas que rigen y determinan qué hacer y qué no. De acuerdo con el IETF (Internet Engineering Task Force) se

define como: "Una serie de declaraciones formales (reglas) a las que todas las personas tienen acceso a cualquier organización de la información y la tecnología". Algunas de las características más importantes de cualquier política de seguridad que se pueden destacar son (IETF, 1999):

- Establece qué herramientas son necesarias y qué procedimientos.
- Se utiliza para comunicar un consenso sobre el uso de datos y aplicaciones dentro de la organización.
- Proporciona una base para demostrar el uso inapropiado de recursos, por empleados o externos.
- Define el comportamiento apropiado para cada caso.

La política de seguridad de una organización es de naturaleza dinámica, siempre actualizada, lo que permite tener en cuenta que el mantenimiento de la seguridad es un proceso vivo y debe ser manejable de forma estructurada y organizada. Así, la seguridad es un proceso que debe lograrse mediante el desarrollo de la política de seguridad. La política debe entenderse como un aspecto en continuo cambio, que debe actualizarse siguiendo una serie de principios de seguridad. Muchos expertos en el tema, hablan de los principios de seguridad que deben regir todo el diseño. Michael Howard y David LeBlanc promulgaron en Writing Secure Code (Howard, 2003) los principios como:

- Seguro por diseño.
- Seguro de forma predeterminada.
- Asegurar el despliegue.
- Principio del privilegio mínimo.
- Principio de profundidad en defensa.
- Principio de diversidad de la defensa.
- Identificación de debilidades.
- Gestión centralizada de la seguridad.
- Principio de simplicidad.
- Aprender de los errores.

- Reducción de la superficie de ataque al mínimo.
- Uso de la seguridad predeterminada.
- Suponer que los servicios externos son inseguros.
- Tener planes en caso de falla.
- Error al modo seguro.
- Los componentes seguros no son seguros.
- No mezcle código y datos.
- Solucionar problemas de seguridad correctamente.

2.1.2 Arquitectura Web

La elección de un estilo arquitectónico de aplicaciones web tiene diferentes implicaciones de seguridad que deben ser analizadas, estudiadas y probadas para hacer la selección más segura posible. A continuación se recopilan algunos ejemplos de aplicaciones y estilos de arquitectura con diferentes implicaciones de seguridad de diseño.

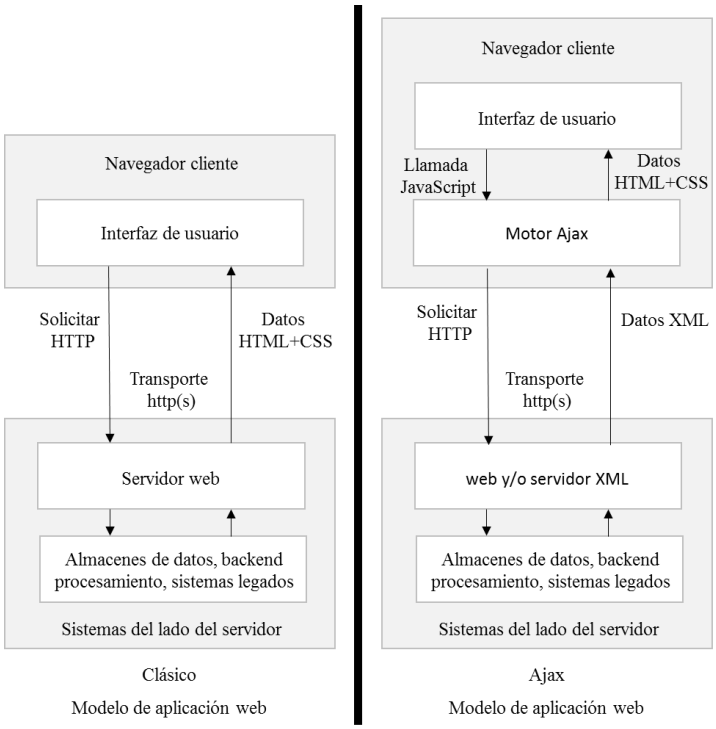

Figura 3 Ajax vs arquitecturas web tradicionales
Diseño propio

En el caso de los tipos de aplicación, los desarrollos web enriquecidos de Ajax, tienen la ventaja de proporcionar interfaces de cliente más ricas y las comunicaciones asíncronas con el servidor lo hacen más rápido. Sin embargo, este tipo de aplicación introduce nuevas fuentes de vulnerabilidades de seguridad en el código del motor Ajax que ejecuta el cliente como, por ejemplo, cross site scripting (XSS) y violaciones de la misma política de origen que acceden a otros dominios no permitidos (figura 3). (Garrett, 2005)

La elección del estilo arquitectónico basado en Service Object Architecture (SOA) es comúnmente utilizado, ya que permite la comunicación estándar entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar. Es por esto que las entidades involucradas en la prestación de un servicio, deben implementar los mismos estándares de seguridad para proporcionar todas las metas de seguridad como autenticación, autorización, confidencialidad, entre otras. Este requisito puede ser difícil de lograr cuando las entidades heterogéneas tienen que interactuar entre sí (Figura 4). (Gutierrez, 2005)

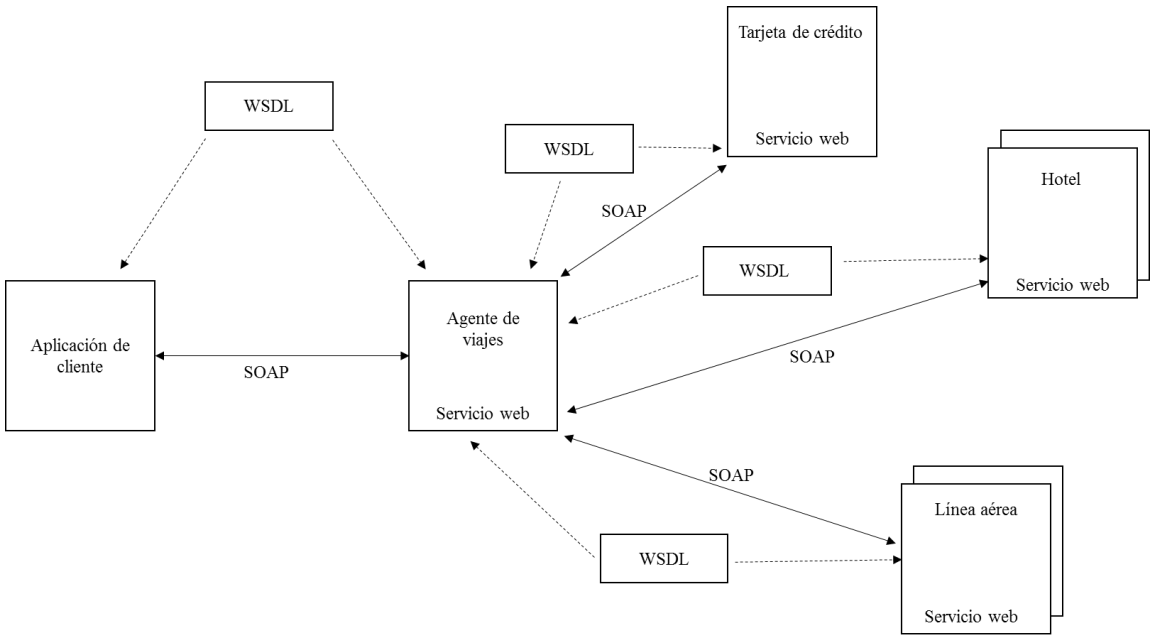

Figura 4 Ejemplo de servicio web en funcionamiento
Diseño propio

Una arquitectura que ha demostrado ser fundamental en el desarrollo de aplicaciones web, es el Modelo-Vista-Controlador (MVC) (Figura 5), este tipo de arquitectura de desarrollo permite separar la interfaz de usuario (Vistas), el modelo del negocio y la lógica de control; como su nombre lo indica, está separado en tres componentes: modelo, controlador y vista. Está basado en la ideología de separación de conceptos y cumple perfectamente con los objetivos de los patrones de diseño. Es por esto que, cualquier Framework creado a partir de MVC permite reutilizar el código, reutilizar vistas evitando así duplicar estilos y contenidos de la misma. Todo el manejo de los datos recae sobre los modelos, por lo que si se modifica la base de datos solo sería necesario modificar el modelo correspondiente para que se permita manejar los datos actualizados sin tener la necesidad de ir a cada lugar donde es utilizado. Sin embargo, al ser solo un patrón de diseño de aplicaciones, se le debe prestar atención a la forma de cómo va a gestionar la seguridad ya que si no se implementan controles tendrían los mismos riegos del desarrollo tradicional (inyección de código, Cross Site Scripting, Ataque de denegación de servicio, suplantación de identidad entre otros). (wcreator, 2017)

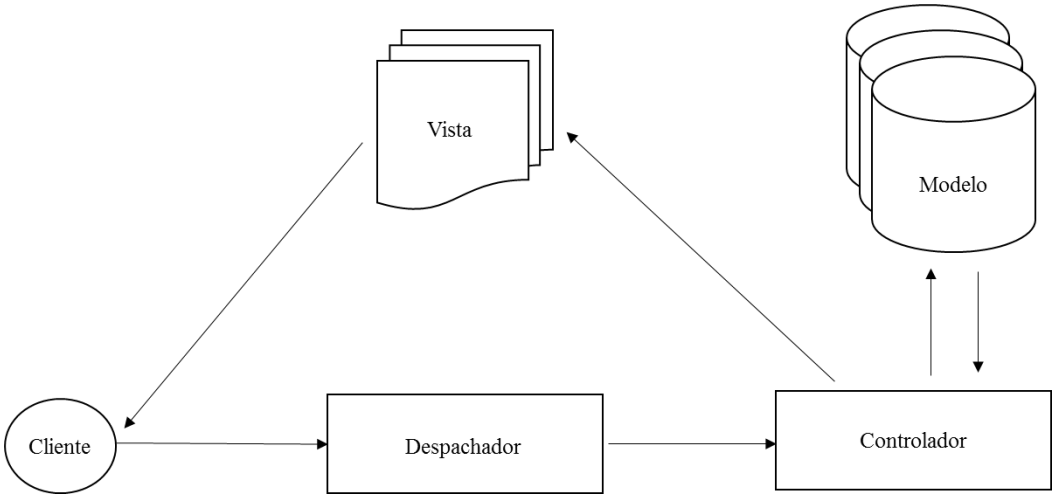

Figura 5 Ejemplo de MVC
Diseño propio

2.1.3 Ciclo de Vida del Desarrollo de Software Seguro

El ciclo de vida del desarrollo de software seguro (SSDLC) es la base para desarrollar aplicaciones web seguras (Figura 6). Este método utiliza diferentes procesos humanos y

tecnológicos, como métodos de requisitos de seguridad, métodos de análisis de riesgos, listas de verificación de seguridad, herramientas de análisis de seguridad entre otras. Las herramientas y métodos que interactúan durante el proceso de desarrollo de software seguro, tienen como propósito desarrollar una aplicación con el mínimo de vulnerabilidades de seguridad. La seguridad hace que la aplicación funcione de una manera deseada y que proporcione defensa contra amenazas de seguridad.

Los controles de seguridad garantizan que la aplicación funcione de una manera deseada y proporciona defensa contra amenazas de seguridad. En la práctica, la seguridad pasa desapercibida en las primeras fases del ciclo de vida de software (SLC) por lo tanto una fase transfiere sus vulnerabilidades a la otra fase. (Daud, 2010)

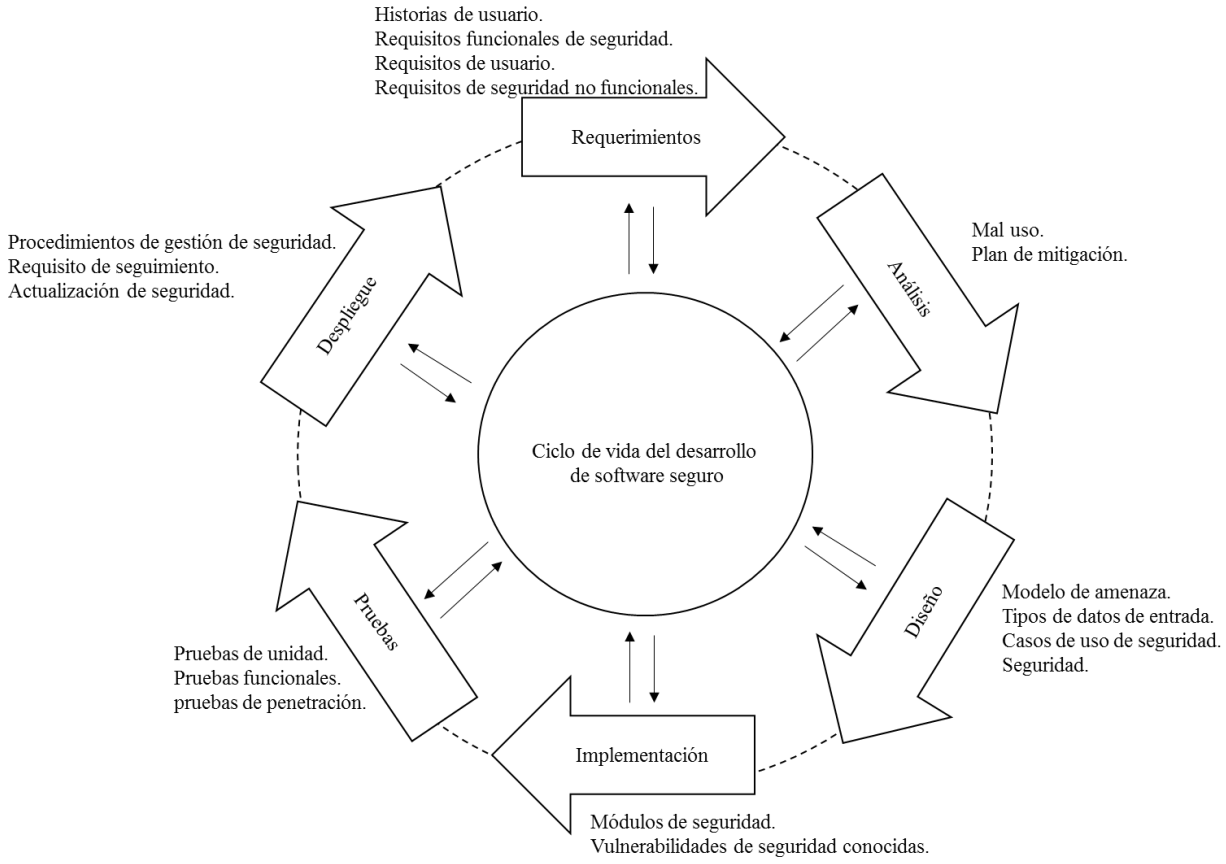

Figura 6 Ciclo de vida desarrollo de software seguro
Diseño propio

2.1.4 Análisis de Requisitos de Seguridad

En cualquier proyecto importante de software, la ingeniería de requisitos ha sido crítica para lograr su éxito. Los requisitos de seguridad se podrían clasificar en tres categorías principales: 1) Requisitos de seguridad funcionales. 2) Requisitos de seguridad no funcionales. 3) Requisitos de seguridad derivados. Los requisitos funcionales, enumeran las funciones que un sistema debe realizar. Estos requisitos hacen referencia a las entradas y salidas de un sistema. Los requisitos no funcionales, enumeran las propiedades que un sistema debe poseer. Los requisitos derivados son los que se derivan de requisitos de seguridad funcionales. (Hope, 2007).

Los requisitos de seguridad de software se podrían tratar, teniendo en cuenta las historias de usuario que son los requisitos de usuario y de ahí se derivan los requisitos de seguridad analizados por los ingenieros de desarrollo encargados de aplicar seguridad al software. Las historias de usuario, son una forma efectiva de derivar los requerimientos de los usuarios de manera eficiente. Es importante anticipar un comportamiento anormal en una aplicación web para que sea segura y fiable. Es por esto que se deben crear casos de uso para mitigar aquellos comportamientos anormales, es decir, un mal uso. (Criteria, 2005)

Antes de definir los requisitos de seguridad, se deben identificar las partes de la aplicación que requieren seguridad. Estas partes de la aplicación se denominan Objetivos de Evaluación (ODE). Una vez se identifica ODE, se pueden encontrar de forma sencilla los Requisitos Funcionales de Seguridad (RFS). En la tabla 1 enumera los RFS y su actividad.

Tabla 1 Requisitos Funcionales de Seguridad
Diseño propio

Clase	RFS	Descripción	ODE		
			Comunicación con el servidor	Certificado digital	Autenticación
FCO: Comunicación	FCO_NRO	No repudio de origen	Si	Si	
	FCO_NRR	No repudio de recepción	Si	Si	

FSC: Soporte criptográfico	FSC_GCC	Gestión de claves criptográficas			Si
	FSC_OC	Operación criptográfica	Si	Si	Si

2.1.5 Diseño e Implementación Segura

La fase de diseño es la encargada de darle forma a los requisitos. Esta fase juega un papel muy importante cuando se da el diseño de los requisitos de seguridad. Es esencial diseñar un modelo de amenaza de aplicaciones seguras, este modelado es una técnica para identificar amenazas, vulnerabilidades y sus contramedidas (Figura 7). Una vez que se tienen los requisitos de seguridad y se tienen los diagramas de flujo se debe identificar los puntos desde donde el atacante puede entrar al sistema, al identificar dichos puntos se van a poder observar las posibles amenazas que un atacante puede explotar desde estos puntos. Los arboles de ataque pueden ser trazados para aclarar la compresión de la metodología del atacante. (Mead, 2008)

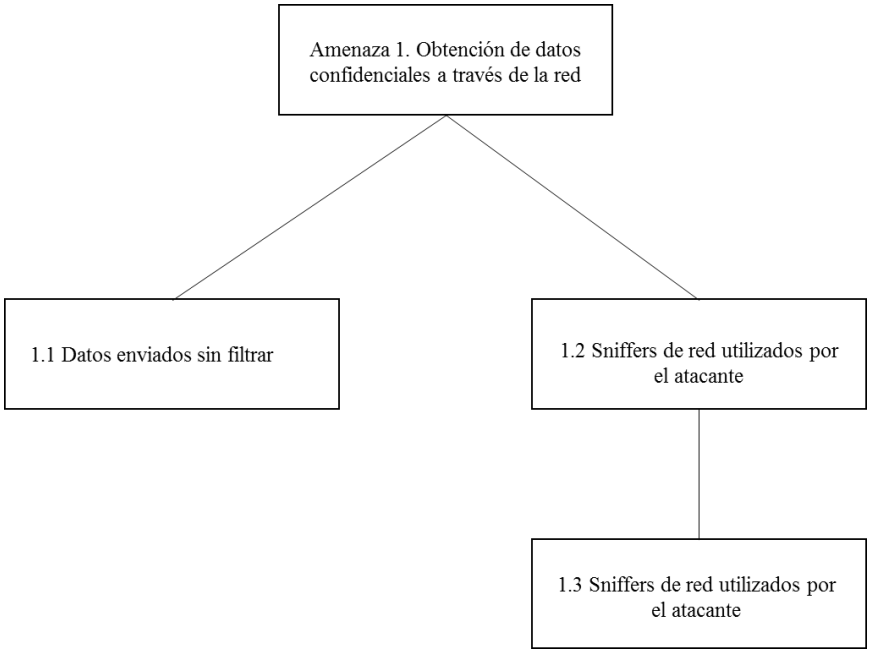

Figura 7 Árbol de ataque
Diseño propio

El análisis de las vulnerabilidades es una parte fundamental del modelado de amenazas. En la tabla 2 se observan algunas áreas relacionadas donde se pueden presentar vulnerabilidades, éstas pueden ocurrir en cualquier fase del ciclo de vida del software, pero es importante identificarlas en la fase de diseño.

Tabla 2 Vulnerabilidades comunes
Diseño propio

Área de vulnerabilidad	Tipo de vulnerabilidad
Sistema Operativo (SO)	Desbordamiento de búfer, punteros nulos, bloqueo de recursos, excepciones, etc.
Comunicación	No repudio de origen, No repudio de recepción etc.
Datos de base de datos / usuario	Tipos de datos no válidos, inyección de SQL, Cross Site Scripting, Rollback, integridad de datos, etc.
Criptografía	Gestión de claves, Operación criptográfica, etc.
Control de acceso	Política de control de acceso, autenticación de datos, política de control de flujo de información, etc.
Privacidad	Privilegios, anonimato, pseudo anonimato, etc.
Programación	Excepciones.

2.1.6 Pruebas de Seguridad y Despliegue

Las pruebas de seguridad son de vital importancia, ya que por medio de ellas se pueden identificar las fallas de seguridad antes de la liberación de la aplicación. En este punto el desarrollador debe pensar como un atacante y tratar de lanzar una serie de ataques los cuales le permitirán identificar errores en el sistema es por esto que para comprobar que el software cumple con los requisitos de seguridad se deben aplicar 1) Pruebas Funcionales y 2) Pruebas

basadas en riesgos. Las pruebas funcionales tratan de probar la aplicación de software con requisitos funcionales, éstos definen el comportamiento funcional de la aplicación para un estado específico. Las pruebas basadas en riesgos, se enfocan en los estados o comportamientos que no debe presentar una aplicación. Durante la aplicación de esta prueba se crean planes de prueba para componentes específicos de software que requieren seguridad. Una vez se tenga toda la información sobre amenazas de seguridad, vulnerabilidades, y las acciones para enfrentarlas, entonces se puede proceder a aplicar este tipo de pruebas (McGraw, 2006). En la siguiente tabla de describen algunas pruebas que se podrían aplicar para comprobar la seguridad de un aplicativo web.

Tabla 3 Tipos de pruebas
Diseño propio

Pruebas dirigidas	Pruebas realizadas por el equipo de pruebas de TI y el equipo de pruebas de penetración.
Pruebas externas	Pruebas realizadas en servidores externos y firewalls.
Pruebas internas	Pruebas para verificar las amenazas internas por el usuario autorizado.
Prueba ciega	Se examinan todas las acciones y procedimientos que un atacante real puede realizar.
Prueba de doble ciego	Pruebas ciegas realizadas por desarrolladores que los demás integrantes del equipo desconocen.

2.2 Guía del cuerpo de conocimientos de ingeniería de software (SWEBOK®)

La Guía del Cuerpo de Conocimientos de Ingeniería de Software (SWEBOK) describe los conocimientos generalmente aceptados sobre ingeniería de software. Sus 15 áreas de conocimiento resumen conceptos básicos e incluyen una lista de referencias que apunta a

información más detallada. Para la versión 3 de la guía se incluyen las siguientes áreas de conocimiento.

- Requisitos de Software
- Diseño de Software
- Construcción de Software
- Pruebas de Software
- Mantenimiento de Software
- Gestión de la configuración
- Gestión de la Ingeniería de Software
- Proceso de Ingeniería de Software
- Herramientas y métodos de la Ingeniería de Software
- Calidad del Software
- Práctica Profesional de la Ingeniería de Software
- Economía de la Ingeniería de Software
- Fundamentos de Computación
- Fundamentos Matemáticos
- Fundamentos de Ingeniería

Para el desarrollo de la estrategia, se abordará todo lo referente a la construcción de Software, ya que se busca fortalecer en el interior del departamento de sistemas, la construcción de software agregándole el componente de seguridad que se describirán con los estándares, guías y normas que se explicarán a continuación.

El término construcción de software se refiere a la creación detallada de software a través de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración. Esta área de conocimiento está vinculada con todas las demás, pero especialmente con las de Diseño de Software y Pruebas de Software porque el proceso de construcción del software involucra un diseño y pruebas de software significativo. El proceso utiliza la salida de diseño y proporciona una entrada para las pruebas. Los límites entre el diseño, la construcción

y las pruebas (si las hay) variarán dependiendo de los procesos del ciclo de vida del software que se utilicen en un proyecto. (SWEBOK, 2014)

2.3 Software Web Seguro

El software inseguro está afectando la infraestructura financiera, de salud, de defensa, de energía, entre otras. A medida que el software se vuelve más complejo y conectado, la dificultad de lograr la seguridad de las aplicaciones aumenta exponencialmente. Es por ello que el uso de buenas prácticas de seguridad durante el ciclo de vida del desarrollo de software minimiza los posibles riesgos a los que estarán expuestas las aplicaciones.

La fase de implementación de ciclo de vida del software logra el desarrollo de código de toda la aplicación. Hay varios aspectos relativos a la seguridad que deben desarrollarse en un patrón seguro correcto para lograr con los objetivos de seguridad durante la fase de implementación: (Higuera, 2014)

Gestión de excepciones: Como actúa la aplicación cuando falla una llamada al método y la información se muestra en mensajes de error a los usuarios finales. Si pasa información valiosa de excepción de nuevo al código de llamada. Si ayuda a los administradores a realizar análisis de causa raíz de la falla. La administración de excepciones se refiere a cómo se manejan las excepciones dentro de la aplicación.

Validación de datos de entrada y salida: Todas las entradas y salidas de la aplicación deben ser validadas. El contenido de todos los datos de entrada debe validarse para comprobar la longitud de los arrays, comprobar el contenido malintencionado, las fuentes de datos como las bases de datos y los recursos compartidos de archivos, etc.

Gestión de sesiones: La aplicación debe gestionar y proteger las sesiones de usuario mediante identificadores fuertes con números aleatorios sólidos, utilizando un nuevo identificador para cada nueva sesión, con un tiempo de espera de sesión predeterminado, etc.

Auditoría y Registro: La aplicación debe registrar y auditar eventos relacionados con la seguridad para informar cómo es su operación continuada. Registro se refiere a cómo la

aplicación publica información sobre su funcionamiento. La información a revelar debe ser el mínimo necesario.

Para esto es importante conocer los estándares y normas que se deben aplicar para garantizar la seguridad del aplicativo y de la información que este vaya a gestionar. Es por esto que existen organizaciones como OWASP (Open Web Application Security Project), SANS (SysAdmin Audit, Networking and Security Institute) y WASC (Web Application Security Consortium) que tienen como uno de sus objetivos aumentar la conciencia sobre la seguridad de las aplicaciones al identificar algunos de los riesgos más críticos que enfrentan las organizaciones.

2.4 Open Web Application Security Project – OWASP

Es un proyecto de código abierto que se dedica a documentar y combatir las causas que hacen que un software sea inseguro. La comunidad OWASP está formada por empresas, organizaciones educativas y particulares de todo mundo. Juntos constituyen una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías que se liberan y pueden ser usadas gratuitamente por cualquiera. A continuación se referencian los principales proyectos que se tendrán en cuenta para el desarrollo de la estrategia de seguridad (scmagazine, 2014):

2.4.1 OWASP Top 10 de Riesgos de seguridad de aplicaciones

OWASP Top 10 es un documento de los diez riesgos de seguridad más importantes en aplicaciones web según la organización OWASP (en inglés Open Web Application Security Project, en español Proyecto Abierto de Seguridad de Aplicaciones Web). Esta lista se publica y actualiza cada tres años por dicha organización.

El objetivo de este proyecto según la OWASP top 10, es crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones (OWASP, 2017). Para la versión del año 2017 se presentan los siguientes riesgos:

Tabla 4 OWASP Top 10
Tomado de (OWASP, 2017)

OWASP Top 10 – 2017	
A1	Inyección
A2	Autenticación rota y gestión de sesiones
A3	Cross-Site Scripting (XSS)
A4	Control de acceso roto
A5	Configuración incorrecta de seguridad
A6	Exposición de datos sensibles
A7	Insuficiente protección de ataque
A8	Falsificación de solicitudes entre sitios
A9	Uso de componentes con vulnerabilidades conocidas
A10	API desprotegidas

2.4.2 OWASP Guía para Construir Aplicaciones y Servicios Web Seguros

Este documento que proporciona orientación detallada sobre la seguridad de aplicaciones web, se ocupa de diferentes cuestiones desde la más antigua como la inyección SQL, hasta la más modernas como lo es la suplantación de identidad, manipulación de tarjetas de crédito, fijación del período de sesiones, falsificaciones de petición en sitios cruzados, el cumplimiento de las reglas y cuestiones de privacidad. En esta guía, se puede encontrar detalles sobre la seguridad de la mayoría de las formas de aplicaciones web y servicios, con una orientación muy práctica utilizando ejemplo en lenguajes de programación como PHP. De igual forma se tienen en cuenta las vulnerabilidades de seguridad mencionadas en el top 10 pero se

abordan de una forma más profunda. El modelado de Riesgo de Amenazas es el método más importante de mitigación en el desarrollo de aplicaciones web seguras en los últimos años. En la guía se introducen los conceptos básicos de Microsoft sobre el modelado de riesgo de amenazas, y los detalles de varias estrategias de otras empresas que se pueden tomar como referencia. Si se analiza detenidamente y se seleccionan los controles a través del modelado de riesgo de las amenazas, el resultado final será una implementación de sistemas que demostrablemente reducen los riesgos de negocio, que generalmente conduce a un aumento de la seguridad y la reducción de los fraudes y pérdidas. (OWASP, 2005)

2.5 ISO 17799

ISO 17799 es un marco de gestión de la Seguridad basado en riesgos que deriva directamente de los estándares AS/ NZS 4444 y BS 7799. Es un estándar internacional, muy utilizado por la mayoría de las organizaciones fuera de los Estados Unidos. Aunque algo más raro, algunas organizaciones en los Estados Unidos también utilizan ISO 17799, particularmente si poseen subsidiarias fuera del país. Los orígenes de la ISO 17799 se remontan a mediados de los 90, y algunos de los objetivos de control reflejan esta antigüedad - por ejemplo al llamar a las interfaces administrativas “puertos de diagnóstico”. Las organizaciones que utilizan ISO 17799 pueden usar OWASP como una guía detallada al seleccionar e implementar una amplia gama de controles de la ISO 17799, particularmente aquellos detallados en el capítulo de Desarrollo de Sistemas, entre otros. (ISO 17799, 2005)

2.6 COBIT 5 for Information Security

Esta guía tiene como finalidad proporcionar una guía práctica en la seguridad de la empresa, en todos sus niveles. Proporciona orientación para ayudar a los profesionales de TI y seguridad a entender, utilizar, implementar y dirigir actividades importantes relacionadas con la seguridad de la información y tomar decisiones más informadas al tiempo que mantiene conciencia sobre las tecnologías emergentes y las amenazas. La guía está dividida en tres secciones: Seguridad de la Información, Uso de Habilitadores de COBIT 5 para Implementar la Seguridad de la Información en la Práctica y Adaptación de COBIT 5 para la Seguridad de la Información al Entorno Empresarial. (ISACA, 2012)

2.7 ISO/IEC 27001

ISO/IEC 27001 es un estándar para la seguridad de la información (Information technology - Security techniques - Information security management systems - Requirements) aprobado y publicado como estándar internacional en octubre de 2005 por International Organization for Standardization y por la comisión International Electrotechnical Commission. Especifica los requisitos necesarios para establecer, implantar, mantener y mejorar un sistema de gestión de la seguridad de la información (SGSI) según el conocido como "Ciclo de Deming": PDCA - acrónimo de Plan, Do, Check, Act (Planificar, Hacer, Verificar, Actuar) (ISO27001, 2005).

2.8 ISO/IEC 27002

ISO/IEC 27002 (anteriormente denominada ISO 17799) es un estándar para la seguridad de la información publicado por la Organización Internacional de Normalización y la Comisión Electrotécnica Internacional. La versión más reciente es la ISO/IEC 27002:2013. ISO/IEC 27002 proporciona recomendaciones de las mejores prácticas en la gestión de la seguridad de la información a todos los interesados y responsables en iniciar, implantar o mantener sistemas de gestión de la seguridad de la información. La seguridad de la información se define en el estándar como "la preservación de la confidencialidad (asegurando que sólo quienes estén autorizados pueden acceder a la información), integridad (asegurando que la información y sus métodos de proceso son exactos y completos) y disponibilidad (asegurando que los usuarios autorizados tienen acceso a la información y a sus activos asociados cuando lo requieran)" (ISO27002, 2007).

2.9 CWE/SANS Top 25

Es un proyecto que trata de clasificar vulnerabilidades de Aplicaciones, en términos de su importancia relacionada con la frecuencia con la que se producen en las aplicaciones y el peligro de los ataques que pueden desencadenar Common Weakness Enumeration (CWE) es un diccionario de las debilidades de software comunes que pueden ocurrir en la arquitectura (Tabla 5), diseño, código o implementación del software que puede conducir a vulnerabilidades de seguridad explotables. CWE fue creado para:

Servir como un lenguaje común para describir las debilidades de seguridad del software.

Servir como un estándar de medición para herramientas de seguridad de software dirigidas a estas debilidades.

Proporcionar un estándar común de referencia para la identificación de debilidades, mitigación y esfuerzos de prevención. Las debilidades del software son fallas, errores, vulnerabilidades y otros errores en la implementación del software, el código, el diseño o la arquitectura que, si no se resuelven, podrían resultar en sistemas y redes vulnerables al ataque.

Las debilidades del software podrían ser por ejemplo: desbordamientos de búfer, problemas de estructura y validez; Manipulación de elementos especiales comunes; Errores de canal y ruta; Errores de la interfaz de usuario; Trayectoria y errores de equivalencia; Errores de autenticación; Errores de gestión de recursos; Verificación insuficiente de los datos; Evaluación e inyección de código; Y aleatoriedad y previsibilidad (Systems, 2015).

Tabla 5 CWE/SANS Top 25
Tomado de (Systems W. R., 2015)

Rango	Tabla	ID	Nombre
1	93.8	CWE-89	Neutralización inadecuada de elementos especiales usados en un comando SQL ('Inyección SQL')
2	83.3	CWE-78	Neutralización inadecuada de los elementos especiales utilizados en un comando del sistema operativo ('Inyección de comando del sistema operativo')
3	79.0	CWE-120	Copia de almacenamiento intermedio sin verificar el tamaño de la entrada ('Desbordamiento de búfer clásico')
4	77.7	CWE-79	Neutralización inadecuada de entrada durante la generación de páginas web ('Cross-Site Scripting')
5	76.9	CWE-306	Falta la autenticación para la función crítica
6	76.8	CWE-862	Falta la autorización

7	75.0	CWE-798	Uso de Credenciales Duras
8	75.0	CWE-311	Falta el cifrado de datos confidenciales
9	74.0	CWE-434	Carga sin restricciones del archivo con tipo peligroso
10	73.8	CWE-807	Confianza en insumos no confiables en una decisión de seguridad
11	73.1	CWE-250	Ejecución con privilegios innecesarios
12	70.1	CWE-352	Falsificación de solicitudes entre sitios (CSRF)
13	69.3	CWE-22	Limitación incorrecta de un nombre de ruta a un directorio restringido ('Trayectoria de ruta')
14	68.5	CWE-494	Descarga del código sin verificación de integridad
15	67.8	CWE-863	Autorización incorrecta
16	66.0	CWE-829	Inclusión de la funcionalidad de la esfera de control que no es de confianza
17	65.5	CWE-732	Asignación de permiso incorrecta para recurso crítico
18	64.6	CWE-676	Uso de la función potencialmente peligrosa
19	64.1	CWE-327	Uso de un algoritmo criptográfico roto o arriesgado
20	62.4	CWE-131	Cálculo incorrecto del tamaño del búfer
21	61.5	CWE-307	Restricción incorrecta de intentos de autenticación excesiva
22	61.1	CWE-601	Redireccionamiento de URL al sitio que no es de confianza ('Abrir redirección')
23	61.0	CWE-134	Cadena de formato no controlada
24	60.3	CWE-190	Desbordamiento de enteros o envolvente
25	59.9	CWE-759	Uso de un hash de una vía

2.10 WASC

El Consorcio de Seguridad de Aplicaciones Web (WASC), es una organización sin fines de lucro compuesto por un grupo internacional de expertos, profesionales de la industria y representantes de organizaciones que producen código abierto y ampliamente acordados estándares de seguridad para la World Wide Web. Como una comunidad activa, WASC facilita el intercambio de ideas y organiza varios proyectos de la industria. WASC constantemente publica información técnica, artículos aportados, directrices de seguridad y otra documentación útil. Empresas, instituciones educativas, gobiernos, desarrolladores de aplicaciones, profesionales de la seguridad y vendedores de software de todo el mundo utilizan los conceptos aportados por esta organización como apoyo en los desafíos presentados por la seguridad de las aplicaciones web.

La clasificación de amenazas WASC v2.0 es un esfuerzo cooperativo para aclarar y organizar las amenazas a la seguridad de un sitio web. Los miembros del Consorcio de Seguridad de Aplicaciones Web han creado este proyecto para desarrollar y promover la terminología estándar de la industria para describir estos problemas. Los desarrolladores de aplicaciones, los profesionales de seguridad, los proveedores de software y los auditores tendrán la capacidad de acceder a un lenguaje y definiciones coherentes para problemas relacionados con la seguridad web. En la siguiente tabla se describe la clasificación de amenazas (WASC, 2011).

Tabla 6 Clasificación de amenazas WASC v2.0.
Tomado de (WASC, 2011)

Ataques	Debilidades
Abuso de funcionalidad	Configuración de la aplicación
Fuerza bruta	Indexación de directorios
Desbordamiento de búfer	Permisos de sistema de archivos incorrectos
Spoofing de contenido	Manejo incorrecto de entrada

Credencial / Sesión de Predicción	Manejo incorrecto de salida
Scripting entre sitios	Fuga de información
Cross-Site Solicitud de Falsificación	Indexación Insegura
Negación de servicio	Anti-automatización insuficiente
Huellas dactilares	Autenticación insuficiente
Formato de cadena	Autorización insuficiente
Contrabando de respuesta HTTP	Recuperación de contraseña insuficiente
División de respuestas HTTP	Validación insuficiente del proceso
Contrabando de solicitudes HTTP	Expiración de sesión insuficiente
División de solicitud HTTP	Protección insuficiente de la capa de transporte
Desbordamientos de enteros	Configuración errónea del servidor
Inyección LDAP	
Inyección de Comando de Correo	
Inyección de Byte Nulo	
Comando de OS	
Camino Traversal	

Ubicación de recursos predecibles	
Inclusión de archivos remotos (RFI)	
Desvío de enrutamiento	
Sesión de fijación	
Abuso de SOAP Array	
Inyección de SSI	
Inyección SQL	
Abuso del redirector de URL	
Inyección XPath	
Error de atributo de XML	
XML Entidades externas	
Expansión de entidades XML	
Inyección XML	
Inyección XQuery	

2.11 OASIS WS-Security

WS-Security es un protocolo de comunicaciones que suministra un medio para aplicar seguridad a los Servicios Web. La versión 1.1 fue publicada en el año 2006, su objetivo principal es el de proporcionar una base técnica para la implementación de funciones de seguridad tales como integridad y confidencialidad en los mensajes que implementan aplicaciones de servicios web de nivel superior. El estándar WS-Security 1.1 se compone de las siguientes especificaciones (Oasis, 2006):

WS-Security Core Specification 1.1: Describe mejoras a la mensajería SOAP para proporcionar integridad y confidencialidad del mensaje. Los mecanismos especificados se pueden utilizar para acomodar una amplia variedad de modelos de seguridad y tecnologías de cifrado. De igual forma proporciona un mecanismo de propósito general para asociar tokens de seguridad con contenido de mensaje.

Username Token Profile 1.1: Describe cómo utilizar el UsernameToken con la especificación WSS: SOAP Message Security [WSS]. Es decir, establece cómo un consumidor de servicio web puede proporcionar un UsernameToken como medio para identificar al solicitante mediante "nombre de usuario" y, opcionalmente, usar una contraseña para autenticar esa identidad al productor de servicios web.

X.509 Token Profile 1.1: Describe el uso del marco de autenticación X.509 con la especificación Seguridad de Servicios Web: SOAP Message Security specification [WS-Security]. Un certificado X.509 especifica un enlace entre una clave pública y un conjunto de atributos que incluye (al menos) un nombre de asunto, un nombre de emisor, un número de serie y un intervalo de validez.

SAML Token profile 1.1: Describe cómo utilizar las aseveraciones V1.1 y V2.0 de Security Assertion Markup Language (SAML) con la especificación de Seguridad de Servicios Web (WSS): SOAP Message Security V1.1.

Kerberos Token Profile 1.1: Describe el uso de tokens Kerberos [Curb] con respecto a la especificación WSS: SOAP Message Security [WSS]. Es decir, define cómo codificar los

tickets Kerberos y adjuntarlos a los mensajes SOAP. Además, específica cómo agregar firmas y encriptación al mensaje SOAP, de acuerdo con WSS: SOAP Message Security, que utiliza y hace referencia a los tokens de Kerberos.

Rights Expression Language (REL) Token Profile 1.1: Describe cómo utilizar las expresiones de derechos ISO / IEC 21000-5 con la especificación de seguridad de servicios web (WSS).

SOAP with Attachments (SWA) Profile 1.1: Describe cómo utilizar el estándar de seguridad de mensajes de SOAP OASIS Web Services Security [WSS-Sec] con mensajes SOAP con adjuntos [SwA]. Es decir, establece cómo un consumidor de servicios web puede proteger los archivos adjuntos de SOAP mediante SOAP Message Security para la integridad de los datos adjuntos, la confidencialidad y la autenticación de origen y cómo un receptor puede procesar tal mensaje.

2.12 WS-Policy

WS-Policy es una especificación que forma parte de la familia de especificaciones de tecnologías basadas en servicios web del W3C. Esta especificación permite a los programadores de servicios web anunciar sus políticas relativas a seguridad, calidad de servicio, etc. y a los clientes de servicios web especificar sus requisitos de calidad de servicio, seguridad, latencia, etc. La versión actual de la política es la 1.5 publicada en el año 2007.

Web Services Policy 1.5 proporciona un modelo de propósito general y la sintaxis correspondiente para describir las políticas de las entidades en un sistema basado en servicios web. El Marco de políticas de servicios web define un conjunto básico de construcciones que pueden utilizarse y extenderse por otras especificaciones de servicios Web para describir una amplia gama de requisitos y capacidades de servicio.

Una política es una colección de alternativas de política. Una alternativa de política es una colección de afirmaciones de política. Una afirmación de política representa un requisito, capacidad u otra propiedad de un comportamiento. Una expresión de política es una representación de Infoset de XML de su política, ya sea en una forma normal o en su forma compacta equivalente. Algunas afirmaciones de políticas especifican requisitos y capacidades tradicionales que se manifiestan en los mensajes intercambiados (por ejemplo, esquema de

autenticación, selección de protocolo de transporte). Otras afirmaciones de política no tienen ninguna manifestación de cable en los mensajes intercambiados, pero son relevantes para la selección y uso de servicios. Web Services Policy 1.5 - Framework proporciona un lenguaje de políticas único que permite expresar y evaluar de una manera coherente ambos tipos de aserciones. (W3C, 2007)

2.13 Norma ISO/IEC 15408, Common Criteria

El Common Criteria permite la comparabilidad entre los resultados de las evaluaciones de seguridad independientes. El Common Criteria lo hace proporcionando un conjunto común de requisitos para la funcionalidad de seguridad de los productos de TI y para las medidas de aseguramiento aplicadas a estos productos de TI durante una evaluación de seguridad. Estos productos de TI pueden implementarse en hardware, firmware o software.

El proceso de evaluación establece un nivel de confianza de que la funcionalidad de seguridad de estos productos de TI y las medidas de aseguramiento aplicadas a estos productos de TI, cumplen estos requisitos. Los resultados de la evaluación pueden ayudar a los consumidores a determinar si estos productos de TI satisfacen sus necesidades de seguridad. El Common Criteria es útil como guía para el desarrollo, evaluación y / o adquisición de productos de TI con funcionalidad de seguridad. De igual forma es intencionalmente flexible, permitiendo que una gama de métodos de evaluación se apliquen a una gama de propiedades de seguridad de una gama de productos de TI. (Criteria, 2017)

2.14 Estado del Arte

Para hablar de seguridad en el desarrollo de aplicaciones web, es necesario resaltar algunos trabajos realizados sobre el tema para así tener un referente del estado actual del objeto de estudio. Es por ello que a continuación, se mencionarán algunos artículos y tesis de grado donde se abordó el tema de desarrollo de aplicaciones web seguras y se hace mención de las técnicas aplicables para este tipo de desarrollo.

En el artículo WEBLOG: A Declarative Language for Secure Web Development, los autores Hinrichs et al. (2013) de la Universidad de Illinois en Chicago, proponen WEBLOG como un

lenguaje declarativo para el desarrollo de aplicaciones web, diseñado para eliminar automáticamente varias vulnerabilidades de seguridad comunes a las aplicaciones web actuales. En este artículo, se presenta WEBLOG, se detallan las vulnerabilidades de seguridad que elimina y se discute la forma como lo realizan.

Durante la VI Conferencia Internacional de Ingeniería Web ACM Ge et al. (2006) en su artículo Agile Development of Secure Web Applications, afirman que un sistema de seguridad es el que se aplica a las necesidades específicas. Ofrecer un sistema seguro, y particularmente una aplicación web segura, no es fácil. La integración de métodos de desarrollo de sistemas de información de carácter general con actividades de desarrollo de la seguridad podría ser un medio útil para superar estas dificultades. Los procesos ágiles, como la programación extrema, tienen un interés creciente en el desarrollo de software. Lo más importante para las aplicaciones web, es que los procesos ágiles fomentan y aceptan el cambio de requisitos, lo cual es una característica deseable para el desarrollo de aplicaciones web. Se presenta un proceso ágil para desarrollar aplicaciones web seguras. La contribución de la investigación no es el desarrollo de un nuevo método o proceso que aborde los problemas de seguridad. En su lugar, se investigan métodos de desarrollo de sistemas de información de propósito general (por ejemplo, Desarrollo orientado a las características (FDD)) y métodos de seguridad maduros, es decir, análisis de riesgo, con esto se logra una integración para abordar el desarrollo de aplicaciones web seguras. Las principales características de este enfoque son (1) un proceso capaz de hacer frente a los desafíos clave del desarrollo de aplicaciones web, a saber, la disminución de los tiempos de ciclo de vida y los requisitos que cambian con frecuencia; Y (2) un enfoque iterativo del análisis de riesgos que integra el diseño de seguridad a lo largo del proceso de desarrollo.

En la conferencia ACM SIGSAC sobre seguridad de las comunicaciones informáticas, Reischuk, Schröder y Gehrke (2013) en su artículo DEMO: Secure and Customizable Web Development in the SAFE Activation Framework, proponen una demostración de SAFE con algunas de sus más nuevas características de seguridad. SAFE es un marco para el desarrollo de aplicaciones Web modernas con consistencia de estado automatizada, seguridad reforzada en varios niveles y diseño para la personalización y extensibilidad de la Web. Con la creciente complejidad del desarrollo de aplicaciones Web (extensibles) basadas en datos, en particular en

términos de gestión coherente de datos con múltiples clientes, la preservación de la propiedad y la privacidad de los datos, proponen un demo de un marco de aplicación Web centrada en datos y seguridad con especificaciones declarativas para muchas características Web modernas la cual será de considerable interés para la comunidad de seguridad.

Li y Xue (2014) de la Universidad de Vanderbilt en su artículo A Survey on Server-Side Approaches to Securing Web Applications, describieron que las aplicaciones web son una de las plataformas más frecuentes para la información y la prestación de servicios a través de Internet en la actualidad. A medida que se utilizan cada vez más para servicios críticos, las aplicaciones web se han convertido en un objetivo popular y valioso para los ataques de seguridad. Aunque se han desarrollado un gran número de técnicas para fortalecer las aplicaciones web y mitigar los ataques lanzados contra ellas, ha habido poco esfuerzo dedicado a establecer conexiones entre estas técnicas y construir el panorama general de la investigación de seguridad de aplicaciones web. De igual forma examinan el área de seguridad de las aplicaciones web desde el lado del servidor, con el objetivo de sistematizar las técnicas existentes en un panorama general que promueve la investigación futura. En primer lugar, presentan los aspectos únicos del desarrollo de aplicaciones web que causan problemas inherentes en la creación de aplicaciones web seguras. Describen tres vulnerabilidades de seguridad comúnmente vistas en aplicaciones web: vulnerabilidades de validación de entrada, vulnerabilidades de administración de sesiones y vulnerabilidades de lógica de aplicación, junto con ataques que explotan estas vulnerabilidades. Se organizan las técnicas existentes a lo largo de dos dimensiones: (1) las vulnerabilidades y ataques de seguridad que abordan y (2) el objetivo de diseño y las fases de una aplicación web durante la cual se pueden llevar a cabo. Estas fases son la construcción segura de nuevas aplicaciones web, análisis / pruebas de seguridad de aplicaciones web heredadas y protección en tiempo de ejecución de aplicaciones web heredadas.

En la XVII Conferencia ACM sobre Seguridad de las Computadoras y las Comunicaciones, Li et al. (2010) en su artículo SecTag: A Multi-Policy Supported Secure Web Tag Framework, describen los problemas de seguridad a los que se enfrentan las aplicaciones web tradicionales ya que a menudo se enfrenta a un problema de acoplamiento estrecho entre la lógica de control

de acceso y la lógica empresarial. Es difícil configurar y modificar políticas de control de acceso después de implementar un sistema. Es por esto que se presenta SecTag, un marco de etiquetas web seguro con varias directivas, para abordar este problema. Definieron una serie de atributos seguros de uso general que satisfacen la demanda de control de acceso en la capa de presentación web. También diseñaron un conjunto de etiquetas interactivas de alta seguridad, que encapsulan funciones seguras para proporcionar componentes seguros reutilizables para el desarrollo web.

En la Universidad de Carolina del Norte en Charlotte, Xie et al. (2011) en su artículo titulado ASIDE: IDE Suppor t for Web Application Security afirman que muchas de las vulnerabilidades de seguridad de aplicaciones de hoy en día son introducidas por desarrolladores de software que escriben un código de seguridad inseguro. Esto puede deberse a una falta de comprensión de las prácticas de programación seguras y a la falta de atención de los desarrolladores sobre la seguridad. Mucho trabajo en la seguridad del software se ha concentrado en la detección de vulnerabilidades de software a través de técnicas de análisis automatizadas. Aunque son eficaces, no son suficientes. Es por esto que se debe proponer aumentar la conciencia de los desarrolladores y promover la práctica de la programación segura recordando interactivamente a los programadores el uso de buenas prácticas de programación seguras dentro de los Entornos de Desarrollo Integrados (IDE). Es por esto que, se ha implementado un controlador de prueba de concepto para Eclipse y Java. Los resultados de las evaluaciones iniciales muestran que este enfoque puede detectar y abordar las vulnerabilidades comunes de las aplicaciones web y puede servir como una ayuda eficaz para los programadores. El enfoque propuesto también complementa eficazmente las prácticas existentes de seguridad de software y aumenta significativamente la productividad del desarrollador.

En la Universidad de Houston Rahman, Nguyen y Yang (2006) a través de su artículo Developing Certificate-based Projects for Web, afirman que el creciente número de aplicaciones que se están utilizando en Internet para intercambiar datos, que van desde chat en línea hasta números de tarjetas de crédito y otro tipo de información sensible. El acompañamiento del uso generalizado de redes es el problema omnipresente de ataques maliciosos en las aplicaciones y las redes subyacentes. Los datos transmitidos sin una protección adecuada están sujetos a acceso

no autorizado y manipulación indebida. Para fortalecer una aplicación contra ataques, es importante integrar medidas de seguridad adecuadas. Es por esto que este artículo presenta proyectos de seguridad web utilizando mecanismos basados en certificados para proteger aplicaciones web. Los proyectos implican imitar ataques y proteger los recursos de esos ataques. Los proyectos implican el uso de tecnologías de seguridad como Secure Socket Layer (SSL), certificados digitales y HTTPS (Secure HyperText Transport Protocol) para asegurar canales de comunicación. Al integrar los proyectos en los cursos de desarrollo web, los instructores pueden proporcionar ejercicios prácticos que ayudan a los estudiantes a adquirir un conocimiento real de cómo se realizan estos ataques y cómo funcionan las medidas de control.

Walden, y Frank (2007) en el Departamento de Ciencias de la Computación de la Universidad Northern Kentucky, en su artículo Web Application Security Tutorial, describen al desarrollo de aplicaciones web como un área grande y creciente de empleo para los graduados de ciencias de la computación. Mientras los graduados han aprendido a diseñar e implementar aplicaciones web que funcionan correctamente con los insumos esperados, pocos han aprendido a diseñar e implementar software que sea seguro contra vulnerabilidades comunes de aplicaciones web. Las vulnerabilidades de seguridad más comunes en el software son las secuencias de comandos entre sitios, la inyección de código SQL y el manejo de errores por ejemplo en lenguajes como PHP. Estas vulnerabilidades pueden permitir a los atacantes acceder a las aplicaciones sin permiso, obtener información confidencial como tarjetas de crédito o números de seguridad social, robar mercancía o transferir fondos de sitios web comerciales. El número de vulnerabilidades descubiertas cada año ha aumentado a un ritmo exponencial desde el año 2000. En este artículo, se describe cómo los atacantes explotan las vulnerabilidades comunes de las aplicaciones web y muestran demostraciones en vivo de dichos ataques. De igual forma se muestra a los participantes cómo enseñar a sus estudiantes a diseñar y escribir código seguro de que es inmune a estos ataques. El artículo presenta recursos que los participantes pueden utilizar para incorporar la seguridad de aplicaciones web en la programación, bases de datos, desarrollo web, y cursos de seguridad de la información.

La XI Conferencia Internacional sobre la World Wide Web Scott y Sharp (2002) genera la publicación del artículo Abstracting Application-Level Web Security, en el que deducen que la

seguridad de la web a nivel de aplicación se refiere a vulnerabilidades inherentes al código de una aplicación web en sí misma (independientemente de las tecnologías en las que se implemente o de la seguridad de la base de datos del servidor web en la que se construye). En los últimos tiempos, las vulnerabilidades a nivel de aplicación se han explotado con graves consecuencias: por ejemplo, los piratas informáticos han engañado a los sitios de comercio electrónico enviando productos sin costo, los nombres de usuario y contraseñas han sido recolectados e información confidencial (como direcciones y números de tarjetas de crédito) se ha filtrado. En este trabajo los autores investigan nuevos métodos y técnicas que abordan el problema de la seguridad en la web a nivel de aplicación. (1) Describen un mecanismo de estructuración escalable que facilite la abstracción de las políticas de seguridad a partir de las grandes aplicaciones web desarrolladas en entornos heterogéneos de múltiples plataformas; (2) Presentan un programa que ayuda a los programadores a desarrollar aplicaciones seguras que son resistentes a una amplia gama de ataques comunes y (3) Informe de los resultados y la experiencia derivada de la aplicación de estas técnicas.

En el simposio ACM 2007 sobre computación aplicada, Yao, Koglin, Bertino y Tamassia (2007) en su trabajo titulado Decentralized Authorization and Data Security in Web Content Delivery, abordan el rápido desarrollo de los servicios web, o más ampliamente, las arquitecturas orientadas a servicios (SOA), las cuales han llevado a más organizaciones a mover contenidos y aplicaciones a la Web. Las aplicaciones web permiten disfrutar de una variedad de servicios, por ejemplo, la traducción de textos en otros idiomas y la conversión de un documento de un formato a otro. En este artículo se aborda el problema de mantener la integridad y confidencialidad de los datos en la entrega de contenido web cuando se necesitan modificaciones de contenido dinámico. Se propone un modelo flexible y escalable para la entrega de contenido seguro basado en el uso de roles y certificados de roles para gestionar los intermediarios web. Los proxies se coordinan para procesar y entregar contenidos, y la integridad del contenido entregado se aplica mediante una estrategia descentralizada. Para lograr esto, se utiliza una tabla de búsqueda de roles distribuidos y un mecanismo de enrutamiento basado en números de rol. Proporcionan un protocolo seguro eficiente, iDeliver, para el procesamiento y entrega de contenido, y también se describe un método para actualizar de forma segura las tablas de búsqueda de roles. La solución también se aplica al problema de

seguridad en los flujos de trabajo basados en web, por ejemplo, mantener la integridad de datos en el comercio automatizado, la autorización de contratos y la gestión de la cadena de suministro en grandes organizaciones.

Okubo y Tanaka (2008), en la XV Conferencia sobre lenguajes de patrones de programas presentan el artículo Web Security Patterns for Analysis and Design, en donde afirman que el análisis de requisitos de seguridad desempeña un papel muy importante en el desarrollo de software seguro, es difícil ya que requiere mucha experiencia en seguridad y poder humano. Se necesitan patrones sencillos y prácticos de requisitos de seguridad. Presentaron un enfoque de análisis visualizado para obtener los requisitos de seguridad mediante la extensión de casos de uso indebido, y se encontró que algunos de sus resultados pueden ser candidatos a patrón. Este artículo propone 8 nuevos patrones de requisitos de seguridad web con el enfoque de análisis propuesto. Los patrones proporcionados dan a analistas una manera de encontrar un patrón apropiado para una meta de seguridad específica. Están relacionados con soluciones de seguridad, y también producen algunas posibilidades de diseño de seguridad.

Whitson (2008) del Departamento de Informática de la Universidad de Texas, asegura que la mayoría de personas están de acuerdo en que la programación de aplicaciones se está moviendo a Internet. Este tipo de aplicación es intrínsecamente inseguro, ya que permite el acceso al servidor en la solicitud HTTP inicial. Se desarrolló una estrategia de seguridad bastante bien pensada para las aplicaciones Web de segunda generación mediante la creación de una caja de seguridad alrededor de la solicitud HTTP (hasta que el usuario se autenticó), utilizando la seguridad del servidor estándar para proteger el procesamiento y, finalmente, extender las ideas de la computación distribuida Para asegurar el acceso a los datos. La programación de aplicaciones Web se está moviendo actualmente a una arquitectura orientada a servicios (SOA) que incluye un amplio uso de servicios Web. Hay una furia en el mundo de UNIX para perfeccionar el Bus de Servicios Empresariales (ESB) y en el mundo de Microsoft para terminar con Windows Communications Foundation (WCF). Y la belleza de SOA es que una aplicación web desarrollada con WCF podrá acceder a los datos de un servicio Web Java. SOA soporta un entorno de desarrollo de aplicaciones distribuido que es verdaderamente independiente del proveedor.

En la III Conferencia Anual sobre Desarrollo del Currículo de Seguridad de la Información Walden y Frank (2006), dieron a conocer el diseño de un curso de ingeniería de software seguro que enseñará a los estudiantes cómo incorporar la seguridad durante todo el ciclo de vida del desarrollo de software. La clase servirá como una piedra angular para un nuevo certificado de posgrado en ingeniería de software seguro. Este trabajo describe los objetivos de clase, el diseño de la clase y los materiales que se desarrollaran para enseñar la ingeniería de software segura. Se crearon diez módulos para cubrir los temas centrales en seguridad de software. Cada módulo cubrirá uno o más objetivos de la clase y consistirá de materiales explicativos y asignaciones para dar a los estudiantes la oportunidad de aplicar sus aprendizajes en un contexto pequeño. La clase también incorporará un proyecto de desarrollo web basado en equipo que los estudiantes trabajarán a lo largo del semestre para obtener experiencia aplicando principios de seguridad a Un proyecto a gran escala.

Sathiaseelan, Rabara y Martin (2009), durante la II Conferencia Internacional de Ciencias de la Interacción: Tecnología de la Información, Cultura y Humanidad, afirmaron que la tecnología de servicios web es el campo emergente en el área de ingeniería web que enfatiza el diseño de aplicaciones basadas en servicios web, modelos y estándares arquitectónicos, plataformas tecnológicas de servicios web, herramientas y especificaciones para la construcción de servicios Web estándar y tecnologías WS-Security. Las tecnologías de servicios web cambian drásticamente la industria del software desarrollando e integrando servicios y aplicaciones web empresariales para permitir a los usuarios acceder a ellos. La plataforma de servicios web es una plataforma de integración para aplicaciones y datos de las empresas. Los investigadores realizan esfuerzos para sacar a la luz el estado de la técnica en tecnologías de servicios web y tecnologías de seguridad para servicios web. La composición del servicio web es la combinación de servicios web de diferentes proveedores de servicios para brindar mayor sofisticación y valor agregado a los usuarios; siendo otra área prometedora para la investigación y el desarrollo. Existen varias iniciativas de investigación en el campo de los servicios Web para aplicaciones empresariales, sin embargo, ninguna de las iniciativas se ha intentado para aplicaciones de instituciones educativas que utilizan tecnología de servicios Web. Un estudio revela que existen varios enfoques que proporcionan servicios web seguros. Sin embargo, no hay ningún intento realizado hasta ahora para asegurar los servicios en varios niveles. Este

documento propone un marco de seguridad multi-nivel (MLSF) para instituciones académicas que ofrece servicios web compuestos seguros, tales como servicio de admisión, servicio de pago de tasas, servicio de biblioteca, servicio personalizado para los estudiantes, personal y gestión, etc. Varios niveles como nivel de usuario, nivel de dispositivo, nivel de administrador de servicios web y nivel de servicios institucionales.

Durante la Conferencia sobre Diseño, Automatización y Pruebas en Europa, Kazmierski y Yang (2003) en su artículo A Secure Web-Based Framework for Electronic System Level Design, presentan el concepto de una implementación segura de un marco de diseño electrónico distribuido basado en la web. La arquitectura de servidores webserver-toolserver de dos niveles se ha extendido para soportar bases de datos permanentes para el desarrollo de diseño distribuido. En la aplicación de ejemplo el framework fue desarrollado en Java y la configuración de los servidores puede ser en Linux, MS Windows o Sun-SPARC con la configuración para trabajar servlet de Java. La comunicación entre applets y el servidor web se controla usando el sistema 'javakey' para 'firmar' el applet. Los datos sólo se proporcionan a un applet con la firma correcta. Una conexión segura garantiza privacidad, integridad de transmisión, autenticación y autorización. Sin embargo, el vínculo todavía puede no ser seguro y privado. Por lo tanto, el sistema también implementa conexiones seguras utilizando el paquete JSSE (Java Secure Socket Extension) con RMI. Esto permite crear sockets RMI para usar el protocolo SSL.

Durante el XLIV Simposio técnico sobre educación en ciencias de la computación de ACM, Zhu, Lipford, Heather y Chu (2013) presentan el trabajo titulado Interactive Support for Secure Programming Education, en el que afirman que las fallas de software son una causa raíz de muchas de las vulnerabilidades actuales de seguridad de la información. El actual énfasis del currículo en los temas tradicionales de seguridad de la información no aborda esta causa raíz. Es por ello que proponen educar a los estudiantes sobre técnicas de programación seguras a través del soporte de herramientas interactivas en el Entorno de Desarrollo Integrado (IDE). Este enfoque puede complementar otros programas curriculares mediante la enseñanza y el refuerzo continuo de las prácticas a lo largo de las tareas de programación. En este artículo, se evalúa la herramienta de prototipo, ASIDE, que proporciona advertencias de seguridad

instantáneas, explicaciones detalladas de vulnerabilidades y generación de código. Se presentan los resultados de un estudio observacional de 20 estudiantes de un curso avanzado de programación web. Los resultados proporcionan evidencia temprana de que la herramienta podría potencialmente ayudar a los estudiantes a aprender y practicar la programación segura en el contexto de sus asignaciones de programación.

En el simposio ACM 2004 sobre Computación Aplicada, Chan (2004) presenta su investigación en el artículo titulado Cookies On-the-move: Managing Cookies on a Smart Card, en el que sostiene que a pesar del uso generalizado y la adopción de cookies como base para las aplicaciones web para mantener el estado de la información, las cookies presentan algunos problemas de diseño que aún no se han abordado completamente. El hecho de que las cookies estén almacenadas en la memoria del cliente significa que están estrechamente acopladas a la máquina que está interactuando con el servidor web. A menudo, estas cookies son iniciadas por aplicaciones web para identificar las preferencias e identificaciones del usuario. A medida que el usuario se desplaza a través de diferentes máquinas para acceder al mismo sitio, la información previamente grabada se pierde y la aplicación web no tiene forma de restaurar el estado, a menos que el usuario vuelva a visitar la misma máquina cliente, donde se establecieron las cookies originales. En este documento se presenta una solución novedosa para abordar la necesidad de que las cookies sean "móviles" aprovechando la tarjeta inteligente para administrar las cookies, con el beneficio de la movilidad en un bolsillo. Se describe el diseño e implementación del marco de CookiesCard que utiliza la tarjeta inteligente como medio de almacenamiento seguro y móvil para administrar cookies personalizadas. El artículo presenta el desarrollo del proxy CookiesCard que interactúa directamente con la tarjeta inteligente para proporcionar la gestión de cookies, mientras que actúa como un intermediario entre el navegador del cliente y un servidor web.

En el documento titulado An Access Control Language for Web Services, escrito por Sirer y Wang (2002), se presenta un enfoque para la especificación formal y aplicación de las políticas de seguridad en las implementaciones de servicios web. Los servicios en red en general y los servicios web en particular, requieren cantidades extensas de código para garantizar que los clientes respeten las restricciones de integridad del sitio. Se proporciona un lenguaje mediante

el cual estas restricciones pueden ser expresadas y aplicadas de forma automática, portátil y eficiente. Las políticas de seguridad del sistema se especifican en un lenguaje basado en la lógica temporal y son procesadas por un motor de aplicación para proporcionar código de control de acceso específico para sitios y plataformas. Este código se integra con un servidor web y bibliotecas específicas de la plataforma para aplicar la política especificada en un servicio web determinado. Este enfoque desacopla la especificación de políticas de seguridad de implementaciones de servicios, proporciona un modelo de control de acceso obligatorio para servicios web y logra un buen rendimiento. Se demostró que hasta un 22% del código en un módulo de servicio web tradicional está dedicado a la funcionalidad de comprobación de seguridad, incluyendo comprobaciones para la secuenciación de clientes y validación de parámetros. Se demuestra que el prototipo de implementación de lenguaje, WebGuard, permite a los programadores web reducir significativamente la cantidad de código de seguridad que necesitan desarrollar manualmente. La calidad del código generado por WebGuard a partir de las especificaciones formales de la política es competitiva con la latencia del código artesanal a un pequeño porcentaje.

En la XIV conferencia académica internacional: visualización de futuros entornos mediáticos, Salminen et al. (2010) en su trabajo Developing Client-side Mashups: Experiences, Guidelines and the Road Ahead proponen el uso de mashups de software que combinan el contenido de varios sitios web con una experiencia integrada son una tendencia popular. Sin embargo, los métodos y herramientas para crear mashups todavía están poco desarrollados, y hay poco apoyo de ingeniería detrás de ellos. En este artículo, se ofrece información sobre el desarrollo de mashup basado en experiencias prácticas en la implementación de varias aplicaciones de mashup de muestra y herramientas para crearlas. A diferencia de las herramientas de desarrollo de mashup más utilizadas comúnmente, centradas en el servidor, se centran principalmente en el desarrollo de mashup de cliente. Se han agrupado diferentes experiencias, que pueden servir como un punto de partida útil para el diseño de nuevos mashups. El objetivo a largo plazo es facilitar el desarrollo de aplicaciones de mashup robustas, seguras y convincentes y, en general, facilitar la transición hacia el desarrollo de software basado en web.

En el trabajo Position Paper: ¿Why Are There So Many Vulnerabilities in Web Applications?, los autores Du et al. (2011) exponen durante el taller de 2011 sobre Paradigmas de Seguridad, que a medida que la Web se ha vuelto cada vez más ubicua, el número de ataques a aplicaciones web ha aumentado sustancialmente. Según informes, más del 80 por ciento de las aplicaciones web han tenido al menos una vulnerabilidad seria. Este porcentaje es alarmantemente ya que es más alto que las aplicaciones tradicionales. Algo se debe estar haciendo mal en el desarrollo web. Sobre la base de esta investigación, se ha formulado la siguiente posición: al elegir el marco sin estado para la Web, se han ignorado una serie de propiedades de seguridad que son esenciales para las aplicaciones. Como resultado, la Base de Computación Confiable (TCB) de la Web tiene debilidades significativas. Para construir aplicaciones seguras con estado sobre un TCB debilitado, los desarrolladores tienen que implementar una lógica de protección adicional en sus aplicaciones web, lo que dificulta el desarrollo y es propenso a errores, causando así una serie de problemas de seguridad en aplicaciones web.

Smith y Dehlinger (2014) en la Conferencia sobre Investigación en Adaptación y Convergencia, aseguraron que los sistemas basados en Web impregnan la sociedad, apoyando las aplicaciones críticas para el negocio que solicitan / almacenan con frecuencia la información personal del cliente, requiriendo niveles cada vez más altos de aseguramiento de la información. Los programadores web novatos, con poca o ninguna habilidad de programación segura, sin saberlo desarrollan aplicaciones web maduras con vulnerabilidades de seguridad, comprometiendo así la integridad de la aplicación. Como resultado, se han desarrollado varias herramientas de seguridad de análisis estático para señalar posibles vulnerabilidades de seguridad. Sin embargo, estas herramientas son difíciles de usar, divorciadas de los entornos de desarrollo integrado de software (IDE) y siguen siendo desconocidas para los desarrolladores principiantes. En este trabajo se aporta un complemento de Eclipse que permite el análisis estático del código fuente de PHP utilizando las herramientas existentes directamente dentro de un IDE común para permitir a los desarrolladores principiantes crear aplicaciones web más seguras. Se hacen dos reclamaciones para la extensión de SSVChecker. En primer lugar, se integra a la perfección en un IDE común lo que es fácil de usar para los desarrolladores principiantes. Segundo, proporciona funcionalidad que aprovecha múltiples herramientas para reducir los falsos

positivos reportados y enfocar mejor a los desarrolladores principiantes en vulnerabilidades potenciales de seguridad.

En el VII Simposio ACM sobre Seguridad de la Información, la Computación y las Comunicaciones Cao et al. (2012) en su trabajo Virtual Browser: a Virtualized Browser to Sandbox Third-party JavaScripts with Enhanced Security, proponen la creación de un navegador virtualizado para un entorno de pruebas JavaScript con seguridad mejorada, afirmando que los JavaScripts de terceros no sólo ofrecen características ricas a la Web y sus aplicaciones, sino que también introducen nuevas amenazas. Estos scripts no pueden ser completamente confiados y ejecutados con los privilegios dados a los sitios web de host. Debido a la virtualización incompleta y la falta de seguimiento de todos los flujos de datos, todos los enfoques existentes sin soporte de sandbox nativo pueden proteger sólo un subconjunto de JavaScripts de terceros y son vulnerables a ataques codificados en HTML / Java no estándar. Estos enfoques analizarán código JavaScripts independientemente en el lado del servidor sin tener en cuenta las peculiaridades de análisis no estándar del lado del cliente. Al mismo tiempo, las cajas de arena nativas son vulnerables a ataques basados en errores de motor de JavaScript nativos desconocidos. En este trabajo, se propone un entorno virtualizado a nivel de explorador completo dentro de los navegadores existentes para ejecutar código de terceros no confiable. Este enfoque soporta funciones de lenguaje JavaScript más completas, incluidas aquellas funciones difíciles de proteger, como with y eval. Dado que el navegador virtual no se basa en el comportamiento de análisis de explorador nativo, no hay posibilidad de que se ejecuten ataques a través de peculiaridades del navegador. Además, dado que los JavaScripts de terceros se ejecutan en el explorador virtual en lugar de los navegadores nativos, es más difícil para los atacantes explotar vulnerabilidades desconocidas en el motor de JavaScript nativo. En este diseño, primero se aísla completamente el navegador virtual de los componentes nativos del navegador y luego se introduce la comunicación agregando flujos de datos cuidadosamente examinados para seguridad.

Mundada et al. (2016) en su artículo Half-Baked Cookies: Hardening Cookie-Based Authentication for the Modern Web, aseguran que los sitios web modernos utilizan múltiples cookies de autenticación para permitir a los visitantes del sitio diferentes niveles de acceso. La

complejidad de las aplicaciones web modernas puede dificultar que un programador de aplicaciones web se asegure de que el uso de cookies de autenticación no introduce vulnerabilidades. Incluso cuando un programador tiene acceso a todo el código fuente, este análisis puede ser un reto; El problema se vuelve aún más fastidioso cuando los programadores web se emparedan de las bibliotecas para implementar la autenticación. En el artículo se reúne una lista de comprobación para los programadores web modernos para verificar que el mecanismo de autenticación basado en cookies está implementado de forma segura. De igual forma, se desarrolla una herramienta llamada Newton, para ayudar a un programador de aplicaciones web a identificar cookies de autenticación para partes específicas del sitio web y verificar que se implementan de forma segura de acuerdo con la lista de comprobación. Se aplican pruebas de análisis sobre 149 sitios utilizando Newton, incluyendo el Alexa top-200 y muchos otros sitios populares a través de una gama de categorías incluyendo búsqueda, compras y finanzas. Se evidencia que 113 de ellos incluyendo sitios de alto perfil como Yahoo, Amazon y Fidelity eran vulnerables a ataques de secuestro. Muchos sitios web ya han reconocido y corregido las vulnerabilidades que se encontraron usando Newton y reportadas a ellos.

Durante la Conferencia de Desarrollo del Currículo de Seguridad de la Información de 2015 Simpkins et al. (2015), presentan su trabajo A Course Module on Web Tracking and Privacy, en el cual sostienen que las empresas rastrean el comportamiento en línea de los usuarios con fines de lucro, utilizando tecnologías como cookies de navegador, cookies flash, AdID y guías web. Estos datos son típicamente anónimos, pero todavía hay preocupaciones de privacidad. Los usuarios también pueden no saber que están siendo rastreados de esta manera. Hay regulaciones en los EE.UU. y la legislación en la UE en relación con el seguimiento del comportamiento en línea, y varios métodos para prevenir o limitar este seguimiento. Este documento cubre un módulo de curso que incluye una introducción al tema de seguimiento web y privacidad, y dos estudios de caso. Este módulo de curso puede ser adoptado en cursos de seguridad web que introducen cuestiones legales y de privacidad relacionadas con la web.

Durante la Conferencia ACM SIGSAC de 2016 sobre Seguridad de la Computación y las Comunicaciones Grubbs et al. (2016), desarrollaron en su trabajo Breaking Web Applications Built On Top of Encrypted Data, un enfoque sistemático para analizar las aplicaciones cliente-

servidor que intentan ocultar los datos confidenciales de los usuarios de servidores no confiables. Lo aplicaron a Mylar, un marco que utiliza cifrado de búsqueda multi-clave (MKSE) para construir aplicaciones Web en la parte superior de los datos cifrados. Se demostró que (1) el modelo Popa-Zeldovich para MKSE no implica seguridad contra ataques pasivos o activos; (2) Las aplicaciones Web basadas en Mylar revelan los datos y consultas de los usuarios a servidores pasivos y activos; Y (3) Mylar es genéricamente inseguro frente a ataques activos debido a defectos de diseño del sistema. Como resultado se muestra que el problema de proteger aplicaciones cliente-servidor contra servidores maliciosos es desafiante y aún no resuelto. Se concluye con lecciones generales para los diseñadores de sistemas que se basan en la preservación de propiedades o el cifrado de búsqueda para proteger los datos de servidores no confiables.

Bhargavan et al. (2007), en su artículo Secure Sessions for Web Services, abordan el problema de asegurar secuencias de mensajes SOAP intercambiados entre los servicios web y sus clientes. La norma WS-Security define los mecanismos básicos para proteger el tráfico SOAP, un mensaje a la vez. Sin embargo, para los servicios web típicos, utilizar WS-Security independientemente para cada mensaje es bastante ineficiente; Además, a menudo es importante asegurar la integridad de toda una sesión, así como de cada mensaje. Para estos fines, las especificaciones recientes proporcionan otros mecanismos de nivel SOAP. WS-SecureConversation define contextos de seguridad, que pueden usarse para asegurar sesiones entre dos partes. WS-Trust especifica cómo se emiten y se obtienen los contextos de seguridad. Se desarrolló una semántica para los principales mecanismos de WS-Trust y WS-SecureConversation, expresada como una biblioteca para TulaFale, un lenguaje de scripting formal para protocolos de seguridad. Se modelan protocolos típicos basados en estos mecanismos y se prueban automáticamente sus principales propiedades de seguridad. También se discute informalmente algunas dificultades y limitaciones de estas especificaciones.

Adida (2008) en su trabajo SessionLock: Securing Web Sessions against eavesdropping, afirma que las sesiones web pueden ser secuestradas fácilmente por un intruso de la red en los ataques que han llegado a ser designado "sidejacking". El aumento de las redes inalámbricas omnipresentes, a menudo desprotegido en la capa de transporte, ha agravado significativamente

este problema. Mientras que SSL puede proteger contra el espionaje, sus desventajas de usabilidad a menudo lo hacen inadecuado cuando los datos no se consideran altamente confidenciales. La mayoría de los servicios de correo electrónico basados en la web, por ejemplo, usan SSL únicamente en su página de inicio de sesión y, por lo tanto, son vulnerables a los ataques laterales. Es por esto que propone SessionLock, un enfoque sencillo para proteger las sesiones web contra el espionaje sin extender el uso de SSL. SessionLock es fácilmente implementado por los desarrolladores web utilizando sólo JavaScript y la lógica del servidor simple. Su impacto en el rendimiento es insignificante, y todos los principales navegadores web son compatibles. Utilizando la función de identificador de fragmentos HTTP existente para crear un canal seguro de cliente entre HTTPS y HTTP, se ha diseñado e implementado SessionLock, una forma de proteger las sesiones HTTP simples de las escuchas espontáneas. Esta propuesta es relativamente fácil de implementar, especialmente en el caso de aplicaciones con gran capacidad de AJAX, como Gmail. De hecho, parece que las sesiones HTTP de Gmail pueden protegerse con un código mínimo de nivel de aplicación web y una sobrecarga de rendimiento insignificante. De igual forma, se puede observar el atractivo de las soluciones, como SessionLock, que utilizan sólo las modificaciones en el nivel de la aplicación web: pueden ser implementadas inmediatamente por los desarrolladores web. De esta forma se espera que la exploración de características de seguridad mejoradas utilizando sólo la pila Web existente sea informativa para la mejora del navegador web como una plataforma extensible para la seguridad.

Mavromoustakos et al. (2016) en la IV Conferencia Internacional sobre Seguridad de la Información y las Redes, a través de su trabajo Causes and Prevention of SQL Injection Attacks in Web Applications, abordan el tema de los ataques de inyección en las aplicaciones web y afirman que la inyección de SQL es una de las principales amenazas a la seguridad de las aplicaciones web. Los atacantes tratan de obtener acceso no autorizado a la base de datos, que tiene información vital y privada de los usuarios. Muchos investigadores han proporcionado varias técnicas y prácticas para proteger las aplicaciones web de los atacantes. Hay una plétora de técnicas disponibles para realizar la inyección de SQL y por lo general no todo el mundo está familiarizado con cada ataque. Por lo tanto, este tipo de ataque sigue siendo el más frecuente.

En este trabajo, se presentan los tipos de ataques de inyecciones SQL y la mayoría de las formas dominantes para prevenirlos durante el desarrollo de aplicaciones web.

Durante el Taller sobre Servicios Web Seguros (2005), Gutiérrez, Fernandez-Medina y Piattini (2005) presentan el trabajo titulado Web Services Enterprise Security Architecture: A Case Study, en el que se afirma que la seguridad en los Servicios Web (WS) es un aspecto crucial para las tecnologías basadas en este paradigma para ser completamente adoptado por la industria. Como consecuencia de ello, han surgido muchas iniciativas en los últimos años fijando como objetivo principal la estandarización de los factores de seguridad relacionados con este paradigma. De hecho, en los últimos años, los consorcios más importantes de Internet, como IETF, W3C u OASIS, están produciendo un gran número de estándares de seguridad basados en WS. A pesar de este crecimiento, todavía no existe un proceso que oriente a los desarrolladores en la crítica tarea de integrar la seguridad en todas las etapas del ciclo de vida del desarrollo de software basado en WS. Este proceso debe facilitar a los desarrolladores en las actividades de especificación de los requisitos de seguridad específicos de servicios web, diseño de arquitectura de seguridad basada en servicios web y selección de estándares de seguridad de servicios web, integración e implementación. En este artículo se presenta brevemente el proceso PWSSec (proceso de seguridad de servicios web) que se compone de tres etapas: WSSecReq (requisitos de seguridad de servicios web), WSSecArch (arquitectura de seguridad de servicios web) y WSSecTech (tecnologías de seguridad de servicios web) respectivamente. También se ofrece una explicación detallada de la fase WSSecArch (etapa de seguridad de servicios web) destinada a diseñar la arquitectura de seguridad basada en servicios web.

En la XIII conferencia internacional de Middleware ACM / IFIP / USENIX, Oliveira, Laranjeiro y Vieira (2012), a través de su investigación WSFAggressor: An Extensible Web Service Framework Attacking Tool, presentan una herramienta para probar la seguridad de los marcos de servicios web. La herramienta implementa un gran conjunto de tipos de ataque, se definen basados en estudios previos de investigación de seguridad, herramientas de prueba existentes y experiencia de campo. La motivación es que los desarrolladores crean frecuentemente servicios web basados en la suposición de que los marcos subyacentes son seguros, lo que no siempre es

así. A pesar de la evidente necesidad de seguridad en las plataformas que soportan servicios, las herramientas existentes de pruebas de seguridad son muy limitadas. En la práctica, la mayoría de las herramientas se centran en vulnerabilidades de nivel de aplicación, y los pocos que permiten que las plataformas de pruebas implementen un conjunto muy limitado de tipos de ataque. Hasta donde se sabe, la herramienta incluye más ataques que cualquier otra herramienta existente. Además, al implementar una arquitectura extensible (basada en plugins), la herramienta puede ampliarse fácilmente con ataques adicionales, soportando también una gran variedad de configuraciones de prueba. Los resultados muestran que puede utilizarse para revelar problemas críticos de seguridad en marcos bien conocidos.

Bates et al. (2017) en la 26ª Conferencia Internacional sobre la World Wide Web, sostienen que detectar y explicar la naturaleza de los ataques en los servicios web distribuidos es difícil ya que determinar la naturaleza de la actividad sospechosa requiere seguir el rastro de un atacante a través de una cadena de componentes de software heterogéneos incluyendo equilibradores de carga, proxies, nodos de trabajo y servicios de almacenamiento. Desafortunadamente, las soluciones forenses existentes no pueden proporcionar el contexto necesario para vincular eventos a través de flujos de trabajo complejos, particularmente en instancias en las que se necesita semántica de capa de aplicación (por ejemplo, consultas SQL, RPC) para comprender el ataque. En este trabajo, se presenta un enfoque basado en la procedencia transparente para la auditoría de servicios web a través de la introducción de las funciones de procedencia de la red (NPFs). Los NPFs son una arquitectura distribuida para capturar datos detallados de procedencia para componentes de servicios web, aprovechando la clave de que la mediación de los protocolos de una aplicación puede ser utilizada para inferir sus actividades sin requerir instrumentación invasiva o cooperación con desarrolladores. Se diseña e implementa NPF con la consideración de la complejidad de los modernos servicios web basados en la nube, y se evalúa la arquitectura en contra de una variedad de aplicaciones incluyendo DVDStore, RUBiS y WikiBench para demostrar que el sistema propuesto impone tan poco como un promedio de 9.3% en las conexiones para cargas de trabajo realistas. Finalmente, se consideran varios escenarios en los que el sistema puede ser utilizado para explicar concisamente los ataques. NPF permite el despliegue sin complicaciones de la auditoría semántica rica basada en la procedencia para flujos de trabajo de aplicaciones complejas en la nube.

Neville-Neil (2007) en su trabajo Building Secure Web Applications, describe que en estos días de phishing y anuncios casi diarios de robo de identidad a través de grandes pérdidas de datos, parece casi ridículo hablar de asegurar la Web. En este punto, la mayoría de la gente parece lista para lanzar sus manos en la idea o para bloquear un pequeño componente que pueden controlar con el fin de mantener el caos percibido en la red. Se describen principalmente tres problemas de seguridad al momento de desarrollar aplicaciones web; El primer problema es la autenticación. ¿Cómo sabe la aplicación quién está accediendo y qué se les permite acceder. El problema dos es la capacidad de un atacante para engañar a los usuarios, una vez que se han autenticado, en hacer el trabajo en nombre del atacante. Y el último problema es el riesgo que supone alojar UGC (contenido generado por el usuario) en un sitio Web. En este artículo se analiza cada uno de estos problemas, el estado actual de las soluciones existentes y los problemas no resueltos que siguen afectando a las grandes aplicaciones Web.

Near y Jackson (2016) en su artículo Finding Security Bugs in Web Applications Using a Catalog of Access Control Patterns, proponen una técnica libre de especificaciones para encontrar comprobaciones de seguridad faltantes en aplicaciones web utilizando un catálogo de patrones de control de acceso en el que cada patrón modela un caso de uso de control de acceso común. La implementación denominada, SPACE, comprueba que cada exposición de datos permitida por el código de una aplicación coincida con una exposición permitida de un patrón de seguridad en el catálogo descrito. La única entrada proporcionada por el usuario es una asignación de tipos de aplicación a los tipos del catálogo; El resto del proceso es totalmente automático.

Busch et al. (2012) en su trabajo Towards Model-driven Development of Access Control Policies for Web Applications, introducen una notación basada en UML para modelar gráficamente los aspectos de seguridad de los sistemas de una manera simple e intuitiva y un proceso impulsado por el modelo que transforma las especificaciones gráficas de las políticas de control de acceso en XACML. Estas políticas XACML se traducen a continuación en FACPL, un lenguaje de políticas con una semántica formal y las políticas resultantes se evalúan mediante una herramienta de software basada en Java.

En el Taller CoNEXT sobre estudiantes de 2014, Ikram et al. (2014) aseguraron que varios componentes web y JavaScripts se han utilizado para recopilar información personal identificable que resulta en preocupaciones de privacidad. Aunque varias herramientas de preservación de la privacidad se han propuesto para limitar la publicidad en línea y el seguimiento de su uso ha sido limitada y en su mayoría la audiencia de tecnología inteligente. Además de mantenimiento pobre y manual de lista de filtros y configuración confusa, estas herramientas de preservación de la privacidad tienen, posiblemente, problemas de usabilidad y de intrusión. Entre otros, el bloqueo de fuerza bruta de todos los JavaScripts en un sitio web, puede resultar en la rotura de funcionalidades, por lo tanto, la experiencia del usuario web se ve afectada. En este trabajo, se propone un marco para cuantificar la intrusividad de JavaScripts con el objetivo final de medir la usabilidad de las herramientas de preservación de la privacidad. Se postula que los JavaScripts intrusivos llevan características distintas que podrían usarse para diferenciarlos de los JavaScripts funcionales, es decir, los scripts que se usan genéricamente para mejorar la experiencia web del usuario. Se propone una metodología de medición que puede separar automáticamente el seguimiento y la privacidad intrusiva JavaScripts de los JavaScripts funcionales. Esta metodología supone sólo un conocimiento parcial de los scripts de intrusos de privacidad.

Miller y Connolly (2015) en su artículo Introduction to the Special Issue on Web Development, sostienen que a pesar de su predominio en la informática, el desarrollo web está subrepresentado en los programas informáticos de computación y la investigación de la educación en computación. Este trabajo da un paso hacia la mejora de su representación con tres artículos sobre educación para el desarrollo web. Basándose en diversos métodos de una variedad de contextos, los artículos abordan los retos de la enseñanza del desarrollo web y las dificultades comunes que los estudiantes encuentran al aprender conceptos particulares. Los tres artículos identifican el desarrollo web como un camino prometedor para motivar a los estudiantes en su estudio de la informática.

En el XLVI Simposio Técnico ACM sobre Educación en Ciencias de la Computación, Whitney et al. (2015) afirmaron que muchas de las vulnerabilidades de seguridad comunes en el software actual pueden evitarse con las prácticas estándar de codificación segura. Los estudiantes de

ciencias de la computación que se convertirán en los desarrolladores de ese software necesitan aprender sobre esas prácticas para que puedan prevenir tales vulnerabilidades. Muchos programas informáticos están abordando esta necesidad a través de conferencias adicionales, cursos electivos o enfoques más holísticos para integrar la seguridad a través de los currículos. Se está explorando un enfoque complementario, integrando la educación de codificación segura en el IDE para proporcionar una oportunidad de aprendizaje en el contexto de la escritura de código. En este trabajo, se informa sobre dos estudios de campo utilizando una herramienta IDE en un curso avanzado de programación web. El resultado indica que la herramienta puede aumentar la conciencia de los estudiantes y el conocimiento de la programación segura, pero para ser más eficaz, los instructores deben incentivar su uso a través de métodos en la clase y el calendario cuidadoso de su introducción.

En el artículo Quite a Mess in My Cookie Jar!: Leveraging Machine Learning to Protect Web Authentication Calzavara et al. (2014) aseguraron que las defensas basadas en el navegador han sido recientemente defendidas como un mecanismo eficaz para proteger las aplicaciones web contra las amenazas de secuestro de sesión, fijación y ataques relacionados. En los enfoques existentes, todas estas defensas dependen en última instancia de la heurística del lado del cliente para detectar automáticamente las cookies que contienen información de sesión, para luego protegerlas contra el robo o uso no intencional. Si bien es claramente crucial para la eficacia de los mecanismos de defensa resultantes, estas heurísticas no han sido sometidas a ninguna evaluación rigurosa de su adecuación. En este artículo, se realiza la primera evaluación formal, basada en un conjunto de cookies de oro que se recopilaron de 70 sitios web populares del ranking de Alexa. Para obtener el conjunto de oro, se diseñó un procedimiento semiautomático que se basa en una novedosa noción de token de autenticación, que se introduce para capturar múltiples esquemas de autenticación web. Se probaron las defensas basadas en navegador existentes en la actualidad contra el conjunto de oro, desvelando varios errores tanto en la heurística adoptada como en los métodos utilizados para evaluarlos. Se propone un nuevo método de detección basado en el aprendizaje supervisado, donde el conjunto de oro se utiliza para formar un clasificador binario, e informar sobre la evidencia experimental de que el método supera a las propuestas existentes. Curiosamente, la clasificación resultante, junto con la

experiencia práctica en la construcción del conjunto de oro, proporciona una nueva visión sobre cómo se implementa la autenticación web en la práctica.

Witschey et al. (2015) durante la X Reunión Conjunta sobre Fundamentos de Ingeniería de Software, abordaron el tema del desarrollo de software web seguro en su trabajo Quantifying Developers' Adoption of Security Tools y explicaron gracias a encuestas aplicadas a diferentes desarrolladores, que las herramientas de seguridad podrían ayudar a los desarrolladores a encontrar vulnerabilidades críticas, pero estas herramientas siguen siendo infrautilizadas. Al encuestar a desarrolladores de 14 compañías y 5 listas de correo sobre sus razones para usar y no usar herramientas de seguridad. Los treinta y nueve predictores resultantes del uso de herramientas de seguridad proporcionan ideas tanto esperadas como inesperadas. Como se esperaba, los desarrolladores que perciben que la seguridad es importante son más propensos a usar herramientas de seguridad que aquellos que no lo hacen. Pero esa no era la predicción más fuerte del uso de herramientas de seguridad, sino la capacidad de los desarrolladores de observar a sus compañeros usando herramientas de seguridad. Las herramientas de seguridad son una parte importante del desarrollo de software seguro, pero muchos desarrolladores no las utilizan, incluso cuando creen que la seguridad es importante. Esto deja el software menos seguro de lo que podría ser. En este artículo, se describe una cuantificación de los factores que influyen en la adopción de herramientas de seguridad. Los resultados, producto de un proyecto de investigación a largo plazo y de métodos mixtos, proporcionan a los expertos en herramientas y a los responsables de la formulación de políticas una mayor comprensión de por qué las herramientas se adoptan o no. Aunque los fabricantes de herramientas y los encargados de la formulación de políticas tienen más control sobre algunos factores, los resultados proporcionan una perspectiva holística a través de la cual estos interesados pueden comprender mejor la adopción. Se espera que futuras aplicaciones de los hallazgos formulados resulten en que más desarrolladores utilicen herramientas de seguridad, ayudando a aumentar la seguridad del software.

De Groef, Massacci y Piessens (2014) en su trabajo NodeSentry: Least-privilege Library Integration for Server-side JavaScript, argumentan que Node.js es un popular framework de servidor de JavaScript con un tiempo de ejecución eficiente para arquitecturas basadas en

eventos basadas en la nube. Su fuerza es la presencia de miles de bibliotecas de terceros que permiten a los desarrolladores crear y desplegar aplicaciones rápidamente. Estas mismas bibliotecas son una fuente de amenazas de seguridad, ya que una vulnerabilidad en una biblioteca puede (y en algunos casos) comprometer todo el servidor. Para esto con el fin de apoyar la integración de las bibliotecas menos privilegiadas, desarrollaron NodeSentry, la primera arquitectura de seguridad para JavaScript del lado del servidor. Esta infraestructura de cumplimiento de políticas soporta un fácil despliegue de técnicas de endurecimiento web y políticas de control de acceso sobre las interacciones entre las bibliotecas y su entorno, incluyendo cualquier biblioteca dependiente. En el artículo se discute la implementación de NodeSentry, y se presenta su evaluación práctica. Para cientos de clientes concurrentes, NodeSentry tiene la misma capacidad y rendimiento que Node.js. Sólo a gran escala, cuando Node.js se cede a una carga pesada, NodeSentry muestra una sobrecarga limitada.

En la Conferencia Internacional Conjunta IEEE / WIC / ACM 2009 sobre Inteligencia Web y Tecnología de Agentes Inteligentes Basilico, Gatti y Amigoni (2009) proponen el desarrollo de sistemas autónomos que patrullan entornos para detectar intrusos como un tema de creciente relevancia en aplicaciones de seguridad. Un aspecto importante de estos sistemas es la estrategia de patrullaje; A saber, la determinación de dónde moverse para detectar convenientemente las intrusiones. Si bien una gran parte de las estrategias de patrullaje propuestas hasta ahora adoptan algún tipo de movimientos aleatorios, las estrategias determinísticas pueden ser útiles en algunas situaciones de interés. En este artículo se propone un enfoque para encontrar una estrategia determinista que permita al agente de patrulla detectar siempre un intruso que intenta entrar en un entorno. El problema se formula como la determinación de un camino cíclico que visita, bajo restricciones temporales, todos los vértices de un gráfico que representa el entorno. Se propone un algoritmo de resolución, se estudian sus propiedades y se realiza una evaluación experimentalmente.

En el ámbito local se pueden citar los siguientes trabajos donde se trata el objeto de estudio, ya sea desde el punto de vista de la seguridad como desde el desarrollo de aplicaciones web.

López, Suárez y Meneses (2011) en su artículo Firma digital: instrumento de transmisión de información a entidades financieras, abordan el tema de la seguridad de la información y

afirman que la tecnología está presente en la mayoría de aspectos de la vida cotidiana, en esta investigación se presenta un avance en temas como encriptación y seguridad, aprovechar las oportunidades y mejorar la competencia empleando herramientas como la firma digital, la cual genera nuevas formas de acceder a los clientes con garantías de seguridad en transmisión de información y en el mejoramiento de procesos bancarios. La firma digital es equivalente a la firma manuscrita y permite incorporar las garantías básicas de seguridad de: autenticidad, confidencialidad, integridad y no repudio. Además, identifica (con una llave criptográfica) a una persona autora y emisora (certificada) de un documento informático. El empleo de firmas digitales certificadas puede generar impactos importantes en innovación para las empresas, en especial para las empresas del sector financiero donde es alta la exposición al riesgo.

Chamorro (2011) en su tesis de maestría titulada Modelo para la evaluación en seguridad informática a productos software, basado en el estándar ISO/IEC 15408 Common Criteria, afirma que la seguridad en las tecnologías de la información y comunicaciones (TICs), se hace tan indispensable como su funcionalidad misma. Preservar la disponibilidad, integridad y confidencialidad, de sus datos y operaciones, es un reto que se hace cada día más complejo, por su misma evolución y los riesgos que cada día se vuelven más sofisticados, al estar cada vez los usuarios mejor conectados y menos controlados. Actualmente existen diferentes estándares, modelos, sistemas de gestión y buenas prácticas que promueven la seguridad de la información en las compañías y tecnologías, dentro del más relevante es el estándar ISO/IEC 15408 Common Criteria, el cuál es un acuerdo internacional entre diferentes organizaciones de todo el mundo para que con base al cumplimiento de funciones y niveles de evaluación, se garantice diseño, desarrollo y puesta en producción con medidas de seguridad adecuadas para el mercado, entidades vigilantes y la comunidad en general. En Colombia, este estándar no es muy común y en Latinoamérica no existe hasta el momento un laboratorio o centro de investigación que certifique la aplicación de este estándar, pero que a medida que la globalización y las apuestas productivas del país se enfocan al desarrollo y producción de tecnología, se hace necesario adoptarlo. Se realizó entonces un análisis de riesgo a un conjunto de sistemas software seleccionados de acuerdo a los requerimientos de ley en Colombia, con el fin de determinar qué tan distantes están del cumplimiento del estándar ISO/IEC 15304 y sus falencias generales en seguridad. El resultado no fue bueno, y refleja la falta de documentación y detalle en las

funciones de seguridad del software que se desarrolla en las empresas seleccionadas y la no prevención de incidentes de seguridad ante las amenazas reales de un ambiente de producción.

Velasco (2011) en su modelo para la evaluación y selección de un software de seguridad para controlar el ciclo de vida de la identidad digital, asegura que las organizaciones y grandes grupos empresariales en su búsqueda por garantizar la seguridad, cumplimiento y control de los distintos accesos a sus aplicaciones de negocio, han abocado múltiples esfuerzos para encontrar una solución de software dé seguridad que permita de una manera eficiente y segura, garantizar el Ciclo de vida de la Identidad digital de sus empleados, contratistas y externos en la organización. En este documento se pretende ofrecer a las organizaciones un modelo para la evaluación y selección de un Software de Seguridad, a través de la definición de la Arquitectura del Sistema, gracias a la aplicación de unas plantillas preestablecidas que permiten de una manera mucho más formal, las esquematización y caracterización de las variables y atributos más significativos de la plataforma. El modelo permitirá a las organizaciones evaluar las diferentes plataformas de Gestión de Identidad, a partir de siete agrupadores que se componen por un total de ciento siete preguntas definidas y caracterizadas dentro del modelo; lo cual permite a las empresas llegar a una calificación y valoración del Sistema por medio del diligenciamiento de la matriz de calificación, la cual es diligenciada y calificada por el panel de expertos. El modelo desarrollado se aplicó al caso de estudio del Grupo Empresarial Coomeva y las conclusiones fueron comparadas con el proceso que normalmente se utiliza para la evaluación y selección de este tipo de proyectos de software.

Mogollón (2012) realiza un análisis sobre métodos de cifrado en su tesis de grado Análisis de viabilidad de la aplicación de métodos de cifrado al Framework EFI, y afirma que la atmósfera de amenazas para aplicaciones de Internet está en constante cambio y por lo tanto los avances e investigaciones también lo deben estar pero esto ya no depende tanto de las innovaciones por parte de los investigadores como estudios individuales sino también sobre los avances por parte de los atacantes por eso se vuelve cada vez más complejo la investigación en esta disciplina y al mismo tiempo se convierte en una necesidad, sobre todo cuando se tiene un software el cual maneja datos tan delicados como lo son la información de un usuario tal como lo hace el Framework EFI y por esto surge la idea de hacer un estudio de vulnerabilidades del EFI y así

mismo corregirlas, puesto que en los últimos años ha tenido bastante acogida, diferentes actualizaciones y por su estructura podría llegar a ser un Framework bastante apetecido por los desarrolladores. En este trabajo de grado se plantea la necesidad de dicho estudio y la implementación de diversas técnicas para mejorar el Framework desde el punto de vista de la seguridad y así brindar un producto más eficiente y confiable a los usuarios a nivel de cifrado de datos y autenticación.

Cacua (2012) en su trabajo de grado Implementación de Mecanismos para el Manejo de Errores, Auditoría y Generación de Logs al Framework EFI, trata el tema de seguridad web en el Framework EFI y propone la mejora de su seguridad por medio de la implementación de mecanismos de manejo de errores y generación de logs de auditoría y monitoreo, esto originado para la generación y lucha contra el software inseguro. Para dar cumplimento a este objetivo, se aplican las herramientas brindadas por el Open Web Application Security Project (OWASP) el cual es un espacio abierto de la comunidad dedicada a la búsqueda y la lucha contra las causas del software inseguro, seguido se seleccionaron mecanismos adecuados para la generación de logs en lenguaje Java tomando como base la estabilidad de los algoritmos y las opciones funcionales que brindan, después se implementan estrategias para el manejo de excepciones al Framework EFI también basándose en librerías Java que facilitan el proceso de manejo de errores para poder brindar flujos alternativos en el momento en el que se genera un error o excepción en la aplicación, y por último se incorporan mecanismos de generación de logs al Framework EFI, también basándose en brindar diferentes opciones al usuario del Framework a la hora de generar los logs, tomando como principio la integridad de los logs generados para prevenir modificaciones y eliminaciones causales o motivadas.

Bolaños (2013) en su tesis de maestría, aborda el tema de las arquitecturas orientadas a servicios (SOA) y la necesidad de las empresas por implementar un estilo de arquitectura que les permita mejorar la interoperabilidad entre los diferentes sistemas de información, el uso de este tipo de arquitectura mejora la seguridad de las aplicaciones web. El método MeMSOA propuesto en la tesis implementa las mejores prácticas de los métodos de evaluación de arquitecturas de software y valoración inspirado en referentes de madurez de calidad de software utilizado para determinar el nivel de madurez y capacidad de las organizaciones que desarrollan software. Los

métodos tomados como referencia son el resultado de una comparativa basada en los criterios de evaluación de cada uno, con lo que se espera diseñar un método de evaluación ágil para soportar la evaluación de la madurez de principios y patrones SOA. Para realizar la valoración se llevó a cabo un análisis del conjunto de patrones que se requieran analizar. Seguido, los patrones seleccionados fueron agrupados de acuerdo a los principios de SOA, esto, con el objetivo de establecer una caracterización que permita identificar los atributos a evaluar. Los resultados obtenidos a partir del método desarrollado pueden ser utilizados como línea base (base line) o punto de partida para realizar el plan de mejora que permitan contrarrestar los hallazgos que se evidencien en una evaluación.

Martínez et al. (2010) en su artículo Seguridad basada en parámetros SIM para entornos de comercio electrónico móvil, abordan el tema de la seguridad en las aplicaciones de comercio electrónico y afirman que los requerimientos de seguridad son los más exigentes en el dominio del comercio electrónico. Cuando se habla de comercio electrónico móvil los requisitos a nivel de seguridad no solo conservan su nivel de exigencia, sino que se debe mantener un equilibrio entre el grado de seguridad que se requiere y las capacidades de los dispositivos, tanto a nivel hardware como de usabilidad. Estas características exigen el diseño de modelos con un esquema simple de autenticación y autorización transparente para los usuarios, que además garantice la integridad de la información que se intercambia durante cualquier transacción electrónica. Como respuesta a esta necesidad, el Grupo de Interés en el Desarrollo de Aplicaciones Móviles e Inalámbricas W@PColombia, ha desarrollado la plataforma P3SIM con el objeto de brindar las facilidades necesarias para la construcción de aplicaciones móviles seguras basadas en parámetros SIM; a través de un Framework, un ambiente de compilación y de simulación como sus principales componentes, la plataforma P3SIM combina las ventajas de identificación que proporciona el módulo SIM con las capacidades que en materia de seguridad ofrecen API como SATSA y JavaCard en el entorno Java ME, una de las plataformas más utilizadas en el desarrollo de aplicaciones para dispositivos móviles. Igualmente, a través del desarrollo de un prototipo aplicado al contexto del comercio electrónico móvil, no solo se demuestran las facultades de la plataforma para operar en ambientes seguros sino también su capacidad de adaptación a los requerimientos de seguridad fijados por el entorno.

Rosero y Guzmán (2011) en su artículo Arquitectura de aprendizaje para el manejo de riesgo de falla en ambientes de composición de servicios web, presentan una arquitectura de aprendizaje basada en el aprendizaje de máquinas inductivas, la cual hace posible obtener datos durante el proceso de ejecución de servicios Web. Estos datos son analizados y con base a ellos, se genera una serie de información que se interpreta en riesgos de falla, ocasionados por los servicios Web en cualquier parte de su ejecución. Esta información es aprendida por el sistema, evitando la generación de errores en la toma de decisiones para la composición de servicios y por lo tanto, realizar una composición de servicios más precisa, utilizando aquellos ser vicios Web que cumplan los requerimientos solicitados sin conflictos o complicaciones durante su ejecución.

Arciniegas et al. (2011) en su trabajo Proceso de requerimiento y análisis para la definición de la arquitectura desde la perspectiva de usabilidad para el desarrollo de aplicaciones en la Web, presentan un proceso para requerimiento y análisis para la definición de la arquitectura considerando aspectos de usabilidad para el desarrollo de aplicaciones en la Web. Los procesos propuestos en este artículo corresponden a las etapas tempranas del proceso de ingeniería el cual abarca las etapas de requerimientos y análisis en las cuales el rol de la arquitectura tiene un papel relevante. Dicho proceso ha sido definido con base en la investigación realizada sobre las metodologías existentes, sin embargo, a diferencia de trabajos relacionados en este artículo se propone un proceso ágil que tienen como foco de interés la Usabilidad de las Aplicaciones Web realizadas por MIPYMEs.

Jaramillo y Chaverra (2012) en su artículo An environment based on pre-conceptual schemas for automatically generating source code under the MVC pattern, se refieren a las herramientas que facilitan las fases del desarrollo de software y afirman que hoy en día es casi imposible pensar en un desarrollo de software sin el apoyo de una de ellas. Una de las características más deseadas de estas herramientas es la generación automática de código. Actualmente, existen herramientas CASE y metaCASE que realizan esta función, pero con dos problemas principales: parten de lenguajes técnicos que no permiten la validación de los interesados y se genera artículo se propone una herramienta CASE para generar automáticamente código ejecutable bajo el patrón de programación MVC (model view controller), en el lenguaje de programación PHP, a

partir de los esquemas preconceptuales. De esta forma se busca hacer más completo el código resultante y permitir la validación de los interesados en el proceso.

Serna, Londoño y Zapata (2011) aseguran que las organizaciones empresariales deben responder a los cambios del entorno, optimizando sus procesos de negocio, usando las tecnologías de la información y las comunicaciones como factor clave para ser competitivas y flexibles a los cambios que afectan a la actividad de negocio. La arquitectura SOA establece un marco de diseño para lograr la integración de diferentes aplicaciones empresariales, para que desde sitios remotos sea posible acceder a los múltiples servicios que las empresas ofrecen. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular.

Pantoja, Collazos y Penichet (2013) abordan el tema de la mejora de procesos de desarrollo software por medio de entornos colaborativos y afirman que la Ingeniería de Software reconoce que una buena estrategia para aumentar la calidad de sus productos es por medio de la mejora de los procesos. Un factor que puede ayudar a las organizaciones a dirigir con éxito un proyecto de mejora es disponer de un soporte tecnológico por medio de herramientas groupware que ayuden a gestionar los proyectos de mejora de procesos. En este artículo se plantea un modelo de colaboración y una herramienta, orientados a fomentar la comunicación y coordinación entre los integrantes de un equipo de mejora, como apoyo a la implantación de proyectos SPI (Software Process Improvement) en pequeñas organizaciones de software.

Vega (2013) en su tesis de maestría describe la importancia del uso de Servicios Web y afirma que los Servicios Web Semánticos ofrecen beneficios que coadyuvan a la evolución de la Web. Beneficios como el descubrimiento, invocación y composición dinámica y automática de recursos, habilitan efectivamente la interoperabilidad entre sistemas, permitiendo una amplia gama de nuevos servicios y oportunidades de negocios en la Internet. La estructura necesaria para proveer estos beneficios por parte de los Servicios Web Semánticos hace que su desarrollo sea un proceso complejo. Para ello es necesario establecer formas más fáciles y dinámicas de desarrollar este tipo de software, que garanticen reutilización, calidad y rapidez en el proceso de desarrollo. El desarrollo dirigido por modelos realiza una contribución eficiente en estos

aspectos, dado que trabaja de manera intrínseca conceptos relacionados con estos aspectos: Separación de conceptos, reusabilidad e interoperabilidad entre componentes. En este trabajo se presenta un enfoque para desarrollo de software dirigido por modelos, orientado al desarrollo de los servicios web semánticos, generando las especificaciones más utilizadas para este tipo de aplicaciones.

Martínez (2014) investiga en su tesis de maestría, la forma de facilitar el proceso de desarrollo de software, partiendo de la abstracción del problema para construir modelos que permitan la representación de una solución general. Se emplea el desarrollo dirigido por modelos para la elaboración de un lenguaje de dominio específico y las plantillas para la generación de código, tomando como base una implementación de referencia. Se desarrolla una herramienta encargada de proporcionar un entorno de trabajo al desarrollador, y la generación de código fuente basado en un meta modelo, contribuyendo a la construcción de aplicaciones en la optimización de la productividad sobre el equipo que elabora software, asegurando aspectos principales como son la calidad, mantenibilidad y reutilización de elementos. La generación de código en forma automática bajo la arquitectura modelo vista controlador proporciona un mantenimiento factible de las aplicaciones construidas y la facilidad para la distribución de elementos dentro de un equipo de desarrollo de software.

3 Estrategia de seguridad propuesta

3.1 Estado actual institucional

Para determinar el estado actual de las políticas aplicadas al desarrollo de software web de la Institución se aplicó un instrumento (Anexo 1) el cual contiene 38 preguntas clasificadas por las siguientes categorías: estado actual, autenticación, permisos, base de datos, configuración y protección de aplicación que fueron dirigidas al jefe de Sistemas de la Institución, y que permite conocer si se están aplicando medidas que ayuden a prevenir vulnerabilidades de software en las aplicaciones desarrolladas en el Departamento de Sistemas. A continuación se muestra los resultados obtenidos por categoría.

1. **Estado actual**

   En esta categoría se busca conocer aspectos básicos de la Institución al momento de desarrollar aplicaciones Web en lo referente a lenguajes de programación, políticas establecidas, arquitectura entre otros.

   Análisis de resultados

   Las respuestas obtenidas en esta categoría por parte del jefe de sistemas de la institución, permiten precisar que el departamento de Sistemas no tiene establecida ninguna política de desarrollo seguro, pero si cuenta con una arquitectura de desarrollo definida la cual esta estandarizada y documentada. El lenguaje de programación en el que se desarrollan las aplicaciones web es PHP, éstas interactúan con servicios web implementados con estándares SOAP y WSDL. De igual forma se presta especial cuidado con las librerías utilizadas en los desarrollos, las cuales son descargadas de sitios oficiales.

2. **Autenticación**

   En esta categoría se busca establecer el estado de la autenticación de las aplicaciones de la institución, es decir si tiene implementado controles para los usuarios que hacen uso de la misma.

Análisis de resultados

De acuerdo a las respuestas obtenidas en esta categoría, se puede concluir que la autenticación de las aplicaciones web de la Institución permite el correcto funcionamiento de las sesiones, ya que garantiza su funcionamiento y evita problemas como la suplantación, ya que controla los tiempo de inicio de sesión y actividad de la misma, para que en caso de detectar inactividad, cerrarla. La autenticación entre la aplicación y los servicios web está establecida por medio de token entre la sesión que inicio en la aplicación y el servicio web. Se debe mejorar el cifrado de las contraseñas ya que están sobre criptografía débil y en la actualidad la aplicación no está requiriendo periódicamente el cambio de la misma.

3. **Base de datos**

En esta categoría se busca establecer los sistemas de bases de datos que son utilizados en los desarrollos web de la Institución y si se toman algunos controles sobre las tablas y datos almacenados.

Análisis de resultados

De acuerdo a las respuestas obtenidas se concluye que los sistemas de gestión de bases de datos utilizados en los desarrollos web de la institución son DB2 y MySQL. Para el caso de la base de datos DB2 se encuentra en una versión desactualizada, lo cual genera algunos inconvenientes en su rendimiento con los aplicativos. La información que se almacena en las tablas tiene establecida la respectiva validación de acuerdo al tipo de dato y longitud, de igual forma se establece que las tablas se les aplica integridad adecuada (primaria y foránea). Cada aplicativo web cuenta con su usuario propio el cual tiene los permisos adecuados para su perfecto funcionamiento y este usuario no se almacena en ninguna parte del código, lo cual garantiza que las credenciales no están expuestas.

4. **Permisos**

   En esta categoría se busca establecer el estado del acceso a los recursos con los que cuenta la aplicación.

   Análisis de resultados

   De acuerdo a las respuestas obtenidas en esta categoría se puede establecer que los mecanismos que garantizan la autorización a los recursos de la aplicación están correctamente implementados, ya que están establecidos sobre perfiles y roles que se asignan a cada uno de los usuarios que van a interactuar con el aplicativo. Se debe establecer un control sobre los formularios, ya que las peticiones que son enviadas deben tener una comprobación que si son del usuario que está utilizando el sistema.

5. **Configuración**

   En esta categoría se busca establecer el estado actual de la plataforma que soporta las aplicaciones web de la institución.

   Análisis de resultados

   Las respuestas obtenidas en esta categoría permiten concluir que el software que soportan las aplicaciones web está desactualizados, siendo el caso más preocupante la versión de la base de datos DB2 la cual se encuentra en la versión 6 y tiene problemas con el correcto rendimiento de la aplicación, de igual forma no se tiene control sobre los permisos establecidos en las carpetas y archivos lo que puede permitir que se acceda a ellas para copiar contenido malicioso. Los servidores web de la institución tienen solo habilitado lo necesario para funcionar a nivel de puertos y servicios y las personas que acceden a ellos tienen los roles definidos.

   Se debe mejorar la forma como se manejan los errores tanto a nivel de servidor, como de aplicación, ya que los errores revelan información confidencial de la aplicación y de la base de datos; estos mensajes de error deben ser genéricos para no revelar los fallos que han ocurrido. De igual forma, se deben establecer ventanas de mantenimiento

periódicas para evaluar las configuraciones y hacer revisión de log de errores, de seguridad y actualización del software que soporta la aplicación.

6. Protección de aplicación

En esta categoría se busca establecer las medidas de protección que tiene definidas las aplicaciones para evitar vulnerabilidades.

Análisis de resultados

A nivel de protección de la aplicación se puede concluir lo siguiente de acuerdo a las respuestas obtenidas:

- No se tiene establecidas librerías o mecanismos que ayuden a evitar ataques XSS en el aplicativo.
- Las interfaces que son de acceso público no están protegidas mediante encriptación de datos o token de seguridad, lo cual hace imposible que se tenga control sobre las mismas permitiendo que se puedan realizar ataques a la aplicación al acceder a ellas.
- La validación de los datos se hace de forma adecuada ya que se tienen establecidos métodos para comprobar su tipo y longitud antes de enviar la petición, de igual forma esta validación se hace del lado del cliente y del servidor.
- Se tiene control para evitar ataques de inyección gracias al mapeo de objetos relacionales con DCOTRINE que se tiene establecido en los aplicativos.
- No se tiene establecidos algoritmos de cifrado para proteger los datos que son considerados sensibles en la institución lo cual los podría dejar disponibles en caso que se presente alguna suplantación o robo de información ya que se almacena en cache de forma natural.
- Las aplicaciones no son capaces de detectar comportamientos atípicos o uso indebido por parte de los usuarios, en caso de detectar comportamientos que no son normales se debería bloquear la cuenta que lo esté generando o los rangos de IP desde donde se estén enviado dichas peticiones ya que esto se podría

considerar como un ataque lo cual pondría en peligro el correcto funcionamiento de la aplicación así como el acceso de otros usuarios.

De acuerdo a lo diligenciado en el instrumento (Anexo 4), se evidencian falencias y elementos por fortalecer a nivel de seguridad. Estas falencias se deben mejorar, ya que es un hecho que los nuevos aplicativos desarrollados en la institución serán sobre ambiente web y los que ya estén funcionando se irán migrando poco a poco hacia éstos. Al establecer una política de desarrollo seguro, se dotará a los desarrolladores de buenas prácticas de desarrollo que garanticen el buen funcionamiento de los aplicativos y el buen manejo de la información con la que van a interactuar los usuarios que accedan a ellos.

3.2 Estado actual frente a otras Instituciones de la región

Para conocer el estado actual de las políticas de seguridad aplicadas a los desarrollos web de la institución frente a otras Instituciones de Educación Superior de la región, se aplicó un instrumento (Anexo 1) a 5 instituciones que permitió realizar una comparación entre ellas. A continuación se muestran los resultados a través del porcentaje obtenido frente al estado actual de la institución.

3.2.1 Estado actual

1. ¿La Institución cuenta con políticas para el desarrollo seguro de aplicaciones web?

Figura 8 Políticas desarrollo seguro
Diseño propio

Análisis de resultados.

Las políticas establecidas al interior de la institución para el desarrollo de software son importantes pues garantizan la correcta construcción de los aplicativos institucionales. De acuerdo a los resultados obtenidos, la universidad no cuenta con estas políticas de desarrollo en comparación con las demás instituciones de educación superior de la región.

2. ¿La Institución tiene establecida una arquitectura de desarrollo de software web documentada y estandarizada?

Figura 9 Arquitectura de desarrollo
Diseño propio

Análisis de resultados.

La arquitectura de desarrollo garantiza que todos los sistemas en producción están construidos bajo los mismos parámetros. En este sentido, la institución cuenta con una arquitectura de desarrollo estandarizada, pero debe fortalecer las políticas de seguridad aplicadas pues la seguridad es implementada por cada desarrollador de forma independiente. De acuerdo con los resultados el 80% de las instituciones de la región cuentan con una arquitectura de desarrollo, contra un 20% que no tienen ninguna establecida.

3. ¿Cuál o cuáles lenguajes de programación utiliza la institución para desarrollar aplicaciones web?

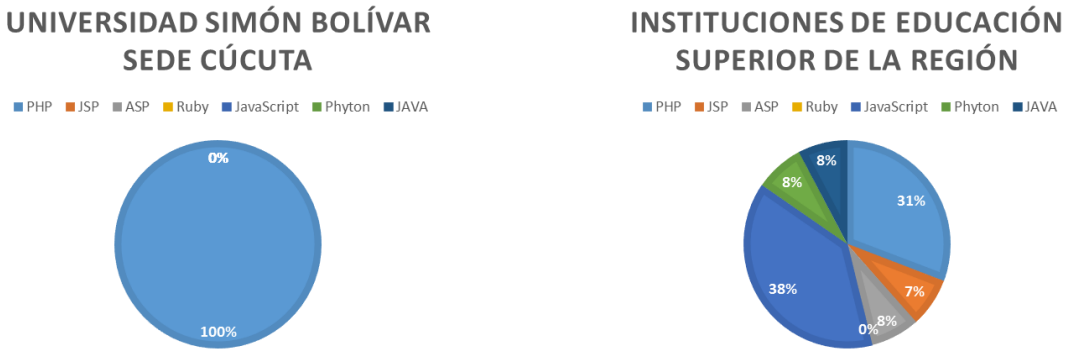

Figura 10 Lenguajes de programación
Diseño propio

Análisis de resultados.

De acuerdo con los resultados obtenidos, la institución implementa sus sistemas web usando el lenguaje de programación PHP. Las instituciones de la región lo hacen en una variedad de lenguajes tales como PHP, JavaScript, JAVA, JSP entre otros.

4. ¿Las librerías JavaScript utilizadas en los desarrollos web de la Institución, provienen de fuentes confiables o de sitios oficiales?

Figura 11 Librerías de fuentes oficiales
Diseño propio

Análisis de resultados.

Es importante la fuente de donde provienen las librerías usadas en los desarrollos, ya que al no ser de sitios oficiales podría tener modificaciones con código malicioso. De acuerdo a los resultados obtenidos, la institución usa librerías de fuentes oficiales, con respecto a las instituciones de la región, las cuales en un 83% usan librerías de fuentes confiables contra un 17% que no tiene en cuenta este aspecto.

5. ¿Las aplicaciones web de su Institución, intercambian datos entre ellas por medio de Servicios Web (WS)?

Figura 12 Uso de servicios web
Diseño propio

Análisis de resultados.

El uso de servicios web posibilita el intercambio de información entre las aplicaciones, la institución tiene implementado servicios web para diferentes funcionalidades de los sistemas. De acuerdo a los resultados obtenidos, un 60% de las instituciones usan servicios web contra un 20% que no los usan o no aplican.

6. ¿Los servicios Web se basan en estándares y protocolos abiertos?

Figura 13 Estándar servicio web
Diseño propio

Análisis de resultados.

Los servicios utilizados en la institución están basados en estándares abiertos. De acuerdo a los resultados obtenidos un 80% de las instituciones de la región usan estándares abiertos para implementar los servicios web.

7. En caso de ser positiva la respuesta anterior, indique ¿Cuáles son utilizados por las aplicaciones web de su Institución?

Figura 14 Estándares usados de servicios web
Diseño propio

Análisis de resultados.

Los estándares utilizados en la institución para implementar los servicios web son WSDL y SOAP. Las instituciones de la región construyen sus servicios en estándares como REST, XML y WSDL.

3.2.2 Autenticación

8. ¿La autenticación de las aplicaciones web, garantiza un correcto funcionamiento en las sesiones y evita la suplantación?

Figura 15 Autenticación de aplicaciones
Diseño propio

Análisis de resultados.

La autenticación es uno de los factores más importantes al momento de desarrollar aplicativos web ya que debe garantizar su buen funcionamiento para evitar problemas como la suplantación. De acuerdo a los resultados obtenidos los aplicativos de la institución y la región garantizan el buen desarrollo en este aspecto.

9. ¿Se controla el inicio, duración y cierre de la sesión que utiliza el usuario?

Figura 16 Duración de sesiones web
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos, la institución controla la duración de las sesiones así como su actividad. El 80% de las instituciones de la región controla este aspecto y un 20% no lo hace. Es importante tener control sobre la duración de las sesiones ya que en caso de inactividad se debe cerrar, evitando así que los aplicativos queden abiertos y que terceras personas hagan uso de ellos.

10. ¿Las aplicaciones web solicitan de forma periódica el cambio de la contraseña, comprobando la longitud y seguridad de los datos ingresados por el usuario?

Figura 17 Cambio de contraseña
Diseño propio

Análisis de resultados.

Es importante solicitar de forma periódica el cambio de contraseñas en los aplicativos web comprobando la seguridad de las mismas, en la actualidad las aplicaciones de la institución no solicitan dicho cambio de forma periódica pero si se comprueba la seguridad de la contraseña. Un 60% de las instituciones de la región si tiene establecido el cambio periódico de contraseñas contra un 40% que no lo hacen.

11. ¿Las contraseñas se almacenan con un algoritmo diseñado específicamente para la protección por contraseña, como por ejemplo bcrypt, PBKDF2 o scrypt?

Figura 18 Algoritmo de protección de contraseña
Diseño propio

Análisis de resultados.

Los algoritmos utilizados para proteger las contraseñas de la institución son seguros pero no son tan fuertes como deberían ser ya que se usan algoritmos MD5 y SHA1. El 80% de las instituciones de la región manifiestan usar algoritmos fuertes frente a un 20% que no lo hacen.

12. ¿Existe una autenticación mutua entre el cliente que accede a los servicios y el servidor que proporciona el servicio?

Figura 19 Autenticación servicio web – aplicación
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos, la institución si cuenta con autenticación entre los servicios web y los aplicativos. Un 60% de las instituciones de la región no tienen en cuenta este aspecto frente a un 40% que si lo tiene en cuenta.

3.2.3 Base de datos

13. ¿Indique cuáles de los siguientes sistemas de gestión de bases de datos se utilizan para el desarrollo de aplicaciones web?

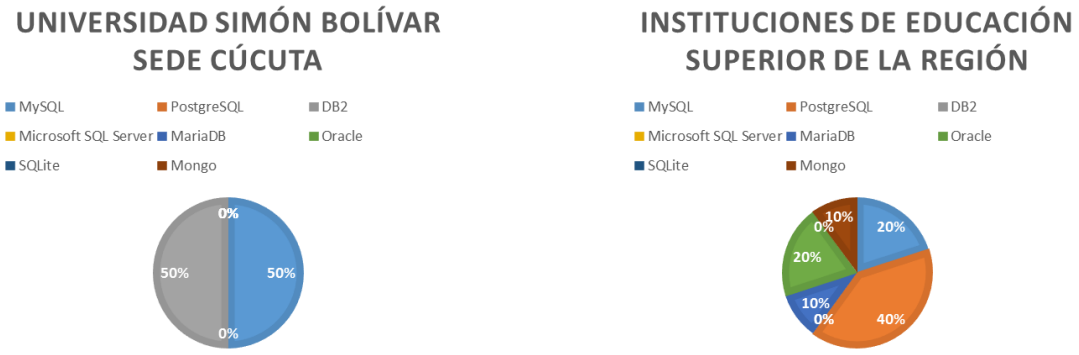

Figura 20 Sistemas de gestión de base de datos
Diseño propio

Análisis de resultados.

Los motores de bases de datos utilizados en las aplicaciones web de la institución son MySQL y DB2. Las instituciones de la región utilizan diversos motores de bases de datos siendo el más usado PostgreSQL con 40%, seguido de Oracle y MySQL cada uno con 20%.

14. ¿La información almacenada en la base de datos, está validada mediante el uso de constraints (restricciones aplicadas a los objetos de una base de datos: default, not null, unique, check) restringiendo de esta forma la entrada para cada campo?

Figura 21 Validación datos base de datos
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos las instituciones de la región aplican las restricciones necesarias a nivel de base de datos sobre cada campo utilizado para almacenar la información de la institución.

15. ¿En el sistema de gestión de base de datos se aplican reglas de integridad (llaves primarias y foráneas)?

Figura 22 Reglas de integridad
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos, las instituciones aplican reglas de integridad sobre las tablas de la base de datos lo que garantiza su buen desempeño a nivel de ejecución y la veracidad de la información.

16. ¿Se cuenta con usuarios distintos en la base de datos para cada desarrollo web, así como sus propios privilegios?

Figura 23 Usuarios base de datos
Diseño propio

Análisis de resultados.

Este punto es importante, ya que cada aplicativo de la institución debe tener definido su usuario, de acuerdo a los resultados las instituciones de la región cumplen con este parámetro.

17. ¿Las credenciales de acceso a la base de datos se almacenan en alguna parte del desarrollo?

Figura 24 Credenciales de acceso
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos, la institución no deja a la vista las credenciales de acceso ya que la cadena de conexión está oculta y sólo se observa un alias de la base de datos, el cual solo está disponible en el servidor donde está alojado el aplicativo. Un 60% de las aplicaciones de la región dejan a la vista las credenciales de acceso.

3.2.4 Permisos

18. ¿Los permisos de acceso de los usuarios a la base de datos, son los adecuados de acuerdo a los privilegios que debería tener?

Figura 25 Privilegios de usuarios
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos, las instituciones de la región otorgan solo los permisos necesarios a los usuarios de la base de datos que se conectan a las diferentes aplicaciones.

19. ¿Las aplicaciones web, garantizan que el usuario esté autorizado mediante un rol, perfil o algún tipo de comprobación de control de acceso, para interactuar con las funcionalidades de la aplicación o acceder a los datos de la misma?

Figura 26 Autorización de funcionalidades
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos las instituciones de la región tienen establecido roles y perfiles para acceder a las funcionalidades de los aplicativos web, lo que garantiza que los usuarios solo accedan a lo que tienen permiso.

20. ¿Los formularios de las aplicaciones web, tienen establecidos un identificador aleatorio único para cada usuario que inicie sesión? (donde el sistema revisa si el formulario es válido, comparando el identificador generado en la sesión antes de enviar una petición).

Figura 27 Identificador de usuario
Diseño propio

Análisis de resultados.

Este aspecto es esencial, ya que las peticiones enviadas deben ser solo la del usuario que inició sesión; en este aspecto los aplicativos de la institución no hacen esa comprobación antes de enviar información a la base de datos. Los resultados, indican que las instituciones de educación superior de la región si hacen esta comprobación antes de realizar la inserción de datos en la base de datos.

3.2.5 Configuración

21. ¿Alguno de los programas que soportan su aplicación web, está desactualizado? (Sistema Operativo, servidor Web/App, motor de Base de Datos, API y todos los componentes de bibliotecas)

Figura 28 Estado software soporte web
Diseño propio

Análisis de resultados.

Es importante tener la plataforma que soporta los aplicativos web actualizados, ya que las últimas actualizaciones vienen con correcciones de seguridad importantes. En la institución no se tiene actualizado el software que soporta los aplicativos, siendo el caso más preocupante la versión de la base de datos. Un 40% de las instituciones de la región no tienen actualizada su plataforma web frente a un 60% que manifiesta que no tienen software desactualizado en los servidores.

22. ¿En los servidores se habilitan o instalan funciones innecesarias, como por ejemplo, puertos, servicios, páginas, cuentas, privilegios?

Figura 29 Funciones en servidores web
Diseño propio

Análisis de resultados.

De acuerdo a los resultados, la institución no instala servicios innecesarios en los servidores web. Un 80% de instituciones de la región tienen definido este control en sus servidores contra un 20% que no lo tienen en cuenta.

23. ¿Se tiene definido los roles de las personas que ingresan con permisos de administrador a los servidores de aplicaciones y bases de datos?

Figura 30 Roles usuarios servidores
Diseño propio

Análisis de resultados.

Los resultados indican que las instituciones de educación superior de la región tienen definido los usuarios que pueden ingresar a los diferentes servidores de la institución y los privilegios con los que cuentan.

24. ¿Se tiene control sobre los permisos establecidos para las carpetas y ficheros de las aplicaciones web?

Figura 31 Permisos carpetas y ficheros
Diseño propio

Análisis de resultados.

Resulta esencial establecer permisos sobre las carpetas y archivos, ya que esto garantiza que piratas informáticos no puedan acceder a las mismas. La institución en la actualidad no tiene establecidos permisos sobre la carpeta, lo cual constituye una vulnerabilidad en las aplicaciones web. De acuerdo a los resultados obtenidos, un 100% de las instituciones de la región si tienen implementados permisos sobre carpetas y archivos.

25. ¿Su manejo de errores revela rastros de pila u otros mensajes de error demasiado informativos a los usuarios?

Figura 32 Errores de aplicación
Diseño propio

Análisis de resultados.

Cuando una aplicación falla se debe mostrar un mensaje genérico que no muestre mayor información a los usuarios. En este aspecto la institución no tiene establecido un control de excepciones, ya que cuando la aplicación falla revela toda la información que ocasiono el fallo. Los resultados arrojaron que un 80% de las instituciones tienen precaución sobre los errores que se muestran a los usuarios frente a un 20% que no lo hace.

26. ¿Cuenta con una arquitectura de aplicación robusta, que proporciona una separación efectiva y segura entre los componentes?

Figura 33 Arquitectura de aplicación
Diseño propio

Análisis de resultados.

Según las respuestas obtenidas, las instituciones de la región cuentan con una arquitectura robusta la cual garantiza el correcto funcionamiento de las aplicaciones web.

27. ¿Se establecen ventanas de mantenimiento periódicas sobre la arquitectura que soporta las aplicaciones web? (para verificar que las configuraciones están correctas, se realice la revisión de logs de seguridad y actualización de software, entre otras).

Figura 34 Ventanas de mantenimiento
Diseño propio

Análisis de resultados.

Este es un aspecto de relevancia, ya que se debe establecer de forma periódica una revisión sobre errores que pueda generar la aplicación, así como para actualizar el software que la soporta. La institución en la actualidad no tiene establecidas ventanas de mantenimiento, por el contrario las demás instituciones de educación superior si tienen establecidos dichos controles.

28. ¿Se cuenta con un control de versiones de los aplicativos web, el cual permita identificar los cambios que ha sufrido la aplicación durante el ciclo de vida o que permita volver a una versión anterior en caso de alguna falla?

Figura 35 Control de versiones
Diseño propio

Análisis de resultados.

A través del control de versión se puede llevar la trazabilidad de los cambios realizados a los aplicativos a lo largo del tiempo. En la actualidad la institución no tiene establecido dicho control de versiones en comparación con las demás instituciones de la región, que manifiestan que si lo tienen establecido para sus aplicativos.

3.2.6 Protección de la aplicación

29. ¿Las aplicaciones web tienen implementado algún framework o librería anti-XSS que ayude a disminuir los inconvenientes con este tipo de ataques?

Figura 36 Librería anti-XSS
Diseño propio

Análisis de resultados.

Resulta relevante tener controles en las aplicaciones que ayuden a prevenir los ataques XSS en los aplicativos de las instituciones. De acuerdo a los resultados, la institución no tiene ningún control establecido para prevenir la inyección de código JavaScript o HTML peligroso en los diferentes aplicativos, en este sentido, un 80% de las instituciones de la región manifiestan que si tienen implementado algún control para evitar este tipo de ataques.

30. ¿Las interfaces de acceso público están protegidas mediante encriptación de datos, token de seguridad o algún tipo de comprobación de control sobre las mismas?

Figura 37 Interfaces de acceso publico
Diseño propio

Análisis de resultados.

Las interfaces públicas deben tener un mayor control ya que van a estar al alcance de cualquier persona, basados en los resultados, la institución no tiene un control sobre estas interfaces, provocando vulnerabilidades sobre el aplicativo. Un 80% de instituciones de la región manifiestan que si aplican controles contra un 20% que no lo hace.

31. ¿La validación de los datos de entrada (longitud, tipo de dato, rango de valores aceptados, sintaxis) de las aplicaciones, se realizan del lado del servidor así como del cliente?

Figura 38 Validación de datos de entrada
Diseño propio

Análisis de resultados.

Los datos ingresados por los usuarios podrían contener caracteres especiales o palabras reservadas del sistema, es por esto que todas las cadenas ingresas deben ser válidas de acuerdo al dato que se espera y filtradas para eliminar caracteres especiales o palabras reservadas. Estas comprobaciones se deben hacer al lado del cliente y del servidor, ya que los controles que se agreguen del lado del cliente se podrían modificar accediendo al código que permite visualizar el navegador. De acuerdo con los resultados obtenidos, la institución establece estos controles sobre los datos de entrada y un 80% de instituciones de la región tienen establecidos estos controles sobre los datos frente a un 20% que manifiestan que no.

32. ¿Los aplicativos web desarrollados en su Institución, poseen controles para evitar ataques de inyección?

Figura 39 Control ataques de inyección
Diseño propio

Análisis de resultados.

Las aplicaciones de la institución tienen implementado un ORM para la interacción con la base de datos, el cual garantiza que no se presenten ataques de inyección. De acuerdo con los resultados obtenidos un 80% de las instituciones aplican controles sobre posibles ataques de inyección frente a un 20% que no los tiene.

33. ¿Se cuenta con algoritmos criptográficos para proteger los datos sensibles de la Institución?

Figura 40 Algoritmos criptográficos
Diseño propio

Análisis de resultados.

En la actualidad los datos sensibles de la institución no están protegidos con algoritmos criptográficos manipulando dichas cadenas en texto blanco. Según los resultados obtenidos un 80% de las instituciones de la región manifiesta que si tienen algoritmos criptográficos para la manipulación de dichos datos.

34. ¿Los datos considerados sensibles (contraseñas, números de tarjetas de crédito, información personal, pagos registrados en la Institución) tienen algún nivel de protección adicional?

Figura 41 Protección datos sensibles
Diseño propio

Análisis de resultados.

De acuerdo a los resultados obtenidos la institución no aplica niveles de protección adicional sobre datos sensibles en la institución. Un 80% de las instituciones de la región manifiesta que si tienen niveles de protección adicional frente a un 20% que manifiestan no tener controles adicionales.

35. ¿Los formularios que solicitan el ingreso de datos sensibles, tienen deshabilitado la opción de autocompletado y almacenamiento en cache?

Figura 42 Formularios datos sensibles
Diseño propio

Análisis de resultados.

Los resultados indican que la institución no tiene deshabilitado las opciones de autocompletar en los formularios así como el almacenamiento en cache de los datos sensibles de la institución. Un 80% de las instituciones de la región manifestaron que si tienen dichos controles sobre los formularios y cache para el manejo de datos sensibles.

36. ¿Las aplicaciones tienen protección de ataques, que permite identificarlos, bloquearlos y proporcionar algún detalle, como por ejemplo, si se está utilizando la aplicación de una forma que un usuario común nunca haría? (tiempo demasiado alto, entradas atípicas, patrones de uso inusuales, peticiones repetidas)

Figura 43 Protección de ataques
Diseño propio

Análisis de resultados.

Es esencial identificar cuando la aplicación se está usando de forma atípica ya que esto podría representar un riesgo de ataque. De acuerdo a los resultados las aplicaciones de la institución no tienen protección de ataques. Un 80% de las demás instituciones de educación superior de la región si aplican controles contra ataques en sus aplicativos y un 20% no tienen implementados controles en sus aplicativos.

37. ¿En caso de un comportamiento atípico, las aplicaciones de su Institución, bloquean solicitudes, direcciones IP o rangos de direcciones IP?

Figura 44 Bloqueo de solicitudes
Diseño propio

Análisis de resultados.

Los resultados indican que las aplicaciones de la institución no bloquean rangos de IP en caso de detectar comportamientos atípicos. Por su parte un 60% de los jefes de sistemas de las instituciones de educación superior indicaron que si bloquean rangos de IP en caso de identificar comportamiento atípico y un 40% expresaron que no implementan bloqueo de rangos de direcciones IP.

38. ¿Las aplicaciones de su Institución, bloquean las cuentas de los usuarios que le dan mal uso a las mismas?

Figura 45 Cuentas de usuario
Diseño propio

Análisis de resultados.

Es importante establecer controles sobre las cuentas de usuario que les dan mal uso a las mismas, ya que un mal comportamiento en la aplicación podría representar vulnerabilidades de seguridad. Frente a esta pregunta, los resultados indican que la institución no bloquea cuentas de usuario que le dan mal uso a las aplicaciones. Por su parte el 80% de las instituciones de la región expresaron que si bloquean usuarios que se comportan mal en la aplicación y un 20% expresaron que no.

3.3 Descripción de la estrategia

3.3.1 Introducción

La estrategia propuesta para el Departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta está definida como una Política de Desarrollo de Software Web Seguro, la cual tiene por objeto establecer los parámetros de seguridad que deben aplicar los desarrolladores de software web de la institución para blindar los aplicativos desarrollados en la institución y así garantizar la integridad de la información que es manipulada por medio de los mismos. En particular la presente política define el objetivo y alcance en lo referente a desarrollo, mantención y adquisición de sistemas de información de la institución.

3.3.2 Objetivo general

Definir las medidas de seguridad con las que deben contar los aplicativos web de la institución en el proceso de construcción, mantención, adquisición e implementación.

3.3.3 Objetivo especifico

Definir los controles para proteger las aplicaciones web, garantizando la seguridad de la información y de los procesos de construcción, mantención o adquisición de las mismas.

Establecer las normas y documentos que deben respaldar los procesos de desarrollo, pruebas, mantención e implementación de los diferentes sistemas web de la institución.

3.3.4 Alcance

El alcance de la política de seguridad aborda todos los desarrollos web o adquisiciones de software web de la Universidad Simón Bolívar Sede Cúcuta.

3.3.5 Destinatario

Todos los funcionarios del Departamento de Sistemas responsables de la definición, planificación, desarrollo, mantención y adquisición de software web de la Universidad Simón Bolívar sede Cúcuta.

3.3.6 Marco Normativo

La normativa relacionada con la política de desarrollo seguro, que se define en este documento, se describe a continuación.

Normativa internacional

- ISO/IEC 27001: proporciona un modelo para establecer, implementar, operar, monitorear, revisar, mantener y mejorar un Sistema de Gestión de Seguridad de la Información (SGSI).
- ISO/IEC 27002: proporciona recomendaciones de las mejores prácticas en la gestión de la seguridad de la información a todos los interesados y responsables en iniciar, implantar o mantener un Sistema de Gestión de la Seguridad de la Información (SGSI).
- Open Web Application Security Project (OWASP): proporciona documentación que permite establecer mejores prácticas al momento de desarrollar software web seguro.
- COBIT 5 for Information Security: Proporciona orientación para ayudar a los profesionales de TI y seguridad a entender, utilizar, implementar y dirigir actividades importantes relacionadas con la seguridad de la información.
- CWE/SANS Top 25: proporciona una clasificación de vulnerabilidades en las aplicaciones las cuales pueden ocurrir en la arquitectura, diseño, código o implementación.
- WASC - The Web Application Security Consortium: proporciona estándares de seguridad para la World Wide Web.
- OASIS WS-Security: proporciona un protocolo de comunicaciones para aplicar seguridad a los servicios web para garantizar la integridad y confidencialidad en los mensajes.
- WS-Policy: proporciona una especificación que permite a los programadores de servicios web anunciar las políticas de seguridad, calidad, etc. y a los clientes de servicios web especificar los requisitos de calidad, seguridad, latencia, etc.

Normativa nacional

- Ley 1273 de 2009: “Por medio de la cual se modifica el Código Penal, se crea un nuevo bien jurídico tutelado - denominado de la protección de la información y de los datos y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones”.
- Conpes 3854: Es la política nacional de seguridad digital la cual proporciona recomendaciones y mejores prácticas internacionales en gestión de riesgos de seguridad digital emitidas por la Organización para la Cooperación y el Desarrollo Económico (OCDE).

3.3.7 Lineamientos

Principios

Los principios que rigen la definición de la política de desarrollo web seguro, son transversales a la aplicación de la misma.

1. Confidencialidad: se debe permitir el acceso únicamente a los datos que el usuario esté autorizado consultar.
2. Integridad: se debe asegurar que los datos no se alteran por usuarios no autorizados.
3. Disponibilidad: se debe asegurar que los sistemas y datos están disponibles para los usuarios autorizados cuando lo necesiten.
4. Responsabilidad: todos los desarrolladores de software web de la institución son responsables de garantizar la seguridad de las aplicaciones, por tanto, deben ser conscientes de la importancia de desarrollar software seguro y conocer las buenas prácticas para garantizar su seguridad.
5. Solucionar problemas de seguridad correctamente: los desarrolladores de software web de la institución deben documentar los fallos encontrados en las aplicaciones, para así desarrollar test y comprender la raíz del problema.
6. Simplicidad: los desarrolladores de software web de la institución deben evitar el uso de algoritmos complejos para que otros desarrolladores puedan retomar y entender de una forma más rápida y simple.

Identidad

Las aplicaciones web desarrolladas deberán seguir los lineamientos planteados por el departamento de Sistemas, en cuanto al diseño, arquitectura de desarrollo y por la política en lo referente a seguridad con el objeto de mantener la organización y estandarización en los sistemas desarrollados.

3.3.8 Política de desarrollo web seguro

A continuación, se describe el conjunto de prácticas y lineamientos propuestos para garantizar la seguridad de los aplicativos web desarrollados en la institución. En la Figura 46 se puede observar la estructura de la política de desarrollo web seguro.

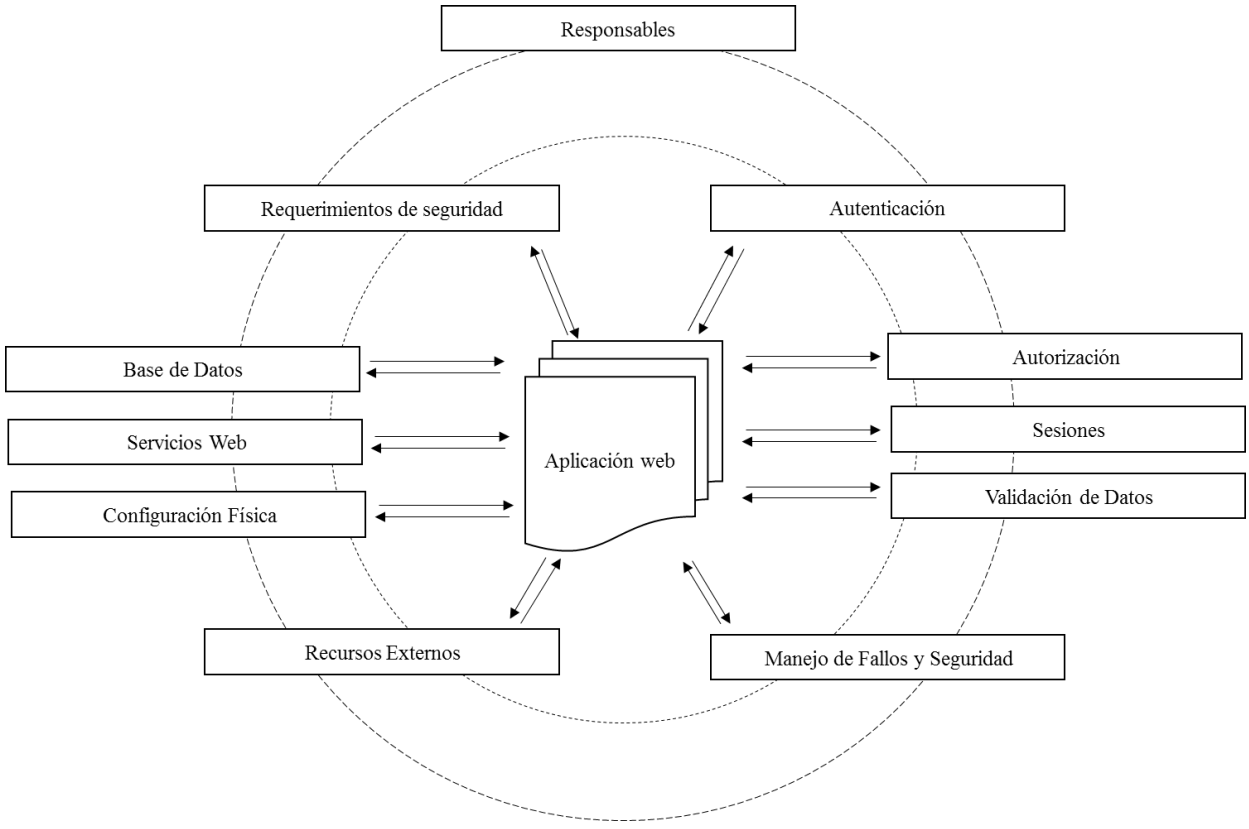

Figura 46 Estructura política de desarrollo web seguro
Diseño propio

3.3.8.1 Responsabilidades

En el proceso de desarrollo de software son las personas (jefe de departamento, coordinador de desarrollo, programador entre otros) encargadas de garantizar la seguridad de las aplicaciones ya que son los encargados del diseño y planeación de los aplicativos web. De acuerdo con Smith y Dehlinger (2014) los responsables de desarrollar las aplicaciones web sin saberlo implementan sistemas web con fallos de seguridad comprometiendo así la integridad de la aplicación.

Jefe de Sistemas

1. Tendrá como responsabilidad definir, aplicar, evaluar y modificar la política de desarrollo web seguro, así como establecer la frecuencia de su actualización.
2. El Jefe de Sistemas en conjunto con el Coordinador de Sistemas, son los responsables de la seguridad de la información en la Universidad Simón Bolívar sede Cúcuta.

Coordinador de Sistemas

1. El Coordinador de Sistemas en conjunto con los desarrolladores serán los responsables de definir el nivel de criticidad del aplicativo web o modulo a desarrollar, y de identificar los controles de seguridad que se deben implementar para protegerlos.
2. El Coordinador de Sistemas deberá revisar, aprobar o rechazar los procesos y controles orientados a mitigar o eliminar los riesgos relacionados en la fase de desarrollo, así como en la adquisición de software web.
3. Verificar el cumplimiento de los controles de seguridad establecidos para el desarrollo de aplicaciones web de la institución, así como en la adquisición de sistemas de información.

Desarrolladores

1. Los desarrolladores de software web de la institución deberán aplicar la política de desarrollo web seguro y el Coordinador de Sistemas hará seguimiento de su cumplimento.
2. Los desarrolladores en conjunto con el Coordinador de Sistemas, son responsables de definir los controles que permitan asegurar que los procesos de desarrollo están

garantizando la mitigación de vulnerabilidades de seguridad, dichos controles pueden ser:

a. Metodología de análisis
b. Implementación de pruebas unitarias y de integración
c. Administración de sistemas y de base de datos
d. Documentación

3.3.8.2 Proceso de desarrollo de software seguro

Implementar controles de seguridad al proceso de desarrollo de software es importante ya que de acuerdo con Daud (2010) nos proporciona los mecanismos de defensa contra las diferentes amenazas que se puedan presentar. Al implementar un proceso de desarrollo seguro se garantiza que una fase de desarrollo no transfiera sus vulnerabilidades a otras fases.

1. Se implementará una metodología de desarrollo seguro, la cual debe garantizar un óptimo desempeño, ya que los procesos de la institución son robustos. La metodología de desarrollo debe poseer una fuerte aceptación en su diseño para que de esta forma ayude a tener una adecuada adaptabilidad por parte de los desarrolladores y así lograr una reducción en los errores de programación y mejorar su productividad, teniendo documentados sus componentes y controles de seguridad.
2. Se debe definir un estándar basado en buenas prácticas, el cual debe considerar la separación de las capas web y servidores de bases de datos, documentación de código fuente, aplicación de pruebas a los módulos desarrollados, manejo de excepciones, método de definición de las variables, funciones o clases, filtrado de cadenas de entrada y código escrito de forma que pueda ser mantenido y legible por cualquier desarrollador del departamento.
3. Todo el código de las aplicaciones web de la institución debe estar versionado para identificar las nuevas funcionalidades agregadas y para volver a un estado inmediatamente anterior en caso de fallos.
4. El ciclo de vida del desarrollo de software web debe incluir la implementación de pruebas funcionales y no funcionales, las cuales deben incluir pruebas de seguridad.

5. El proceso de desarrollo, así como el de pruebas, deben llevarse a cabo en ambientes dispuestos para ello.
6. Los aplicativos que intercambien datos con otros sistemas o bases de datos deben contar con controles de seguridad en ambos extremos.
7. El proceso de construcción de software y en particular la codificación, debe tener en lo posible dos o más responsables para garantizar que el proceso no se detenga en caso de ausencias, es decir los procesos de desarrollo no deben depender de una sola persona.
8. Los aplicativos web tercerizados deben cumplir con esta política de desarrollo seguro y con las indicaciones dadas por el jefe de Sistemas de la institución.

3.3.8.3 Documentación

1. La documentación de los aplicativos web se debe realizar en las plantillas institucionales de manual de usuario y manual del sistema. Excepciones a esta política será solamente en los casos de adquisición de software.

2. La documentación referente al desarrollo del software web debe tener definido procedimientos de control de versionamiento.

3. Se debe permitir el acceso a la documentación de los aplicativos, biblioteca de código fuente y ficheros ejecutables solo a funcionarios autorizados. La excepción a esta política, son los manuales de usuario u otros documentos que son dirigidos a usuarios finales.

3.3.8.4 Definición requerimientos de seguridad

Para el desarrollo de una aplicación web es importante definir los requisitos de seguridad con los cuales debe cumplir la aplicación, estos se deben definir al inicio de cada fase de desarrollo. De acuerdo con Hope (2007) en estos requisitos se debe identificar las partes de la aplicación que requieren especial seguridad. Para definir los requisitos de seguridad es importante tener en cuenta los diferentes patrones disponibles para identificar el apropiado para nuestro aplicativo así como recomiendan Okubo y Tanaka (2008) ya que para realizar dicha definición se debe requerir de mucha experiencia.

1. Para el desarrollo de los aplicativos web o de los módulos del mismo se deben definir los controles de seguridad desde la etapa de levantamiento de requerimientos, como por ejemplo encriptación, mensajes, configuración, auditoria entre otros.

2. En el proceso de identificación de los controles de seguridad deben participar los usuarios finales del aplicativo.

3. La implementación de controles de seguridad debe ser de forma automática para así evitar intervenciones manuales. La excepción a esta política la debe aprobar el jefe de Sistemas de la institución.

4. Durante la etapa de diseño, se deben tener en cuenta los procedimientos necesarios para realizar revisiones periódicas de contenido de campos, registros, tablas o archivos considerados sensibles y procesos de depuración, como por ejemplo limpieza de datos, indexaciones u otros procesos relacionados con el rendimiento de la aplicación.

5. Se puede realizar el desarrollo de los aplicativos utilizando datos de pruebas extraídos desde la base de datos de ambiente de pruebas, evitando así el uso de datos que estén en el ambiente de producción de la institución, pero solo deben ser accedidos desde las oficinas del departamento de Sistemas. La excepción a esta política, debe ser autorizada por el jefe de Sistemas de la institución, para lo cual deberá redactar un formato de confidencialidad.

3.3.8.5 Autenticación

La autenticación es el procedimiento que permite validar que un usuario del aplicativo web es el que está autorizado para acceder al sistema. De acuerdo con Mundana et al. (2016), las web modernas utilizan múltiples métodos para permitir la autenticación de nuestros visitantes y conceder así los diferentes niveles de acceso. Es por ello que la autenticación se debe realizar de forma segura ya que por media de esta es donde concedemos al usuario la autorización para interactuar con nuestro aplicativo.

1. Nombre de Usuario: La asignación de nombres de usuario no debe hacerse de forma arbitraria, para ello se debe tener en cuenta las siguientes características:

   a. Se debe consultar el directorio activo de la institución. El nombre de usuario debe ser único, con un máximo de 8 caracteres y debe estar compuesto por la abreviación de los nombres y apellidos.
   b. Los usuarios de funcionarios no deben ser creados de acuerdo a la función que van a desempeñar en el sistema.
   c. Para el caso de los usuarios docentes, se debe generar una vez se registra la hoja de vida y de manera aleatoria.
   d. El nombre de usuario de los estudiantes debe ser generado por la aplicación de forma automática una vez se registre la primera matricula del estudiante, y debe estar compuesto por iniciales de nombres, apellidos, caracteres especiales y números.
2. Asignación de Contraseñas: se asignará contraseña a las cuentas de los usuarios sólo en el momento de ser creadas. Una vez el usuario inicie sesión, deberá cambiarla, es por esto que se debe registrar la información que le permita al usuario poder restablecerla si tiene algún inconveniente.
3. Seguridad de Contraseñas: las contraseñas digitadas por los usuarios se deben proteger, es por esto que se deben cumplir los siguientes requisitos de seguridad:
   a. No se deben usar algoritmos viejos como MD5 y SHA1 (de 160 bits), los cuales son débiles.
   b. Se debe usar algoritmos como AES-128 o SHA1 de 256 bits los cuales son más fuertes.
   c. Se debe usar un algoritmo de cifrado para toda contraseña almacenada, por ningún motivo se debe almacenar contraseñas en texto claro ya que es inseguro.
   d. De ninguna manera se debe mostrar el hash generado de la contraseña al usuario.
4. Cambio de contraseñas: las contraseñas se deben cambiar de forma periódica por lo menos 2 veces al semestre, por lo tanto, la aplicación debe garantizar la función para realizar dicho cambio de manera segura, teniendo en cuenta que:
   a. Se debe incluir la contraseña anterior, la contraseña nueva y confirmación de la nueva contraseña.

b. Se debe establecer en el formulario la función AUTOCOMPLETE=off para evitar que el navegador guarde la contraseña localmente.
c. Si se ingresa de forma incorrecta la contraseña varias veces, se debe bloquear la cuenta y destruir la sesión.
d. Se debe validar la contraseña frente a un historial de 24 hashes de contraseñas para prevenir contraseñas repetidas.
e. No se debe permitir contraseñas en blanco.
f. La contraseña debe tener una longitud mínima de 8 caracteres (pero nunca una máxima), combinando caracteres especiales, mayúsculas y minúsculas.
g. Se debe usar diccionario de contraseñas para indicarle al usuario si la contraseña es débil.

5. Restablecimiento de Contraseñas: se debe garantizar que el usuario pueda restablecer las contraseñas en caso de olvidarla, cumpliendo con los siguientes parámetros de seguridad:
    a. Se debe enviar un mensaje al usuario explicando que se está intentado restablecer la contraseña, si no se ha solicitado se debe dar la opción de reportar el incidente.
    b. Si el usuario solicitó el restablecimiento, se debe enviar un token criptográficamente único y limitado por tiempo, el cual debe ser copiado y pegado en la aplicación.
    c. No se debe enviar el mensaje con hipervínculos ya que va en contra de las buenas prácticas contra el phishing. El valor debe ser introducido en la aplicación, la cual debe estar esperando por el token.
    d. El token enviado al usuario debe ser verificado, se debe tener en cuenta que no ha expirado y que es válido para esa cuenta de usuario.
    e. Se debe enviar un correo electrónico al usuario notificándole del cambio de contraseña.
6. Inactividad: se debe comprobar frecuentemente la actividad de la cuenta, en caso de permanecer inactiva por mucho tiempo, se debe cerrar la sesión.

7. Iniciar sesión en diferente ubicación: si el usuario inicia sesión desde ubicaciones diferentes, la aplicación deberá cerrar la sesión más antigua y deberá informar al usuario que se inició sesión cerrando una sesión previa.
8. Expiración de Cuentas: se debe establecer un estado en las cuentas de los usuarios que permita verificar si están activas. Si el usuario ya no pertenece a la institución, se debe inhabilitar la cuenta.

3.3.8.6 Autorización

La autorización es la validación de las funcionalidades que tendrá un usuario al acceder al sistema. Neville-Neil (2007), analiza el problema del acceso y la autorización que brinda una aplicación web a las funcionalidades disponibles ya que una vez un atacante se salta la autenticación tendría disponible roles y privilegios que le permitiría ejecutar acciones sobre la información institucional.

1. Perfiles y roles: Se deben definir perfiles y roles para cada módulo de la aplicación web, de esta forma se puede asegurar que sólo usuarios autorizados puedan realizar acciones permitidas con su correspondiente nivel de privilegio.
2. Acceso a recursos: Todas las funcionalidades de la aplicación deben estar validadas por medio de roles o perfiles que garanticen que solo los usuarios autorizados van a hacer uso de ellas. Esta comprobación se debe hacer siempre que un usuario acceda a cualquier interfaz de la aplicación.
3. La creación de roles y perfiles, así como su asignación, deben ser administrados en cada aplicación web. Excepciones a esta política deben ser autorizadas por el jefe de Sistemas de la institución.

3.3.8.7 Sesiones

Las sesiones son fundamentales en la construcción de nuestros sistemas web, Li y Xue (2014) describen las vulnerabilidades de administración de las sesiones y los posibles ataques que se podrían explotar con dicha vulnerabilidad lo cual sería catastrófico para nuestras aplicaciones ya que es un aspecto importante al momento de desarrollar nuestros aplicativos, pues es el que

da acceso a un usuario para que interactúe con el sistema y que se mantendrá mientras este activo en el mismo.

1. Sesiones permisivas: se debe evitar la generación de sesiones permisivas en las aplicaciones web de la institución, es por esto que se debe tener en cuenta:
    a. Cada vez que se inicializa una sesión para un usuario, se debe primero comprobar que se encuentra fuera del sistema y sin roles asignados.
    b. Antes de dar acceso a cualquier recurso de la aplicación, se debe comprobar el estado de la sesión del usuario.
    c. Las interfaces que requieran estar desprotegidas, deben utilizar menos cantidad de recursos para prevenir ataques de denegación de servicios y no comprometer la información que si está protegida.
2. Sesiones expuestas: se debe configurar el servidor de aplicaciones para que las sesiones usen carpetas temporales diferentes a las establecidas por el marco de aplicación (PHP utiliza /tmp) por cliente y aplicación. En caso de no ser posible, los datos de la sesión deben ser cifrados o no contener información relevante.
3. Credenciales en formularios: se deben incluir credenciales de sesiones criptográficamente seguras y únicas en cada inicio de sesión como campos ocultos en los formularios de la aplicación para asegurar que las peticiones son enviadas por el usuario que inició la sesión. Esta comprobación se debe hacer cada que vez se recibe una solicitud.
4. Sesiones débiles: se debe proteger la aplicación de sesiones criptográficamente débiles, para esto se debe tener en cuenta:
    a. Cada credencial debe tener un origen aleatorio, entre números, letras y símbolos especiales.
    b. Cada credencial debe ser relacionada a una instancia HTTP específica (identificador de la sesión y su dirección IP), para prevenir reutilización de sesiones.
5. Finalizar sesión: las sesiones que no expiran representan un fallo de seguridad crítico, es por esto que se deben tener en cuenta las siguientes recomendaciones:
    a. La aplicación no debe utilizar ningún método para que nunca expire la sesión.

   b. La aplicación no debe tener mecanismos que evadan el tiempo de inactividad.
   c. La aplicación no debe tener ninguna opción de “recuérdame”.
6. Regenerar credencial de sesión: se deben regenerar las credenciales de sesión luego de una cierta cantidad de transacciones o después de un determinado tiempo.
7. Cerrar sesión: Cuando un usuario cierra la sesión ésta se debe destruir y las cookies de sesión se deben sobrescribir.

3.3.8.8 Validación de datos

La validación de datos garantiza la integridad de la información con la que interactúa la aplicación. De acuerdo con Higuera (2014), se debe validar las entradas y salidas comprobando así su longitud, tipo de dato o contenido para evitar posible inyección de código malicioso.

1. Integridad: se debe garantizar que los datos no han sido manipulados y que siguen siendo los mismos. Estas revisiones de integridad se deben incluir en lugares donde no se tenga confianza que el dato se va a mantener, como por ejemplo al pasar un dato de la aplicación al navegador del usuario en un campo oculto. El tipo de control de integridad se determina de acuerdo al riesgo que representa el traspaso de los datos a lugares que no son confiables.
2. Validación: todos los datos manipulados deben ser validados, la falta de validaciones conduce directamente a la anulación de todos los controles de seguridad de la aplicación. Es por esto que se debe tener en cuenta:
   a. Todos los datos deben ser siempre tecleados.
   b. Se debe revisar la longitud, ya que se debe ajustar a la longitud del campo.
   c. Se deben revisar los rangos si se trata de cadenas numéricas.
   d. No se deben permitir símbolos especiales, a menos que el campo lo requiera.
   e. Se debe revisar que el dato ingresado contiene la información que espera el campo, es decir tipo, longitud y sintaxis.
   f. Las validaciones se deben realizar del lado del cliente y del lado del servidor.
3. Campos Ocultos: se debe implementar revisiones de integridad y cifrado a la información contenida en campos ocultos, el cifrado de datos se debe hacer aleatorio

para cada usuario que esté activo en el sistema para garantizar que los datos serán manipulados por el usuario que los accede.

4. Cifrado de URL: los datos enviados por la URL deben ser cifrados para reducir la posibilidad de ataques de tipo cross-site-scripting. Se debe tener en cuenta que en lo posible no se debe enviar datos por medio de peticiones GET.
5. Filtrado de Cadenas: todas las cadenas enviadas al servidor se deben filtrar antes de realizar acciones con ellas. Se deben limpiar los caracteres especiales que puedan contener.

3.3.8.9 Manejo de fallos y seguridad

Un buen manejo de fallos garantiza que la aplicación no muestre información sensible al usuario cuando suceda algo inesperado. De acuerdo con Cacua (2012), es importante implementar mecanismos de manejo de errores y de generación de logs de auditoria para hacer seguimiento a los diferentes fallos que se puedan presentar.

1. Generación de logs: los logs generados por la aplicación deben permitir que solo nueva información pueda ser escrita, es decir los registros anteriores se deben mantener y deben cumplir con las siguientes características:
    a. Se debe adoptar una convención de nombre para todos los archivos de log para facilitar su identificación.
    b. Se debe garantizar que la información contenida en el log no se va a sobrescribir, se debería generar un log diario o por intervalos de tiempo.
    c. Los logs deben ser resguardados de forma segura y deben mantener su confidencialidad por al menos seis meses antes de que se pueda proceder a eliminarlos.
2. Auditorías: Los registros de auditoría de la aplicación deben ser almacenados en lugares donde se garantice que no van a ser modificados y/o eliminados y se debe garantizar que:
    a. Los registros generados no deben viajar en texto claro entre el host y el destino.

b. Los registros deben prevenir su modificación mediante mecanismos como HMAC, los cuales garantizan que la información no es alterada desde el momento de su registro hasta que se realiza su revisión.

3. Manejo de errores: los mensajes de error pueden contener información que permita realizar ataques, es por esto que las aplicaciones desarrolladas por el departamento de Sistemas deben tener mecanismos a prueba de fallos que cumplan con las siguientes características:
    a. El código en producción debe tener implementado mensajes o manipuladores de depuración de errores los cuales deben filtrar información privada que pueda generar ataques posteriores.
    b. Se debe implementar manipuladores de excepciones estructurados (try {} catch {}) o manipulación de errores basado en funciones. Si se usa la manipulación de errores basada en funciones, se debe revisar cada valor de salida ya que debe resolver el error de forma apropiada.
    c. Se deben presentar los mensajes de error de forma genérica sin dar mayor información al usuario.
    d. Se deben validar los códigos de respuesta de estado HTTP (errores 404 o 500).

3.3.8.10 Recursos externos

Los recursos externos son las librerías que se implementan en la aplicación web y que son tomadas de ubicaciones externas. De acuerdo con Cao et al. (2012), estas librerías deben ser tomadas de fuentes confiables ya que no solo podrían aportar características para enriquecer nuestras páginas web sino también amenazas al ser manipuladas por terceras personas.

1. Librerías: las librerías utilizadas en las aplicaciones web deben provenir de fuentes confiables o de sitios oficiales. En caso contrario, se debe escribir las librerías que se requieran.
2. Cifrado: la información sensible se debe proteger mediante el uso de protocolo SSL/TLS, el cual establece un nivel de cifrado que estará ejecutándose hasta que se cierre la sesión con el servidor.

3.3.8.11 Base de datos

Los motores de base de datos son los encargados de almacenar toda la información de la institución es por esto que debe estar diseñada aplicando reglas de integridad y una correcta definición de los permisos para cada aplicación. Mavromoustakos et al. (2016) nos habla de las posibles amenazas que se nos pueden presentar en nuestras bases de datos y como prevenirlas una vez tengamos nuestra aplicación web en producción.

1. Credenciales de acceso: las cadenas de conexión a la base de datos no se deben almacenar en ninguna parte del código. La conexión se debe establecer en lo posible por ODBC indicando un alias de la base de datos.
2. Privilegios: se debe definir los usuarios que se conectan a la aplicación web de acuerdo a su nivel de confianza, es decir diferenciando entre usuarios, usuarios invitados, de solo lectura, administradores, entre otros, a los cuales se les debe aplicar los permisos a aquellas tablas que se necesiten utilizar para prevenir accesos no autorizados y modificaciones.
3. Creación de tablas: las tablas creadas para cada módulo de la aplicación web deben contar con reglas de integridad (llaves primarias y foráneas) para garantizar su buen desempeño al momento de realizar ejecuciones.
4. Implementación: la base de datos se debe implementar en una capa de red diferente a la de la aplicación web y debe estar actualizada en lo posible a su última versión.
5. El acceso a las bases de datos de desarrollo, pruebas y producción, debe contar con controles de autenticación y autorización. Se deben definir roles sobre las personas que tienen acceso a la base de datos en sus diferentes ambientes y qué tipo de acceso (consultar, actualizar, eliminar). Durante el proceso de desarrollo o pruebas, no se debe dar acceso a datos de producción.
6. El administrador de la base de datos debe ser una persona que solo cumpla esta función, encargándose de revisar e implementar los modelos de datos diseñados para cada aplicación web.

3.3.8.12 Servicios Web

Los servicios web permiten el intercambio de datos entre aplicaciones, para su correcto funcionamiento se debe validar la autenticación mutua y la seguridad de extremo a extremo para garantizar la integridad de los mensajes intercambiados. Yao, Koglin, Bertino y Tamassia (2007) nos aportan buenas prácticas para mantener la integridad de los datos compartidos por medio del uso de roles y certificados de roles para gestionar a los diferentes intermediarios.

1. Seguridad de comunicación: se debe establecer seguridad de extremo a extremo para cada mensaje enviado a un usuario, ya que se debe garantizar la comunicación entre los dos interlocutores a lo largo del recorrido.
2. Autenticación: se debe garantizar la autenticación entre el usuario y el servidor que provee el servicio, para ello se debe validar por medio de un token único generado cada vez que el usuario inicie sesión y almacenado en la base de datos para consultarlo cada vez que se tiene una petición.
3. Integridad y confidencialidad de los mensajes: se debe garantizar la confidencialidad de los mensajes intercambiados, para esto se deben enviar los mensajes de forma cifrada garantizando que el mensaje enviado es el mismo que va a ser recibido.
4. No repudio: se debe garantizar que cuando se realizan transacciones es posible comprobar qué acción tuvo lugar y por quien fue realizada, con la finalidad de poder demostrar quien utilizó los servicios, pese a que lo niegue.
5. Auditoría: se deben mantener trazas de todas las acciones que se realizan en los servicios web para realizar análisis en caso de escenarios de desastre.

3.3.8.13 Control de versiones

1. La documentación, código fuente y librerías desarrolladas, scripts de la base de datos debe estar bajo procedimientos de control de cambios y de versionamiento.
2. El departamento de Sistemas debe mantener un registro actualizado de las aplicaciones en producción, con datos referentes a la versión, fecha de última compilación y responsables de su soporte.

3.3.8.14 Configuración Física

La configuración física hace referencia a la plataforma que soporta la aplicación a nivel de software (sistemas operativos, sistema de gestión de base de datos, servidor web entre otros) y servidores nuestra aplicación web. De acuerdo con la OWASP (2017) es importante mantener nuestra plataforma actualizada para así minimizar cualquier vulnerabilidad del software que la soporta así mismo se debe tener una buena configuración a nivel de permisos de ficheros y archivos.

1. Permisos de fichero y archivos: se debe establecer permisos sobre los archivos en 644 y sobre las carpetas en 755 para asegurar el contenido de los mismos.
2. Definición de usuarios: se debe definir los roles de las personas y los permisos con los que ingresan a los servidores de aplicaciones y de base de datos.
3. Actualización de software: se debe mantener actualizado el software que soporta la aplicación web (sistema operativo, servidor web/App, motor de base de datos, API y todos los componentes de bibliotecas).
4. Puertos y servicios: sólo se deben habilitar los puertos, servicios y funciones que sean necesarios para el buen funcionamiento de la aplicación web.
5. Mantenimiento: Se debe programar ventanas de mantenimiento para revisar logs del sistema y dar solución a posibles fallos, así como para actualizar el software que soporta la aplicación web.
6. Indexación: se debe utilizar el archivo robots.txt para impedir que los motores de búsqueda, busquen más allá de lo especificado en el archivo.

3.3.8.15 Cambios plataforma operativa

Cuando se presenten cambios, actualizaciones o reconfiguración en los servidores de aplicaciones y de bases de datos, la coordinación de redes y servidores debe realizar un análisis y emitir un informe técnico en el cual debe evaluar los impactos y riesgos que puedan generar los cambios.

3.3.8.16 Tercerización de software

1. La adquisición de software a terceros se debe realizar considerando aspectos y atributos de seguridad de la información, así como el impacto en la seguridad frente a cambios que deban realizarse para su implementación.
2. Las actualizaciones al código fuente de software tercerizado que surjan de la puesta en producción y tengan relación con la seguridad, deben ser aprobadas por el jefe de Sistemas de la institución.

3.3.8.17 Capacitación

La liberación a producción de los aplicativos web, sean construidos por el departamento o adquiridos a terceros, deben siempre incluir actividades de capacitación dirigida a usuarios finales y administradores de plataforma, así como al profesional de la mesa de ayuda.

3.4 Validación de la propuesta

Mediante el proceso de validación de la propuesta se busca establecer la validez de la política de desarrollo web seguro con relación a los indicadores que fueron establecidos para el desarrollo de la misma. Estos indicadores definen la pertinencia que va a contener el modelo de evaluación. Para llevar a cabo la validación se utiliza el método analítico, el cual permite analizar la política de desarrollo web seguro en todas sus partes. De igual forma se utiliza el método Delphi, el cual será aplicado a través de un grupo de expertos que se relacionaran más adelante.

3.4.1 Método Analítico

De acuerdo con (Ruiz, 2006), el método analítico consiste en la desmembración de un todo, descomponiéndolo en sus partes para observar las causas, su naturaleza y los efectos. El análisis es la observación y evaluación de un hecho particular, es necesario conocer la naturaleza del objeto de estudio para conocer su esencia. Este método se aplica en el presente trabajo, ya que ayuda a identificar los procesos que se deben tener en cuenta al desarrollar software web seguro y de esta forma analizarlos a partir de la definición de indicadores.

Teniendo en cuenta el método analítico, a continuación se describen los factores involucrados en el proceso de desarrollo de software web seguro.

3.4.1.1 Factores proceso de desarrollo seguro

Para la definición de los factores se tiene en cuenta los parámetros dictados por la Open Web Application Security Project (OWASP), SysAdmin Audit, Networking and Security Institute (SANS) y Web Application Security Consortium (WASC), los cuales sirven como guía para construir aplicaciones web seguras, por ello los factores a tener en cuenta son los siguientes:

- Responsables

Son las personas que tendrán a cargo las responsabilidades en el diseño y planeación de los aplicativos web (jefe de departamento, coordinador de desarrollo, programador, entre otros).

- Definición de requerimientos de seguridad

Son los requisitos de seguridad con los cuales debe cumplir la aplicación. Estos se deben definir al inicio de cada fase de desarrollo.

- Autenticación

Es el procedimiento que permite validar que un usuario del aplicativo web es el que está autorizado para acceder al sistema.

- Autorización

La autorización es la validación de las funcionalidades que tendrá un usuario al acceder al sistema.

- Sesiones

Es el acceso que se otorga a un usuario que está interactuando con el sistema y que se mantendrá mientras esté activo en el mismo.

- Validación de datos

Por medio de la validación de datos se garantiza la integridad de la información con la que interactúa la aplicación.

- Manejo de fallos y seguridad

El manejo de fallos garantiza que la aplicación no muestre información sensible al usuario cuando suceda algo inesperado.

- Recursos externos

Son las librerías y demás recursos que se implementan en la aplicación y que son tomadas de ubicaciones externas.

- Base de datos

Es donde se almacena toda la información de la institución la cual debe estar diseñada con reglas de integridad y una correcta definición de los permisos para cada aplicación.

- Servicios Web

Permite el intercambio de datos entre aplicaciones, para su correcto funcionamiento se debe validar la autenticación mutua y la seguridad de extremo a extremo para garantizar la integridad de los mensajes intercambiados.

- Configuración física

La configuración física hace referencia a la plataforma que soporta la aplicación a nivel de software (sistemas operativos, sistema de gestión de base de datos, servidor web, entre otros).

3.4.1.2 Indicadores de validación

Una vez se tienen los factores definidos para el proceso de validación, se descomponen en indicadores más específicos para así evaluar cada factor. Estos indicadores se pueden detallar en la tabla 7, la cual corresponde a la matriz de validación en la que se muestra la ponderación de los factores e indicadores. Los factores son ponderados de acuerdo al peso dentro del proceso de desarrollo de software web seguro.

Tabla 7 Matriz de validación
Diseño propio

PONDERACIÓN DE FACTORES E INDICADORES	PONDERACIÓN	OBSERVACIÓN
Factor 1: Responsables	**10%**	
Indicador 1: ¿Es importante establecer las responsabilidades de los involucrados en el desarrollo de software web seguro?	10%	
Factor 2: Definición requerimientos de seguridad	**10%**	
Indicador 2: ¿La definición de requerimientos de seguridad es un aspecto importante para el desarrollo de software web seguro?	5%	
Indicador 3: ¿La participación de usuarios finales es importante para el proceso de construcción de controles de seguridad del aplicativo?	5%	
Factor 3: Autenticación	**10%**	
Indicador 4: ¿La forma como se asignan los nombres de usuario en una aplicación web es importante para el control de seguridad de la misma?	2%	
Indicador 5: ¿Las contraseñas seguras deben incluir parámetros de seguridad adicionales como caracteres especiales, así como la opción de restablecimiento y cambio?	5%	
Indicador 6: ¿Es importante establecer si se detecta inactividad o inicio de sesión en diferentes ubicaciones?	3%	
Factor 4: Autorización	**10%**	
Indicador 7: ¿El uso de roles y perfiles es importante para controlar el acceso a las funcionalidades de la aplicación web por parte de los usuarios?	10%	
Factor 5: Sesiones	**10%**	
Indicador 8: ¿Cada usuario del sistema debe contar con credenciales criptográficas únicas?	5%	
Indicador 9: ¿las sesiones deben contar con mecanismos para que finalicen ya sea de forma automática o por petición del usuario?	5%	
Factor 6: Validación de Datos	**10%**	
Indicador 10: ¿Los datos de entrada de las aplicaciones web se deben manipular de forma adecuada (tipos de datos, longitud) para garantizar de esta forma su integridad y seguridad?	10%	
Factor 7: Manejo de Fallos y Seguridad	**10%**	
Indicador 11: ¿Es importante proteger la aplicaciones web cuando se presentan fallos haciendo buen manejo de los errores generados y almacenando logs y auditorias de los mismos?	10%	
Factor 8: Recursos Externos	**5%**	
Indicador 12: ¿El uso de recursos de terceros como lo son las librerías o bibliotecas debe ser de fuentes oficiales o confiables para así garantizar que no vamos a tener código corrupto en nuestros aplicativos web?	5%	
Factor 9: Base de Datos	**10%**	

Indicador 13: ¿Las credenciales de acceso de la base de datos y el nivel de privilegios de la misma deben ser los adecuados para el usuario con el que se realiza la conexión?	5%	
Indicador 14: ¿Las tablas de la base de datos deben contar con reglas de integridad?	5%	
Factor 10: Servicios Web	**10%**	
Indicador 15: ¿Los servicios web deben tener implementada seguridad de extremo a extremo y auditorias para identificar las trazas de los mensajes intercambiados?	10%	
Factor 11: Configuración Física	**5%**	
Indicador 16: ¿Una buena configuración de la plataforma operativa ayuda a proteger las aplicaciones web si se establecen controles como permisos sobre las carpetas y archivos, se mantiene actualizado el software y se aplican mantenimientos periódicos?	5%	

3.4.2 Método Delphi

El método Delphi es una técnica de comunicación estructurada que exige un consenso de un grupo de expertos con respecto a un tema particular. Para (García, 2013) es una metodología estructurada que se utiliza para recolectar sistemáticamente juicios de expertos sobre un problema, procesar la información y con la ayuda de métodos estadísticos construir un acuerdo general de grupo. El método Delphi se ajusta al proceso de validación de la estrategia de desarrollo de software web seguro planteada en la presente investigación.

Aplicación método Delphi

Para evaluar la política de desarrollo web seguro a través del método Delphi, se contó con la colaboración de tres expertos en el área de desarrollo de software. Las personas seleccionadas tienen amplia experiencia en el desarrollo de software tanto en ambientes web como de escritorio. A continuación se relacionan los expertos en la Tabla 8.

Tabla 8 Panel de expertos
Diseño propio

Experto	Profesión	Empresas	Áreas de desempeño	Experiencia
1	Ingeniero de sistemas	SID (soluciones e indicaciones digitales), Universidad Francisco de Paula Santander	Ingeniero de software y desarrollo web, Analista de Sistemas.	7 años
2	Ingeniero de Sistemas	GRUPO ASD S.A.S – BOGOTÁ, ACSENDO S.A.S - BOGOTÁ	Desarrollador de Software Senior, Senior II Solutions Engineer	5 años

3	Ingeniero de sistemas	ExpertosHosting, FyM Ingenieros S.A.S., Inprosistemas del Norte, Signa Grein (Bogotá)	Desarrollador de sistemas web, Ingeniero de software y desarrollo.	10 años

Una vez se tiene definido el panel de expertos, se aplica el formulario que se detalla en la Tabla 9. En este formulario se establece que cada experto evalué de 1 a 10 cada indicador de acuerdo a su criterio. También se deja la opción que cada experto aporte alguna observación en cada indicador según lo considere. Los resultados de las calificaciones de cada experto se pueden observar en el Anexo 5 y se relaciona en la Tabla 10. Para llevar a cabo el diligenciamiento los expertos deben seguir las siguientes reglas:

1. El proceso de validación debe ser confidencial para garantizar que los criterios no están influenciados.
2. En caso de no tener consenso en los indicadores se tomará en cuenta las observaciones registradas por cada experto al momento de la evaluación.
3. La diferencia entre las calificaciones no debe exceder 3 puntos para los casos de consenso, esto teniendo en cuenta que se hace la validación con un grupo reducido de expertos.

Tabla 9 Formulario de evaluación aplicando método Delphi
Diseño propio

Formulario de evaluación de la política de desarrollo de software web seguro de la Universidad Simón Bolívar sede Cúcuta aplicando método Delphi											
Indicador	**Calificación**										**Observación**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Indicador 1											
Indicador 2											
Indicador 3											
Indicador 4											

La matriz de resultados de la evaluación realizada por los expertos que se observa en la Tabla 10, está compuesta por los siguientes datos:

1. Factores e indicadores: enumera los factores e indicadores relacionados en la Tabla 7.
2. Calificación: es la calificación otorgada por cada uno de los expertos.
3. Promedio: se calcula de acuerdo al peso y calificación de los expertos.

4. Variación estándar: define la variación de acuerdo a las calificaciones asignadas por los experto a cada indicador.
5. Ponderación: hace referencia a la ponderación de cada factor e indicador como se observa en la Tabla 7.
6. Valor Alcanzado: Se calcula de acuerdo al promedio de las calificaciones de los expertos y los pesos de cada factor relacionados en la tabla 7.

4 Conclusiones, resultados y trabajos futuros

4.1 Conclusiones

1. Se elaboró el estado del arte de la seguridad en el desarrollo de software web mediante la consulta bibliográfica y revisión de bases de datos electrónicas.
2. Se diseñó la estrategia de desarrollo de software web seguro para el departamento de Sistemas, la cual se propuso como una política de seguridad institucional.
    a. Se tomó como insumo el estado del arte y el marco teórico y teniendo en cuenta el estado actual de la institución, se vio la necesidad de diseñar la política de desarrollo web seguro institucional.
    b. La estrategia de desarrollo de software web seguro se diseñó teniendo en cuenta los parámetros de las instituciones que dictan los aspectos de seguridad en ambientes web, así como el modelo de desarrollo de software seguro.
    c. La estrategia se implementará como política institucional al interior del Departamento de Sistemas de la institución y garantizará que se desarrollen sistemas seguros, ya que se tendrán buenas prácticas para reducir las vulnerabilidades en los sistemas antes de dejarlos en producción.
3. De los resultados obtenidos en la aplicación del instrumento en las instituciones del departamento Norte de Santander, se puede concluir:
    a. Las instituciones cuentan con políticas de desarrollo seguro en sus instituciones para sus sistemas de información web.

b. Un 80% de instituciones consultadas cuentan con una arquitectura de desarrollo, este aspecto es importante ya que cuentan con una metodología de trabajo establecida y estandarizada.
c. Las instituciones de la región consultadas, garantizan una buena autenticación en sus aplicaciones web. Se debe mejorar en el tema de encriptación de contraseñas para hacerlas más fuertes y en el cambio de la misma de forma periódica.
d. Las instituciones diseñan sus bases de datos aplicando reglas de integridad, tienen control sobre los privilegios dados a los usuarios con el que se conecta la aplicación. Se debe mejorar en la forma como se incluyen las credenciales de acceso en el código, ya que esta conexión debe estar oculta en la aplicación.
e. Las aplicaciones de las instituciones, tienen establecidos control sobre sus funcionalidades por medio de roles o perfiles. Se debe mejorar los controles al momento de enviar peticiones ya que siempre es necesario comprobar que es enviada por el usuario que inicia la sesión.
f. Es importante tener actualizado el software que soporta las aplicaciones web, ya que ayudan a combatir las vulnerabilidades. Las instituciones participantes tienen control sobre la plataforma tecnológica manteniéndola actualizada y aplicando ventanas de mantenimiento de forma periódica.
g. En términos generales las instituciones de la región tienen precauciones para proteger los aplicativos de ataques informáticos, pues establecen controles para evitar que éstos fallen en caso de detectar mal uso sobre ellos.

4.2 Resultados

4.2.1 Resultados de la validación

En la Tabla 10 se muestra los resultados de la validación realizada por el panel de expertos.

Tabla 10 Matriz de resultados
Diseño propio

Matriz de resultados política de desarrollo de software web seguro de la Universidad Simón Bolívar sede Cúcuta							
Factores e Indicadores	**Calificación Experto 1**	**Calificación Experto 2**	**Calificación Experto 3**	**Promedio del Consenso**	**Variación Estándar**	**Ponderación**	**Valor Alcanzado**
Factor 1: Responsables						**10%**	**9,7%**
Indicador 1	10	9	10	9,7	0,6	10%	9,7%
Factor 2: Definición requerimientos de seguridad						**10%**	**9,8%**
Indicador 2	10	9	10	9,7	0,6	5%	4,8%
Indicador 3	10	10	10	10,0	0,0	5%	5,0%
Factor 3: Autenticación						**10%**	**9,3%**
Indicador 4	8	5	8	7,0	1,7	2%	1,4%
Indicador 5	10	10	10	10,0	0,0	5%	5,0%
Indicador 6	10	9	10	9,7	0,6	3%	2,9%
Factor 4: Autorización						**10%**	**10,0%**
Indicador 7	10	10	10	10,0	0,0	10%	10,0%
Factor 5: Sesiones						**10%**	**9,8%**
Indicador 8	10	10	10	10,0	0,0	5%	5,0%
Indicador 9	10	9	10	9,7	0,6	5%	4,8%
Factor 6: Validación de Datos						**10%**	**10,0%**
Indicador 10	10	10	10	10,0	0,0	10%	10,0%
Factor 7: Manejo de Fallos y Seguridad						**10%**	**9,7%**
Indicador 11	10	9	10	9,7	0,6	10%	9,7%
Factor 8: Recursos Externos						**5%**	**5,0%**
Indicador 12	10	10	10	10,0	0,0	5%	5,0%
Factor 9: Base de Datos						**10%**	**9,7%**
Indicador 13	10	9	10	9,7	0,6	5%	4,8%
Indicador 14	10	9	10	9,7	0,6	5%	4,8%
Factor 10: Servicios Web						**10%**	**10,0%**
Indicador 15	10	10	10	10,0	0,0	10%	10,0%
Factor 11: Configuración Física						**5%**	**4,8%**
Indicador 16	10	9	10	9,7	0,6	5%	4,8%

Concluida la consulta a los expertos, se puede observar que los promedios de la mayoría de los indicadores obtuvieron una calificación superior a 9, lo cual indica un alto grado de aceptabilidad y consenso por parte del panel de expertos. El indicador más bajo fue el referente al de la creación de usuarios, ya que los expertos no lo ven como algo que pueda afectar la seguridad de la aplicación, pero si es importante establecer la forma como se debe asignar el nombre de usuario para cada uno de los actores del sistema.

Teniendo en cuenta los valores obtenidos en la evaluación de cada uno de los expertos, se puede concluir que el resultado de la validación es exitoso y es un indicio de pertinencia de la política de desarrollo de software web seguro en la Universidad Simón Bolívar sede Cúcuta.

4.2.2 Otros resultados de la investigación

a. El estado del arte construido en la presente investigación, fue enviado para publicación un artículo en la revista INGENIO UFPSO p-ISSN: 2011-642X e-ISSN: 2389-864X, revista reconocida por Colciencias en la Categoría C en la II actualización de revistas especializadas de 2014 (Anexo 6).
b. El producto de la investigación se establecerá como política institucional al interior del Departamento de Sistemas para futuras construcciones de software web.

4.3 Trabajos futuros

Como trabajos futuros que se pueden realizar de acuerdo a los resultados de la presente investigación, se pueden nombrar los siguientes:

a. Realizar la validación de la política de desarrollo de software web seguro con un proyecto que esté por iniciar en la institución, en el cual se pueda aplicar todas las fases de la política desde la toma de requerimientos hasta su puesta en producción.
b. Realizar ajustes a la política de acuerdo a las necesidades que puedan surgir más adelante al interior del Departamento de Sistemas de la institución.
c. De acuerdo a los resultados obtenidos en la investigación, se puede establecer una estrategia para proteger otros procesos que hacen parte del desarrollo como lo son la infraestructura tecnológica, la seguridad en redes, seguridad en base de datos, entre otras.
d. Tomar como referencia la política establecida para futuros procesos de auditoría sobre los sistemas desarrollados.
e. Publicar un artículo en una revista reconocida por Colciencias como categoría A con los resultados de la investigación.

5 Referencias Bibliográficas

OWASP. (2017). OWASP Top 10 2017. Open Web Application Security Project. Retrieved from https://www.owasp.org/index.php/Top_10_2017-Top_10

IETF. (1999). Site Security Handbook. R F C 2196. IETF. Retrieved from http://www.ietf.org/rfc/rfc2196.txt

Howard M, L. D. (2003). *Writing Secure Code. 2nd ed.* Microsoft Press. ISBN 0-7356-1722-8.

Garrett, J. J. (2005). Ajax: A New Approach to Web Applications. Retrieved from http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

SWEBOK. (2014). *Guide to the Software Engineering Body of Knowledge Version 3.0 - SWEBOK®*. (É. de technologie supérieure (ÉTS) Pierre Bourque, S. Richard E. (Dick) Fairley, & S. E. A. (S2EA), Eds.). Copyright © 2014 IEEE. All rights reserved. ISBN-10: 0-7695-5166-1.

wcreator. (2017). Qué es MVC y por que es tan usado en el desarrollo Web. Retrieved from http://www.wcreator.com.ve/noticias/item/32-que-es-mvc-y-por-que-es-tan-usado-en-el-desarrollo-web

OWASP. (2005). Una Guía para Construir Aplicaciones y Servicios Web Seguros Edición 2.0 Black Hat. https://www.owasp.org/. Retrieved from https://www.owasp.org/images/b/b2/OWASP_Development_Guide_2.0.1_Spanish.pdf

ISO17799. (2005). ISO/IEC 17799. International Standards Organization. Retrieved from https://www.iso.org/standard/39612.html

ISO27001. (2005). ISO/IEC 27001. International Standards Organization. Retrieved from https://www.iso.org/standard/54534.html

ISACA. (2012). *COBIT 5 for Information Security ISBN 978-1-60420-255-7*. Retrieved from www.isaca.org/cobit5info-sec

Systems, W. R. (2015). THE CWE/SANS TOP 25 SECURITY VULNERABILITIES.

ISO27002. (2007). ISO/IEC 27002. International Organization for Standardization. Retrieved from https://www.iso.org/standard/54533.html

WASC. (2011). The WASC Threat Classification v2.0. WASC. Retrieved from http://projects.webappsec.org/w/page/13246978/Threat Classification

OASIS. (2006). OASIS Web Services Security (WSS) TC. Organization for the Advancement of Structured Information Standards. Retrieved from https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

W3C. (2007). Web Services Policy 1.5 - Framework. W3C. Retrieved from https://www.w3.org/TR/ws-policy/

Criteria. (2017). Common Criteria for Information Technology Security Evaluation. Common Criteria. Retrieved from https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf

Hinrichs, T. L., Rossetti, D., Petronella, G., Venkatakrishnan, V. N., Sistla, A. P., & Zuck, L. D. (2013). WEBLOG: A Declarative Language for Secure Web Development. In *Proceedings of the Eighth ACM SIGPLAN Workshop on Programming Languages and Analysis for Security* (pp. 59–70). New York, NY, USA: ACM. https://doi.org/10.1145/2465106.2465119

Ge, X., Paige, R. F., Polack, F. A. C., Chivers, H., & Brooke, P. J. (2006). Agile Development of Secure Web Applications. In *Proceedings of the 6th International Conference on Web Engineering* (pp. 305–312). New York, NY, USA: ACM. https://doi.org/10.1145/1145581.1145641

Reischuk, R. M., Schröder, F., & Gehrke, J. (2013). DEMO: Secure and customizable web development in the safe activation framework. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 1341–1344). New York, NY, USA: ACM. https://doi.org/10.1145/2508859.2512495

Li, X., & Xue, Y. (2014). A Survey on Server-side Approaches to Securing Web Applications. *ACM Comput. Surv.*, *46*(4), 54:1--54:29. https://doi.org/10.1145/2541315

Li, R., Dong, M., Liu, B., Lu, J., Ma, X., & Li, K. (2010). SecTag: A Multi-policy Supported Secure Web Tag Framework. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (pp. 633–635). New York, NY, USA: ACM. https://doi.org/10.1145/1866307.1866379

Xie, J., Chu, B., Lipford, H. R., & Melton, J. T. (2011). ASIDE: IDE Support for Web Application Security. In *Proceedings of the 27th Annual Computer Security Applications Conference* (pp. 267–276). New York, NY, USA: ACM. https://doi.org/10.1145/2076732.2076770

Rahman, S., Nguyen, T. A., & Yang, T. A. (2006). Developing Certificate-based Projects for Web Security Classes. *J. Comput. Sci. Coll.*, *21*(5), 28–37. Retrieved from http://dl.acm.org/citation.cfm?id=1127351.1127359

Walden, J., & Frank, C. E. (2007). Web Application Security Tutorial. *J. Comput. Sci. Coll.*, *23*(1), 77–78. Retrieved from http://dl.acm.org/citation.cfm?id=1289280.1289294

Scott, D., & Sharp, R. (2002). Abstracting Application-level Web Security. In *Proceedings of the 11th International Conference on World Wide Web* (pp. 396–407). New York, NY, USA: ACM. https://doi.org/10.1145/511446.511498

Yao, D., Koglin, Y., Bertino, E., & Tamassia, R. (2007). Decentralized Authorization and Data Security in Web Content Delivery. In *Proceedings of the 2007 ACM Symposium on Applied Computing* (pp. 1654–1661). New York, NY, USA: ACM. https://doi.org/10.1145/1244002.1244353

Okubo, T., & Tanaka, H. (2008). Web Security Patterns for Analysis and Design. In *Proceedings of the 15th Conference on Pattern Languages of Programs* (p. 25:1--25:13). New York, NY, USA: ACM. https://doi.org/10.1145/1753196.1753226

Whitson, G. (2008). Security for Service Oriented Architectures. *J. Comput. Sci. Coll.*, *23*(4), 8–9. Retrieved from http://dl.acm.org/citation.cfm?id=1352079.1352083

Walden, J., & Frank, C. E. (2006). Secure Software Engineering Teaching Modules. In *Proceedings of the 3rd Annual Conference on Information Security Curriculum Development* (pp. 19–23). New York, NY, USA: ACM. https://doi.org/10.1145/1231047.1231052

Sathiaseelan, J. G. R., Rabara, S. A., & Martin, J. R. (2009). Multi-Level Secure Framework (MLSF) for Composite Web Services. In *Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (pp. 580–585). New York, NY, USA: ACM. https://doi.org/10.1145/1655925.1656030

Kazmierski, T., & Yang, X. Q. (2003). A Secure Web-Based Framework for Electronic System Level Design. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1* (p. 11140--). Washington, DC, USA: IEEE Computer Society. Retrieved from http://dl.acm.org/citation.cfm?id=789083.1022891

Zhu, J., Lipford, H. R., & Chu, B. (2013). Interactive Support for Secure Programming Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (pp. 687–692). New York, NY, USA: ACM. https://doi.org/10.1145/2445196.2445396

Chan, A. T. S. (2004). Cookies On-the-move: Managing Cookies on a Smart Card. In *Proceedings of the 2004 ACM Symposium on Applied Computing* (pp. 1693–1697). New York, NY, USA: ACM. https://doi.org/10.1145/967900.968236

Sirer, E. G., & Wang, K. (2002). An Access Control Language for Web Services. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies* (pp. 23–30). New York, NY, USA: ACM. https://doi.org/10.1145/507711.507715

Salminen, A., Mikkonen, T., Nyrhinen, F., & Taivalsaari, A. (2010). Developing Client-side Mashups: Experiences, Guidelines and the Road Ahead. In *Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 161–168). New York, NY, USA: ACM. https://doi.org/10.1145/1930488.1930523

Du, W., Jayaraman, K., Tan, X., Luo, T., & Chapin, S. (2011). Position Paper: Why Are There So Many Vulnerabilities in Web Applications? In *Proceedings of the 2011 Workshop on New Security Paradigms Workshop* (pp. 83–94). New York, NY, USA: ACM. https://doi.org/10.1145/2073276.2073285

Smith, M., & Dehlinger, J. (2014). Enabling Static Security Vulnerability Analysis in PHP Applications for Novice Developers with SSVChecker. In *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems* (pp. 278–283). New York, NY, USA: ACM. https://doi.org/10.1145/2663761.2664213

Cao, Y., Li, Z., Rastogi, V., Chen, Y., & Wen, X. (2012). Virtual Browser: A Virtualized Browser to Sandbox Third-party JavaScripts with Enhanced Security. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (pp. 8–9). New York, NY, USA: ACM. https://doi.org/10.1145/2414456.2414460

Mundada, Y., Feamster, N., & Krishnamurthy, B. (2016). Half-Baked Cookies: Hardening Cookie-Based Authentication for the Modern Web. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (pp. 675–685). New York, NY, USA: ACM. https://doi.org/10.1145/2897845.2897889

Simpkins, L., Yuan, X., Modi, J., Zhan, J., & Yang, L. (2015). A Course Module on Web Tracking and Privacy. In *Proceedings of the 2015 Information Security Curriculum Development Conference* (p. 10:1--10:7). New York, NY, USA: ACM. https://doi.org/10.1145/2885990.2886000

Grubbs, P., McPherson, R., Naveed, M., Ristenpart, T., & Shmatikov, V. (2016). Breaking Web Applications Built On Top of Encrypted Data. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1353–1364). New York, NY, USA: ACM. https://doi.org/10.1145/2976749.2978351

Bhargavan, K., Corin, R., Fournet, C., & Gordon, A. D. (2007). Secure Sessions for Web Services. *ACM Trans. Inf. Syst. Secur.*, *10*(2). https://doi.org/10.1145/1237500.1237504

Adida, B. (2008). Sessionlock: Securing Web Sessions Against Eavesdropping. In *Proceedings of the 17th International Conference on World Wide Web* (pp. 517–524). New York, NY, USA: ACM. https://doi.org/10.1145/1367497.1367568

Mavromoustakos, S., Patel, A., Chaudhary, K., Chokshi, P., & Patel, S. (2016). Causes and Prevention of SQL Injection Attacks in Web Applications. In *Proceedings of the 4th International Conference on Information and Network Security* (pp. 55–59). New York, NY, USA: ACM. https://doi.org/10.1145/3026724.3026742

Gutierrez, C., Fernandez-Medina, E., & Piattini, M. (2005). Web Services Enterprise Security Architecture: A Case Study. In *Proceedings of the 2005 Workshop on Secure Web Services* (pp. 10–19). New York, NY, USA: ACM. https://doi.org/10.1145/1103022.1103025

Oliveira, R. A., Laranjeiro, N., & Vieira, M. (2012). WSFAggressor: An Extensible Web Service Framework Attacking Tool. In *Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference* (p. 2:1--2:6). New York, NY, USA: ACM. https://doi.org/10.1145/2405146.2405148

Bates, A., Hassan, W. U., Butler, K., Dobra, A., Reaves, B., Cable, P., … Schear, N. (2017). Transparent Web Service Auditing via Network Provenance Functions. In *Proceedings of the 26th International Conference on World Wide Web* (pp. 887–895). Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee. https://doi.org/10.1145/3038912.3052640

Neville-Neil, G. V. (2007). Building Secure Web Applications. *Queue*, *5*(5), 22–26. https://doi.org/10.1145/1281881.1281889

Near, J. P., & Jackson, D. (2016). Finding Security Bugs in Web Applications Using a Catalog of Access Control Patterns. In *Proceedings of the 38th International Conference on Software Engineering* (pp. 947–958). New York, NY, USA: ACM. https://doi.org/10.1145/2884781.2884836

Busch, M., Koch, N., Masi, M., Pugliese, R., & Tiezzi, F. (2012). Towards Model-driven Development of Access Control Policies for Web Applications. In *Proceedings of the Workshop on Model-Driven Security* (p. 4:1--4:6). New York, NY, USA: ACM. https://doi.org/10.1145/2422498.2422502

Ikram, M., Asghar, H., Kaafar, M. A., & Mahanti, A. (2014). On the Intrusiveness of JavaScript on the Web. In *Proceedings of the 2014 CoNEXT on Student Workshop* (pp. 31–33). New York, NY, USA: ACM. https://doi.org/10.1145/2680821.2680837

Miller, C. S., & Connolly, R. (2015). Introduction to the Special Issue on Web Development. *Trans. Comput. Educ.*, *15*(1), 1:1--1:5. https://doi.org/10.1145/2724759

Whitney, M., Lipford-Richter, H., Chu, B., & Zhu, J. (2015). Embedding Secure Coding Instruction into the IDE: A Field Study in an Advanced CS Course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 60–65). New York, NY, USA: ACM. https://doi.org/10.1145/2676723.2677280

Calzavara, S., Tolomei, G., Bugliesi, M., & Orlando, S. (2014). Quite a Mess in My Cookie Jar!: Leveraging Machine Learning to Protect Web Authentication. In *Proceedings of the 23rd International Conference on World Wide Web* (pp. 189–200). New York, NY, USA: ACM. https://doi.org/10.1145/2566486.2568047

Witschey, J., Zielinska, O., Welk, A., Murphy-Hill, E., Mayhorn, C., & Zimmermann, T. (2015). Quantifying Developers' Adoption of Security Tools. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (pp. 260–271). New York, NY, USA: ACM. https://doi.org/10.1145/2786805.2786816

De Groef, W., Massacci, F., & Piessens, F. (2014). NodeSentry: Least-privilege Library Integration for Server-side JavaScript. In *Proceedings of the 30th Annual Computer Security Applications Conference* (pp. 446–455). New York, NY, USA: ACM. https://doi.org/10.1145/2664243.2664276

Basilico, N., Gatti, N., & Amigoni, F. (2009). Developing a Deterministic Patrolling Strategy for Security Agents. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02* (pp. 565–572). Washington, DC, USA: IEEE Computer Society. https://doi.org/10.1109/WI-IAT.2009.212

López, M. D. R., Botero, D. M. S., & Durango, C. N. M. (2011). Firma digital: instrumento de transmisión de información a entidades financieras. *Avances En Sistemas E Informática; Vol. 8, Núm. 1 (2011); 7-14 Avances En Sistemas E Informática; Vol. 8, Núm. 1 (2011); 7-14 1909-0056 1657-7663*. Retrieved from http://www.bdigital.unal.edu.co/28822/

Chamorro López, J. A. (2011). *Modelo para la evaluación en seguridad informática a productos software, basado en el estándar ISO/IEC 15408 Common Criteria*. Universidad Icesi. Retrieved from http://hdl.handle.net/10906/67925

Velasco Burbano, G. A. (2011). *Modelo para la evaluación y selección de un software de seguridad para controlar el ciclo de vida de la identidad digital*. Santiago de Cali: Universidad Icesi. Retrieved from http://hdl.handle.net/10906/68006

Mogollon Mantilla, J. A. (2012). *Análisis de viabilidad de la aplicación de métodos de cifrado al Framework EFI.* Universidad de Pamplona, Pamplona , Colombia.

Cacua Velandia, Y. F. (2012). Implementación de Mecanismos para el Manejo de Errores, Auditoría y Generación de Logs al Framework EFI. Universidad de Pamplona, Colombia.

Bolaños Rodríguez, H. A. (2013). *Método para Evaluar la Madurez en la Implementación de Principios y Patrones SOA en un Contexto Empresarial:MeMSOA*. Universidad de San Buenaventura Cali. Retrieved from http://hdl.handle.net/10819/3171

Pabón, F. O. M., Guerrero, J. C., Cuenca, R. H., Rendón, O. M. C., & Guaca, J. A. H. (2010). Seguridad basada en parámetros SIM para entornos de comercio electrónico móvil. *Ingeniería E Investigación; Vol. 27, Núm. 2 (2007); 56-64 Ingeniería E Investigación; Vol. 27, Núm. 2 (2007); 56-64 2248-8723 0120-5609*. Retrieved from http://www.bdigital.unal.edu.co/18890/

Rosero, B. E. P., & Luna, J. A. G. (2011). Arquitectura de aprendizaje para el manejo de riesgo de falla en ambientes de composición de servicios web. *Avances En Sistemas E Informática; Vol. 6, Núm. 2 (2009); 143-148 Avances En Sistemas E Informática; Vol. 6, Núm. 2 (2009); 143-148 1909-0056 1657-7663*. Retrieved from http://www.bdigital.unal.edu.co/23458/

ARCINIEGAS, J. L., FERNÁNDEZ, V., HORMIGA, A., TULANDE, A., URBANO, F. A., & COLLAZOS, C. A. (2011). Proceso de requerimiento y análisis para la definición de la arquitectura desde la perspectiva de usabilidad para el desarrollo de aplicaciones en la web. *Avances En Sistemas E Informática; Vol. 6, Núm. 2 (2009); 205-210 Avances En Sistemas E Informática; Vol. 6, Núm. 2 (2009); 205-210 1909-0056 1657-7663*. Retrieved from http://www.bdigital.unal.edu.co/23465/

Jaramillo, C. M. Z., & Chaverra, J. J. (2012). An environment based on pre-conceptual schemas for automatically generating source code under the mvc pattern. *Dyna; Vol. 79, Núm. 176 (2012); 56-63 DYNA; Vol. 79, N{ú}m. 176 (2012); 56-63 2346-2183 0012-7353*. Retrieved from http://www.bdigital.unal.edu.co/26919/

Serna, M. D. A., Salazar, J. E. L., & Cortés, J. A. Z. (2011). Arquitectura orientada a servicios en el contexto de la arquitectura empresarial. *Avances En Sistemas E Informática; Vol. 7, Núm. 2 (2010); 74-88 Avances En Sistemas E Informática; Vol. 7, Núm. 2 (2010); 74-88 1909-0056 1657-7663*. Retrieved from http://www.bdigital.unal.edu.co/28772/

YEPEZ, W. L. P., COLLAZOS, C. A., & PENICHET, V. M. R. (2013). Entorno colaborativo de apoyo a la mejora de procesos de software en pequeñas organizaciones de software. *Dyna; Vol. 80, Núm. 177 (2013); 40-48 DYNA; Vol. 80, Núm. 177 (2013); 40-48 2346-2183 0012-7353*. Retrieved from http://www.bdigital.unal.edu.co/29622/

Castilla, W. J. V. (2013). *Herramienta prototipo para generación automática de Servicios Web Semánticos a través del desarrollo de software dirigido por modelos*. Retrieved from http://www.bdigital.unal.edu.co/45825/

de Jesús Martínez Acosta, D. (2014). *Herramienta para la generación automática del código fuente para aplicaciones con arquitectura modelo vista controlador (MVC) bajo desarrollo dirigido por modelos textuales (MDD)*. Retrieved from http://www.bdigital.unal.edu.co/45826/

Arias, F. G. (2012). *El Proyecto de Investigación Introducción a la metodología científica 6a Edición, ISBN:980-07-8529-9, pag. 85*. EDITORIAL EPISTEME, C.A , Caracas - República Bolivariana de Venezuela.

de Barrera, J. H. (2010). *Metodología de la Investigación, guia para la comprensíon holística de la ciencia, ISBN:978-980-6306-66-0, pag.43 , 212*. Quiron Ediciones.

Gurdián-Fernández, A. (2007). *El Paradigma Cualitativo en la Investigación Socio-Educativa, ISBN:978-9968-818-32-2 pag. 152*. Colección : Investigación y Desarrollo Educativo Regional (IDER).

Miguelez, M. M. (2009). *Ciencia y arte de la metodología cualitativa , ISBN:978-968-24-7568-9 pag. 100 , 137.* trillas.

Morales, J. T. (2011). Fenomenología y Hermenéutica Como Epistemología de la Investigación. *Departamento de Filosofía, Facultad de Educación. Universidad de Carabobo*.

Noguero, F. L. (2002). El análisis de contenido como método de investigación. *XXI, Revista de Educación, 4 (2002): 167-179. Universidad de Huelva*.

Roberto Hernández Sampieri, C. F. C. y M. del P. B. L. (2010). *Metodología de la investigación Quinta edición, ISBN:978-607-15-0291-9, pag. 364*. McGRAW-HILL.

Mark Dowd John McDonald, J. S. (2006). *The Art of Software Security Assessment - Identifying and Preventing Software Vulnerabilities ISBN-10: 0-321-44442-6 , ISBN-13: 978-0-321-44442-4*. Addison Wesley Professional.

Daud, M. I. (2010). Secure Software Development Model: A Guide for Secure Software Life Cycle. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol I, IMECS 2010, March 17-19,2010, Hong Kong*.

Hope, P., & White, P. (2007). Software Security Requirements the foundation for security,Cigital Inc. Retrieved from http://www.cigital.com

Criteria, C. (2005). Part 2: Security functional requirements August 2005 Version 2.3. Retrieved from https://www.commoncriteriaportal.org/files/ccfiles/ccpart2v2.3.pdf

Mead, J. H. A. S. J. B. R. J. E. G. M. N. R. (2008). *Software Security Engineering: A Guide for Project Managers. ISBN 10: 032150917X ISBN 13: 9780321509178*. Addison-Wesley Professional.

McGraw, G. R. (2006). *Software Security: Building Security In. ISBN-13: 978-0321356703 - ISBN-10: 0321356705*. Addison-Wesley Professional.

Grady Booch, I. J. y J. R. (2000). *El Proceso Unificado De Desarrollo De Software. ISBN 84-7829-036-2*. The Addison Wesley, Pearson Educación S.A.

Higuera, J. R. B. (2014). *Assessment methodology of web applications automatic security analysis tools for adaptation in the development life cycle*. Universidad Nacional de Educación a Distancia, España.

Goseva-Popstojanova, K., & Perhinschi, A. (2015). On the capability of static code analysis to detect security vulnerabilities. *Information and Software Technology*, *68*, 18–33. https://doi.org/http://dx.doi.org/10.1016/j.infsof.2015.08.002

2014, S. C. M. A. (2014). 2014 SC Awards U.S. Retrieved from https://media.scmagazine.com/documents/64/botn2014sm_15794.pdf

JURGENSOSN, J. L. A. Y. G. (2003). *COMO HACER INVESTIGACION CUALITATIVA: FUNDAMENTOS Y METODOLOGIA - ISBN: 9789688535165*. PAIDOS IBERICA.

Taylor, Steven y Bodgan, R. (1992). *Introducción a los métodos Cualitativos de investigación*. (E. Paidós, Ed.). Barcelona, España: Ediciones Paidós.

Ruiz, R. (2006). *Historia y evolución del pensamiento científico*. México: Juan Carlos Martínez Coll. Retrieved from https://books.google.com.co/books?id=HVs87wEe3ZsC

García Valdés, M., & Suárez Marín, M. (2013). El método Delphi para la consulta a expertos en la investigación científica . *Revista Cubana de Salud Pública* . scielocu .

6 Anexos

ANEXO 1. ENCUESTA

Encuesta dirigida a directores de TI de las instituciones de educación superior del departamento de Norte de Santander.
Proyecto "Estrategia de diseño de aplicaciones web enfocada a prevenir vulnerabilidades de seguridad para el departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta".

Objetivo de la encuesta: Determinar el estado actual de las políticas de seguridad aplicadas a los procesos de desarrollo de software web de las instituciones de educación superior de Norte de Santander.

Fecha de aplicación:	

La presente encuesta Mixta está compuesta por 2 tipos de preguntas: Abiertas y Cerradas, las cuales están agrupadas en las categorías: *Estado actual*, *Autenticación*, *Permisos*, *Base de datos*, *Configuración y Protección de aplicación*. A continuación, encontrará un grupo de preguntas cerradas, donde se solicita marque con una X, la opción que considere pertinente. En algunos casos, encontrará opciones de respuesta abierta, en donde agradecemos complete la información solicitada.

Gracias por su colaboración, sinceridad y valoraciones. Sus respuestas y aportes son de carácter confidencial, solo se presentara al público las estadísticas sin hacer referencia a nombres o instituciones y tendrán un fin netamente académico en el desarrollo del proyecto.

ESTADO ACTUAL

1. ¿La Institución cuenta con políticas para el desarrollo seguro de aplicaciones web?

Si ( )
No ( )

2. ¿La Institución tiene establecida una arquitectura de desarrollo de software web documentada y estandarizada?

Si ( )
No ( )

3. ¿Cuál o cuáles lenguajes de programación utiliza la institución para desarrollar aplicaciones web?

a) PHP ( )
b) JSP ( )
c) ASP ( )

d) Ruby ( )
e) JavaScript ( )
f) Phyton ( )
g) Otros ¿Cuáles? _____

4. ¿Las librerías JavaScript utilizadas en los desarrollos web de la Institución, provienen de fuentes confiables o de sitios oficiales?

Si ( )
No ( )

5. ¿Las aplicaciones web de su Institución, intercambian datos entre ellas por medio de Servicios Web (WS)?

Si ( )
No ( )
No Aplica ( )

6. ¿Los Servicios Web se basan en estándares y protocolos abiertos?

Si ( )
No ( )

7. En caso de ser positiva la respuesta anterior, indique ¿Cuáles son utilizados por las aplicaciones web de su Institución?

a) eXtensible Markup Language - XML ( )
b) Simple Object Access Protocol - SOAP ( )
c) Universal Description Discovery and Integration - UDDI ( )
d) Web Service Description Language - WSDL ( )
e) No aplica ( )
f) Otros ¿Cuáles?
_____

AUTENTICACIÓN.

8. ¿La autenticación de las aplicaciones web, garantiza un correcto funcionamiento en las sesiones y evita la suplantación?

Si ( )
No ( )

9. ¿Se controla el inicio, duración y cierre de la sesión que utiliza el usuario?

Si ( )
No ( )

10. ¿Las aplicaciones web solicita de forma periódica el cambio de la contraseña, comprobando la longitud y seguridad de los datos ingresados por el usuario?

Si ( )
No ( )

11. ¿Las contraseñas se almacenan con un algoritmo diseñado específicamente para la protección por contraseña, como por ejemplo bcrypt, PBKDF2 o scrypt?

Si ( )
No ( )

12. ¿Existe una autenticación mutua entre el cliente que accede a los servicios y el servidor que proporciona el servicio?

Si ( )
No ( )
No aplica ( )

BASE DE DATOS

13. ¿Indique cuáles de los siguientes sistemas de gestión de bases de datos se utilizan para el desarrollo de aplicaciones web?

a) MySQL ( )
b) PostgreSQL ( )
c) DB2 ( )
d) Microsoft SQL Server ( )
e) MariaDB ( )
f) Oracle ( )
g) SQLite ( )
h) Otra. ¿Cuál? ( )

_____

14. ¿La información almacenada en la base de datos, está validada mediante el uso de constraints (restricciones aplicadas a los objetos de una base de datos: default, not null, unique,check) restringiendo de esta forma la entrada para cada campo?

Si ( )
No ( )

15. ¿En el sistema de gestión de base de datos se aplican reglas de integridad (llaves primarias y foráneas)?

Si ( )
No ( )

16. ¿Se cuenta con usuarios distintos en la base de datos para cada desarrollo web, así como sus propios privilegios?

Si ( )
No ( )

17. ¿Las credenciales de acceso a la base de datos se almacenan en alguna parte del desarrollo?

Si ( )
No ( )

PERMISOS

18. ¿Los permisos de acceso de los usuarios a la base de datos, son los adecuados de acuerdo a los privilegios que debería tener?

Si ( )
No ( )

19. ¿Las aplicaciones web, garantizan que el usuario esté autorizado mediante un rol, perfil o algún tipo de comprobación de control de acceso, para interactuar con las funcionalidades de la aplicación o acceder a los datos de la misma?

Si ( )
No ( )

20. ¿Los formularios de las aplicaciones web, tienen establecidos un identificador aleatorio único para cada usuario que inicie sesión? (donde el sistema revisa si el formulario es válido, comparando el identificador generado en la sesión antes de enviar una petición).

Si ( )
No ( )

CONFIGURACIÓN

21. ¿Alguno de los programas que soportan su aplicación web, está desactualizado? (Sistema Operativo, servidor Web/App, motor de Base de Datos, API y todos los componentes de bibliotecas)

Si ( )
No ( )

22. ¿En los servidores se habilitan o instalan funciones innecesarias, como por ejemplo, puertos, servicios, páginas, cuentas, privilegios?

Si ( )
No ( )

23. ¿Se tiene definido los roles de las personas que ingresan con permisos de administrador a los servidores de aplicaciones y bases de datos?

Si ( )
No ( )

24. ¿Se tiene control sobre los permisos establecidos para las carpetas y ficheros de las aplicaciones web?

Si ( )
No ( )

25. ¿Su manejo de errores revela rastros de pila u otros mensajes de error demasiado informativos a los usuarios?

Si ( )
No ( )

26. ¿Cuenta con una arquitectura de aplicación robusta, que proporciona una separación efectiva y segura entre los componentes?

Si ( )
No ( )

27. ¿Se establecen ventanas de mantenimiento periódicas sobre la arquitectura que soporta las aplicaciones web? (para verificar que las configuraciones están correctas, se realice la revisión de logs de seguridad y actualización de software, entre otras).

Si ( )

No ( )

28. ¿Se cuenta con un control de versiones de los aplicativos web, el cual permita identificar los cambios que ha sufrido la aplicación durante el ciclo de vida o que permita volver a una versión anterior en caso de alguna falla?

Si ( )
No ( )

PROTECCIÓN DE APLICACIÓN

29. ¿Las aplicaciones web tienen implementado algún framework o librería anti-XSS que ayude a disminuir los inconvenientes con este tipo de ataques?

Si ( )
No ( )
En caso afirmativo ¿Cuál?_____

30. ¿Las interfaces de acceso público están protegidas mediante encriptación de datos, token de seguridad o algún tipo de comprobación de control sobre las mismas?

Si ( )
No ( )

31. ¿La validación de los datos de entrada (longitud, tipo de dato, rango de valores aceptados, sintaxis) de las aplicaciones, se realizan del lado del servidor así como del cliente?

Si ( )
No ( )

32. ¿Los aplicativos web desarrollados en su Institución, poseen controles para evitar ataques de inyección?

Si ( )
No ( )

33. ¿Se cuenta con algoritmos criptográficos para proteger los datos sensibles de la Institución?

Si ( )
No ( )

34. ¿Los datos considerados sensibles (contraseñas, números de tarjetas de crédito, información personal, pagos registrados en la Institución) tienen algún nivel de protección adicional?

Si ( )
No ( )

35. ¿Los formularios que solicitan el ingreso de datos sensibles, tienen deshabilitado la opción de autocompletado y almacenamiento en cache?

Si ( )
No ( )

36. ¿Las aplicaciones tienen protección de ataques, que permite identificarlos, bloquearlos y proporcionar algún detalle, como por ejemplo, si se está utilizando la aplicación de una forma que un usuario común nunca haría? (tiempo demasiado alto, entradas atípicas, patrones de uso inusuales, peticiones repetidas)

Si ( )
No ( )

37. ¿En caso de un comportamiento atípico, las aplicaciones de su Institución, bloquean solicitudes, direcciones IP o rangos de direcciones IP?

Si ( )
No ( )

38. ¿Las aplicaciones de su Institución, bloquean las cuentas de los usuarios que le dan mal uso a las mismas?

Si ( )
No ( )

COMENTARIOS Y/O SUGERENCIAS:

_____

_____

_____

_____

ANEXO 4. INSTRUMENTO APLICADO INSTITUCIÓN

Universidad de Pamplona
Pamplona - Norte de Santander - Colombia
Teléfonos: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750 -
www.unipamplona.edu.co

Encuesta dirigida a directores de TI de las instituciones de educación superior del departamento de Norte de Santander.
Proyecto "Estrategia de diseño de aplicaciones web enfocada a prevenir vulnerabilidades de seguridad para el departamento de Sistemas de la Universidad Simón Bolívar sede Cúcuta".

Objetivo de la encuesta: Determinar el estado actual de las políticas de seguridad aplicadas a los procesos de desarrollo de software web de las instituciones de educación superior de Norte de Santander.

Fecha de aplicación:	08 de Septiembre de 2017

La presente encuesta Mixta está compuesta por 2 tipos de preguntas: Abiertas y Cerradas, las cuales están agrupadas en las categorías: *Estado actual*, *Autenticación*, *Permisos*, *Base de datos*, *Configuración y Protección de aplicación*. A continuación, encontrará un grupo de preguntas cerradas, donde se solicita marque con una X, la opción que considere pertinente. En algunos casos, encontrará opciones de respuesta abierta, en donde agradecemos complete la información solicitada.

Gracias por su colaboración, sinceridad y valoraciones. Sus respuestas y aportes son de carácter confidencial, solo se presentara al público las estadísticas sin hacer referencia a nombres o instituciones y tendrán un fin netamente académico en el desarrollo del proyecto.

ESTADO ACTUAL

1. ¿La Institución cuenta con políticas para el desarrollo seguro de aplicaciones web?

Si ( )
No (X)

2. ¿La Institución tiene establecida una arquitectura de desarrollo de software web documentada y estandarizada?

Si (X)
No ( )

3. ¿Cuál o cuáles lenguajes de programación utiliza la institución para desarrollar aplicaciones web?

a) PHP (X)
b) JSP ( )
c) ASP ( )

d) Ruby ( )
e) JavaScript ( )
f) Phyton ( )
g) Otros ¿Cuáles? _____

4. ¿Las librerías JavaScript utilizadas en los desarrollos web de la Institución, provienen de fuentes confiables o de sitios oficiales?

Si (X)
No ( )

5. ¿Las aplicaciones web de su Institución, intercambian datos entre ellas por medio de Servicios Web (WS)?

Si (X)
No ( )
No Aplica ( )

6. ¿Los Servicios Web se basan en estándares y protocolos abiertos?

Si (X)
No ( )

7. En caso de ser positiva la respuesta anterior, indique ¿Cuáles son utilizados por las aplicaciones web de su Institución?

a) eXtensible Markup Language - XML ( )
b) Simple Object Access Protocol - SOAP (X)
c) Universal Description Discovery and Integration - UDDI ( )
d) Web Service Description Language - WSDL (X)
e) No aplica ( )
f) Otros ¿Cuáles?

_____

AUTENTICACIÓN.

8. ¿La autenticación de las aplicaciones web, garantiza un correcto funcionamiento en las sesiones y evita la suplantación?

Si (X)
No ( )

9. ¿Se controla el inicio, duración y cierre de la sesión que utiliza el usuario?

Si (X)
No ( )

10. ¿Las aplicaciones web solicita de forma periódica el cambio de la contraseña, comprobando la longitud y seguridad de los datos ingresados por el usuario?

Si ( )
No (X)

11. ¿Las contraseñas se almacenan con un algoritmo diseñado específicamente para la protección por contraseña, como por ejemplo bcrypt, PBKDF2 o scrypt?

Si ( )
No (X)

12. ¿Existe una autenticación mutua entre el cliente que accede a los servicios y el servidor que proporciona el servicio?

Si (X)
No ( )
No aplica ( )

BASE DE DATOS

13. ¿Indique cuáles de los siguientes sistemas de gestión de bases de datos se utilizan para el desarrollo de aplicaciones web?

a) MySQL (X)
b) PostgreSQL ( )
c) DB2 (X)
d) Microsoft SQL Server ( )
e) MariaDB ( )
f) Oracle ( )
g) SQLite ( )
h) Otra. ¿Cuál? ( )

---

14. ¿La información almacenada en la base de datos, está validada mediante el uso de constraints (restricciones aplicadas a los objetos de una base de datos: default, not null, unique,check) restringiendo de esta forma la entrada para cada campo?

Si (X)
No ( )

15. ¿En el sistema de gestión de base de datos se aplican reglas de integridad (llaves primarias y foráneas)?

Si (X)
No ( )

16. ¿Se cuenta con usuarios distintos en la base de datos para cada desarrollo web, así como sus propios privilegios?

Si (X)
No ( )

17. ¿Las credenciales de acceso a la base de datos se almacenan en alguna parte del desarrollo?

Si ( )
No (X)

PERMISOS

18. ¿Los permisos de acceso de los usuarios a la base de datos, son los adecuados de acuerdo a los privilegios que debería tener?

Si (X)
No ( )

19. ¿Las aplicaciones web, garantizan que el usuario esté autorizado mediante un rol, perfil o algún tipo de comprobación de control de acceso, para interactuar con las funcionalidades de la aplicación o acceder a los datos de la misma?

Si (X)
No ( )

20. ¿Los formularios de las aplicaciones web, tienen establecidos un identificador aleatorio único para cada usuario que inicie sesión? (donde el sistema revisa si el formulario es válido, comparando el identificador generado en la sesión antes de enviar una petición).

Si ( )
No (X)

CONFIGURACIÓN

21. ¿Alguno de los programas que soportan su aplicación web, está desactualizado? (Sistema Operativo, servidor Web/App, motor de Base de Datos, API y todos los componentes de bibliotecas)

Si (X)
No ( )

22. ¿En los servidores se habilitan o instalan funciones innecesarias, como por ejemplo, puertos, servicios, páginas, cuentas, privilegios?

Si ( )
No (X)

23. ¿Se tiene definido los roles de las personas que ingresan con permisos de administrador a los servidores de aplicaciones y bases de datos?

Si (X)
No ( )

24. ¿Se tiene control sobre los permisos establecidos para las carpetas y ficheros de las aplicaciones web?

Si ( )
No (X)

25. ¿Su manejo de errores revela rastros de pila u otros mensajes de error demasiado informativos a los usuarios?

Si (X)
No ( )

26. ¿Cuenta con una arquitectura de aplicación robusta, que proporciona una separación efectiva y segura entre los componentes?

Si (X)
No ( )

27. ¿Se establecen ventanas de mantenimiento periódicas sobre la arquitectura que soporta las aplicaciones web? (para verificar que las configuraciones están correctas, se realice la revisión de logs de seguridad y actualización de software, entre otras).

Si ( )
No (X)

28. ¿Se cuenta con un control de versiones de los aplicativos web, el cual permita identificar los cambios que ha sufrido la aplicación durante el ciclo de vida o que permita volver a una versión anterior en caso de alguna falla?

Si ( )
No (X)

PROTECCIÓN DE APLICACIÓN

29. ¿Las aplicaciones web tienen implementado algún framework o librería anti-XSS que ayude a disminuir los inconvenientes con este tipo de ataques?

Si ( )
No (X)
En caso afirmativo ¿Cuál?_____

30. ¿Las interfaces de acceso público están protegidas mediante encriptación de datos, token de seguridad o algún tipo de comprobación de control sobre las mismas?

Si ( )
No (X)

31. ¿La validación de los datos de entrada (longitud, tipo de dato, rango de valores aceptados, sintaxis) de las aplicaciones, se realizan del lado del servidor así como del cliente?

Si (X)
No ( )

32. ¿Los aplicativos web desarrollados en su Institución, poseen controles para evitar ataques de inyección?

Si (X)
No ( )

33. ¿Se cuenta con algoritmos criptográficos para proteger los datos sensibles de la Institución?

Si ( )
No (X)

34. ¿Los datos considerados sensibles (contraseñas, números de tarjetas de crédito, información personal, pagos registrados en la Institución) tienen algún nivel de protección adicional?

Si ( )
No (X)

35. ¿Los formularios que solicitan el ingreso de datos sensibles, tienen deshabilitado la opción de autocompletado y almacenamiento en cache?

Si ( )
No (X)

36. ¿Las aplicaciones tienen protección de ataques, que permite identificarlos, bloquearlos y proporcionar algún detalle, como por ejemplo, si se está utilizando la aplicación de una forma que un usuario común nunca haría? (tiempo demasiado alto, entradas atípicas, patrones de uso inusuales, peticiones repetidas)

Si ( )
No (X)

37. ¿En caso de un comportamiento atípico, las aplicaciones de su Institución, bloquean solicitudes, direcciones IP o rangos de direcciones IP?

Si ( )
No (X)

38. ¿Las aplicaciones de su Institución, bloquean las cuentas de los usuarios que le dan mal uso a las mismas?

Si ( )
No (X)

COMENTARIOS Y/O SUGERENCIAS:

Despues de la aplicación de la encuesta se evidencia que hay falencias y elementos por fortalecer a nivel de seguridad.

ANEXO 5. VALIDACIÓN

Formulario de evaluación de la política de desarrollo de software web seguro de la Universidad Simón Bolívar sede Cúcuta aplicando método Delphi											
Indicador	**Calificación**										**Observación**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Indicador 1: ¿Es importante establecer las responsabilidades de los involucrados en el desarrollo de software web seguro?										X	
Indicador 2: ¿La definición de requerimientos de seguridad es un aspecto importante para el desarrollo de software web seguro?										X	
Indicador 3: ¿La participación de usuarios finales es importante para el proceso de construcción de controles de seguridad del aplicativo?										X	
Indicador 4: ¿La forma como se asignan los nombres de usuario en una aplicación web es importante para el control de seguridad de la misma?								X			Que el usuario pueda cambiar el nombre puede ayudar mas a la seguridad
Indicador 5: ¿Las contraseñas seguras deben incluir parámetros de seguridad adicionales como caracteres especiales, así como la opción de restablecimiento y cambio?										X	
Indicador 6: ¿Es importante establecer si se detecta inactividad o inicio de sesión en diferentes ubicaciones?										X	
Indicador 7: ¿El uso de roles y perfiles es importante para controlar el acceso a las funcionalidades de la aplicación web por parte de los usuarios?										X	
Indicador 8: ¿Cada usuario del sistema debe contar con credenciales criptográficas únicas?										X	
Indicador 9: ¿las sesiones deben contar con mecanismos para que finalicen ya sea de forma automática o por petición del usuario?										X	
Indicador 10: ¿Los datos de entrada de las aplicaciones web se deben manipular de forma adecuada (tipos de datos, longitud) para garantizar de esta forma su integridad y seguridad?										X	

Indicador 11: ¿Es importante proteger la aplicaciones web cuando se presentan fallos haciendo buen manejo de los errores generados y almacenando logs y auditorias de los mismos?										X	
Indicador 12: ¿El uso de recursos de terceros como lo son las librerías o bibliotecas debe ser de fuentes oficiales o confiables para así garantizar que no vamos a tener código corrupto en nuestros aplicativos web?										X	
Indicador 13: ¿Las credenciales de acceso de la base de datos y el nivel de privilegios de la misma deben ser los adecuados para el usuario con el que se realiza la conexión?										X	
Indicador 14: ¿Las tablas de la base de datos deben contar con reglas de integridad?										X	
Indicador 15: ¿Los servicios web deben tener implementada seguridad de extremo a extremo y auditorias para identificar las trazas de los mensajes intercambiados?										X	
Indicador 16: ¿Una buena configuración de la plataforma operativa ayuda a proteger las aplicaciones web si se establecen controles como *permisos sobre las carpetas* y archivos, se mantiene actualizado el software y se aplican mantenimientos periódicos?										X	

Ciudad y fecha de presentación: 15-01-2018, cuate.

Nombre del Experto: Cleiver Andrade Rossi

Firma

Formulario de evaluación de la política de desarrollo de software web seguro de la Universidad Simón Bolívar sede Cúcuta aplicando método Delphi											
Indicador	**Calificación**										**Observación**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Indicador 1: ¿Es importante establecer las responsabilidades de los involucrados en el desarrollo de software web seguro?									X		
Indicador 2: ¿La definición de requerimientos de seguridad es un aspecto importante para el desarrollo de software web seguro?									X		
Indicador 3: ¿La participación de usuarios finales es importante para el proceso de construcción de controles de seguridad del aplicativo?										X	**Los usuarios finales son los mas propensos a sufrir problemas de seguridad, su participación el en el proceso es sumamente importante**
Indicador 4: ¿La forma como se asignan los nombres de usuario en una aplicación web es importante para el control de seguridad de la misma?					X						**Depende de si el nombre mismo de usuario revela información que va mas allá de la credencial de acceso. Lo realmente importante es que el usuario conozca los riesgos de seguridad del software**
Indicador 5: ¿Las contraseñas seguras deben incluir parámetros de seguridad adicionales como caracteres especiales, así como la opción de restablecimiento y cambio?										X	**El riesgo que representa el nivel de seguridad de las contraseñas de los usuarios es importante, pero se debe controlar, de manera que estas no sean imposibles de recordar .**
Indicador 6: ¿Es importante establecer si se detecta inactividad o inicio de sesión en diferentes ubicaciones?									X		
Indicador 7: ¿El uso de roles y perfiles es importante para controlar el acceso a las funcionalidades de la aplicación web por parte de los usuarios?										X	**Especialmente en sistemas robustos, donde se tienen gran cantidad de funcionalidades y diferentes niveles de acceso por parte de los usuarios**
Indicador 8: ¿Cada usuario del sistema debe contar con credenciales criptográficas únicas?										X	
Indicador 9: ¿las sesiones deben contar con mecanismos para que finalicen ya sea de forma automática o por petición del usuario?									X		**Estas deben finalizar de forma automática siempre y cuando no se detecte actividad en un tiempo determinado, de lo contrario, puede afectar el trabajo desarrollado por el usuario**
Indicador 10: ¿Los datos de entrada de las aplicaciones web se deben manipular de forma adecuada (tipos de datos,										X	

longitud) para garantizar de esta forma su integridad y seguridad?											
Indicador 11: ¿Es importante proteger la aplicaciones web cuando se presentan fallos haciendo buen manejo de los errores generados y almacenando logs y auditorias de los mismos?									X		**Los registros del logs permiten identificar los fallos mas recurrentes que se presentan en las aplicaciones y definir mecanismos de solución de los mismos.**
Indicador 12: ¿El uso de recursos de terceros como lo son las librerías o bibliotecas debe ser de fuentes oficiales o confiables para así garantizar que no vamos a tener código corrupto en nuestros aplicativos web?										X	**No solo la presencia de código corrupto, también es muy frecuente que estos recursos no reciban actualizaciones o sean descontinuados, cuando no son respaldados por comunidades de usuarios.**
Indicador 13: ¿Las credenciales de acceso de la base de datos y el nivel de privilegios de la misma deben ser los adecuados para el usuario con el que se realiza la conexión?									X		
Indicador 14: ¿Las tablas de la base de datos deben contar con reglas de integridad?									X		
Indicador 15: ¿Los servicios web deben tener implementada seguridad de extremo a extremo y auditorias para identificar las trazas de los mensajes intercambiados?										X	
Indicador 16: ¿Una buena configuración de la plataforma operativa ayuda a proteger las aplicaciones web si se establecen controles como permisos sobre las carpetas y archivos, se mantiene actualizado el software y se aplican mantenimientos periódicos?									X		

Ciudad y fecha de presentación: Bogota DC 19 enero 2018

Nombre del Experto: Miguel Eduardo Ropero Silva

Firma

Formulario de evaluación de la política de desarrollo de software web seguro de la Universidad Simón Bolívar sede Cúcuta aplicando método Delphi											
Indicador	**Calificación**										**Observación**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Indicador 1: ¿Es importante establecer las responsabilidades de los involucrados en el desarrollo de software web seguro?										X	
Indicador 2: ¿La definición de requerimientos de seguridad es un aspecto importante para el desarrollo de software web seguro?										X	
Indicador 3: ¿La participación de usuarios finales es importante para el proceso de construcción de controles de seguridad del aplicativo?										X	
Indicador 4: ¿La forma como se asignan los nombres de usuario en una aplicación web es importante para el control de seguridad de la misma?								X			Muy importante definir políticas de creación de usuarios así como la analaun o deshabilitar los mismos
Indicador 5: ¿Las contraseñas seguras deben incluir parámetros de seguridad adicionales como caracteres especiales, así como la opción de restablecimiento y cambio?										X	
Indicador 6: ¿Es importante establecer si se detecta inactividad o inicio de sesión en diferentes ubicaciones?										X	
Indicador 7: ¿El uso de roles y perfiles es importante para controlar el acceso a las funcionalidades de la aplicación web por parte de los usuarios?										X	
Indicador 8: ¿Cada usuario del sistema debe contar con credenciales criptográficas únicas?										X	
Indicador 9: ¿las sesiones deben contar con mecanismos para que finalicen ya sea de forma automática o por petición del usuario?										X	
Indicador 10: ¿Los datos de entrada de las aplicaciones web se deben manipular de forma adecuada (tipos de datos, longitud) para garantizar de esta forma su integridad y seguridad?										X	

Indicador 11: ¿Es importante proteger la aplicaciones web cuando se presentan fallos haciendo buen manejo de los errores generados y almacenando logs y auditorias de los mismos?										X	tener registro de logs así como de las soluciones que se realicen.
Indicador 12: ¿El uso de recursos de terceros como lo son las librerías o bibliotecas debe ser de fuentes oficiales o confiables para así garantizar que no vamos a tener código corrupto en nuestros aplicativos web?										X	realizar seguimiento de las librerias o bibliotecas. para evitar errores o vulnerabilidades. que se presenten.
Indicador 13: ¿Las credenciales de acceso de la base de datos y el nivel de privilegios de la misma deben ser los adecuados para el usuario con el que se realiza la conexión?										X	
Indicador 14: ¿Las tablas de la base de datos deben contar con reglas de integridad?										X	
Indicador 15: ¿Los servicios web deben tener implementada seguridad de extremo a extremo y auditorias para identificar las trazas de los mensajes intercambiados?										X	
Indicador 16: ¿Una buena configuración de la plataforma operativa ayuda a proteger las aplicaciones web si se establecen controles como permisos sobre las carpetas y archivos, se mantiene actualizado el software y se aplican mantenimientos periódicos?										X	

Ciudad y fecha de presentación: Cucuta 11 de enero 2018

Nombre del Experto: Ramiro Alfonso Gomez C.

Ramiro A. Gomez C.

Firma

ANEXO 6. ARTÍCULO

REVISTA INGENIO UFPSO
Volumen 13 / Año 2017 / ISSN 2011-64

ASPECTOS DE SEGURIDAD EN EL DESARROLLO DE APLICACIONES WEB

José Darío Rodríguez Leal, Sandra Milena Vargas Angulo
Departamento de Sistemas
Universidad Simón Bolívar Sede Cúcuta
Cúcuta Colombia
{j.rodriguezl, s.vargas01}@unisimonbolivar.edu.co

Luis Alberto Esteban Villamizar
CICOM - Grupo de Investigación De Ciencias Computacionales
Universidad de Pamplona
Pamplona Colombia
lesteban@unipamplona.edu.co

Carlos René Angarita Sanguino
GIDIS - Grupo de Investigación de Ingeniería de Software
Universidad Francisco de Paula Santander
Cúcuta Colombia
carlosreneas@ufps.edu.co

Fecha de recepción:
Fecha de aprobación:

Resumen: El presente artículo contextualiza las vulnerabilidades que presentan las aplicaciones web, el ciclo de vida del desarrollo de software seguro y los diferentes ataques que se pueden presentar, basados en información de entidades que a nivel mundial tienen un compromiso con la seguridad y se dedican a documentar los principales fallos que se presentan. En base a lo anterior se pueden establecer estándares o políticas de desarrollo seguro que permitan mitigar el riesgo de acceso no autorizado o la pérdida de la información que contienen los aplicativos web.

Palabras Clave: Arquitectura segura de software, Ataques informáticos, Desarrollo de Software Web, Seguridad informática, Vulnerabilidad de software.

Abstract: This article contextualizes the vulnerabilities presented by web applications, the lifecycle of the development of secure software and the different attacks which may occur, based on information from entities that have a global security commitment and are dedicated to documenting the main failures presented. Based on the above, it is possible establish secure development standards or policies that allow to mitigate the risk of unauthorized access or loss of information contained in web applications.

Keywords: Software Secure Architecture, Computer Attacks, Web software development, IT Security, Software Vulnerabilities.

REVISTA INGENIO UFPSO
Volumen 13 / Año 2017 / ISSN 2011-64

INTRODUCCIÓN

Internet y la Web han influido en el mundo de la informática y en la sociedad en general, rompiendo barreras físicas, económicas y lógicas, abriendo un abanico de nuevas posibilidades con un gran mercado a nivel mundial y con ello una gran migración de sistemas de escritorio a plataformas Web.

Las múltiples ventajas que el desarrollo Web presenta, son distribuidas, son multiplataforma, su mantenimiento es centralizado, diversas tecnologías, tienen estándares claros y definidos, entre otras, han obligado a que el desarrollo en ambientes de escritorio evolucione.

Para garantizar la disponibilidad de las aplicaciones Web, se debe tener claro las condiciones de rendimiento y cuidado en temas de seguridad, debido a que su carácter distribuido, aumenta el número usuarios que acceden y los posibles ataques se incrementan.

Conocer las principales vulnerabilidades de seguridad presentes en una aplicación web en sus distintos niveles y establecer buenas prácticas de desarrollo seguro pueden mitigar el riesgo y garantizar la disponibilidad de la información.

1. APLICACIÓN WEB

Las aplicaciones web son desarrollos de software que permiten la generación automática de contenido, el desarrollo de comercio electrónico y la integración con la banca o sistemas de pago. Además, permite la gestión de clientes, contabilidad o inventario todo esto a través de un navegador web, el cual actúa como cliente y realiza peticiones a un servidor que es el encargado de ejecutar procesos y dar la respuesta correspondiente. Lógicamente suelen distinguirse tres niveles. El nivel superior que interacciona con el usuario (Navegador web), el nivel inferior de datos (la base de datos) y el nivel intermedio que procesa los datos (el servidor web). Esta arquitectura permite la separación de funciones en la lógica de presentación, negocio y datos. (LujanWeb, 2002)

2. PROBLEMAS DE SEGURIDAD

Un ataque es cualquier acto malicioso contra un sistema, tratando de explotar sus vulnerabilidades, que en las aplicaciones web se deben a datos maliciosos, scripts o códigos que tienen el objetivo de robar datos sensibles o acceder de forma no autorizada. Estos ataques se pueden materializar de muchas formas dependiendo de la naturaleza de la vulnerabilidad. Estos ataques pueden basarse en defectos que se presentan a lo largo de cualquier etapa del ciclo de vida del desarrollo de software. (Higuera, 2014)

Los ataques pueden ser de varios tipos (CheswickW.R, 2003), robo de contraseñas, Ingeniería social, vulnerabilidades de software y puertas traseras, fallos de autenticación o en el protocolo, ataques de denegación de servicio, botnets entre otros.

Existen varios factores involucrados para que una amenaza se materialice en una

aplicación, agente o amenaza, la persona que hace el ataque, vectores de ataque, utilizados para llevar a cabo el ataque, vulnerabilidad, ausencia de controles, impacto en los activos de TI o impacto en los negocios.

El desarrollo de web seguras se enfoca en resolver tres principales problemas, primero la autenticación, segundo la capacidad que puede tener un atacante para engañar a los usuarios, una vez que se han autenticado y por último el riesgo que supone alojar el contenido generado por un usuario en un sitio web. (Neville-Neil, 2007)

Los desarrolladores deben utilizar alguna guía o patrón para comprobar la seguridad, antes que sus aplicaciones salgan a producción. Algunos aspectos referentes a la seguridad deben ser tomados en cuenta, para evitar errores específicos de los aplicativos.

Algunos patrones deben ser tomados en cuenta al momento de asegurar tu aplicación, propiedad de los recursos (quien tiene control de los recursos), objetos públicos, autenticación de usuarios, perfiles de usuario, control de administradores, entre otros. (Near, 2016)

Generalmente las vulnerabilidades se deben a defectos en la arquitectura seleccionada y a especificaciones del software. Se pueden abordar en la fase de análisis de requisitos o en las especificaciones de la fase de diseño. Las vulnerabilidades de implementación hacen referencia a los errores de seguridad que comenten los desarrolladores al construir módulos con las especificaciones propuestas. Las operacionales se refieren a los defectos de seguridad que surgen en el despliegue y la configuración del sistema desarrollado en un entorno particular. (MarkDowd, 2006)

Hay varios aspectos relativos a la seguridad que deben tenerse en cuenta al momento de implementar una aplicación web, gestión de excepciones al manejar errores, validación de datos de entrada y salida, gestión de sesiones de usuarios, auditoria y registro de eventos relacionados con la seguridad.

Inyección de código SQL

En esta vulnerabilidad los atacantes tratan de obtener acceso no autorizado a la base de datos, a través del uso de sentencias SQL, o comandos no deseados que cambian o interceptan los datos. Entre los tipos de ataques SQL podemos nombrar los siguientes, regulación del cliente, perjudicar entradas, privilegios generosos sobre los objetos de la base de datos, tamaño variable no controlado que permite el ingreso de sentencias en las variables o desbordamiento de buffer. (Mavromoustakos, 2016)

Tautologías. Son ataques que inyectan código en una o más sentencias condicionales que siempre se evalúan como verdaderas, se utilizan para omitir la página de autenticación y extraer los datos, explotando el campo WHERE. (Zhang, 2011)

SELECT cedula FROM persona WHERE nombre=’ ’ OR 1=1

El código inyectado en el condicional (OR 1 = 1) cambia la cláusula WHERE en una tautología, la consulta recupera todas las filas. (Antunes, 2009)

Consultas ilegales / incorrectamente lógicas. Este ataque publica la estructura de back-end de la base de datos a los atacantes o interceptores. La página de error mostrada por los servidores de aplicaciones, ayuda a los atacantes a obtener más información, que puede ser mal utilizada. (Weiss, 2012)

Consultas Piggy-backed. En este ataque se inyecta una consulta adicional en la consulta original, tratando de incluir consultas nuevas para obtener distintos resultados. Si el ataque es exitoso, se puede insertar virtualmente cualquier tipo de comando SQL. (Karam, 2009)

Union Query. El atacante escribe consultas SQL en tablas cruzadas. El resultado devuelve un conjunto de datos que es la unión de los resultados de la consulta original con los resultados de la consulta inyectada.

Servicios Web inseguros

Actualmente se utilizan Servicios Web (WS) como una solución para la integración entre aplicaciones, permitiendo interoperabilidad y extensibilidad entre las mismas. Lo anterior obliga construirlos con estándares que garanticen la seguridad del flujo de datos entre las diferentes aplicaciones que los van a utilizar. (Gutierrez, 2005)

Al consumir un servicio web se debe garantizar la autenticación entre sus interlocutores, ya sea un usuario y contraseña, un token de seguridad, un certificado o firma digital. Igualmente se debe garantizar que al interactuar varias veces mantenga la autenticación. (Token, 2006)

La autorización sobre un WS determinar quién y que se puede hacer sobre cada recurso. La autorización concede los permisos de ejecución de ciertos tipos de operaciones o recursos a los usuarios que estén autenticados. Otro aspecto es la confidencialidad que define la manera de ocultar la información crítica intercambiada. La confidencialidad se consigue mediante técnicas de cifrado. (C.Gutierrez, 2005)

Para el caso de los WS es necesario demostrar que un cliente utilizo un servicio pese a que este lo niegue (no repudio del solicitante), así como demostrar que un servicio fue ejecutado (no repudio del receptor). Es por esto que los servicios web deben implementar mecanismos para auditar las acciones que se ejecutan con ellos.

Es necesario disponer de facilidades que permitan establecer un contexto único de seguridad y de extremo a extremo.

Para establecer un único contexto de seguridad, se deben aplicar operaciones criptográficas sobre la información que no dependa de la seguridad configurada por debajo de la capa de aplicación (por ejemplo a nivel IP mediante VPN) garantizando los servicios de seguridad básicos. (Saltzer, 1984)

3. DESARROLLO DE SOFTWARE SEGURO

El ciclo de vida del desarrollo de software seguro (SSDLC) es la base para desarrollar aplicaciones web seguras. Este método utiliza diferentes procesos humanos y tecnológicos, como métodos de requisitos de seguridad, métodos de análisis de riesgos, listas de verificación de seguridad, herramientas de análisis de seguridad entre otras para reducir al mínimo las vulnerabilidades de seguridad.

Los controles de seguridad garantizan que la aplicación funcione de una manera deseada y proporciona defensa contra amenazas de seguridad. En la práctica, la seguridad pasa desapercibida en las primeras fases del ciclo de vida de software (SLC) por lo tanto una fase transfiere sus

vulnerabilidades a la otra fase. (Daud, 2010)

3.1 ANÁLISIS DE REQUISITOS DE SEGURIDAD

Los requisitos de seguridad se clasifican en funcionales que enumeran las funciones que un sistema debe realizar, los no funcionales que enumeran las propiedades que un sistema debe poseer y los que se derivan de requisitos de seguridad funcionales. (Hope, 2007)

Los requisitos de seguridad de software se pueden tratar en base a las historias de usuario. Las historias de usuario son una forma efectiva de derivar los requerimientos de los usuarios de manera eficiente. Es importante anticipar un comportamiento anormal en una aplicación web para que sea segura y fiable. Es por esto que se deben crear casos de uso para mitigar aquellos comportamientos anormales. (Criteri, 2005)

Antes de definir los requisitos de seguridad se deben identificar las partes de la aplicación que requieren seguridad. Estas partes de la aplicación se denominan Objetivos de Evaluación (ODE). Una vez se identifica ODE, se pueden encontrar de forma sencilla los Requisitos Funcionales de Seguridad (RFS). En la tabla 1 enumera los RFS y su actividad.

Clase	RFS	Descripción	ODE		
			Comunicación con el servidor	Certificado digital	Autenticación
FCO: Comunicación	FCO_NRO	No repudio de origen	Si	Si	
	FCO_NRR	No repudio de recepción	Si	Si	
FSC: Soporte criptográfico	FSC_GCC	Gestión de claves criptográficas			Si
	FSC_OC	Operación criptográfica	Si	Si	

Tabla 1 RFS y sus actividades

3.2 DISEÑO E IMPLEMENTACIÓN SEGURA

La fase de diseño es la encargada de darle forma a los requisitos. Esta fase juega un papel muy importante cuando se da el diseño de los requisitos de seguridad. Es importante diseñar un modelo de amenaza de aplicaciones seguras, este modelado es una técnica para identificar amenazas, vulnerabilidades y sus contramedidas.

Figura 1 Ciclo de vida desarrollo de software seguro

Una vez que tenemos los requisitos de seguridad y tenemos los diagramas de flujo debemos identificar los puntos desde donde el atacante puede entrar al sistema, al identificar dichos puntos se van a poder observar las posibles amenazas que un atacante puede explotar desde estos puntos. Los arboles de ataque pueden ser trazados para aclarar la compresión de la metodología del atacante. (Mead; 2008)

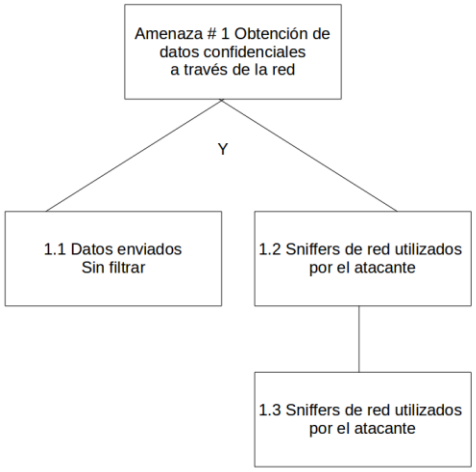

Figura 2 Árbol de ataque

El análisis de las vulnerabilidades es una parte fundamental del modelado de

amenazas. En la tabla 3 se puede observar algunas áreas relacionadas donde se pueden presentar vulnerabilidades, estas vulnerabilidades pueden ocurrir en cualquier fase del ciclo de vida del software, pero es importante identificarlas en la fase de diseño.

Área de vulnerabilidad	Tipo de vulnerabilidad
Sistema Operativo (SO)	Desbordamiento de bufer, punteros nulos, bloqueo de recursos, excepciones, etc.
Comunicación	No repudio de origen. No repudio de recepción.
Datos de base de datos / usuario	Tipos de datos no validos, inyección SQL, Cross Site Scripting, Rollback, integridad de datos, etc.
Criptografía	Gestión de claves, operación criptográfica, etc.
Control de acceso	Política de control de acceso, autenticación de datos, política control de flujo de información, etc.
Privacidad	Privilegios, anonimato, pseudoanonimato, etc.
Programación	Excepciones

Tabla 2 Vulnerabilidades comunes

3.3 PRUEBAS DE SEGURIDAD Y DESPLIEGUE

Las pruebas de seguridad son de vital importancia ya que por media de ellas podemos identificar las fallas de seguridad antes de la liberación de la aplicación. En este punto el desarrollador debe pensar como un atacante y tratar de lanzar una serie de ataques los cuales le permitirán identificar errores en el sistema. Para comprobar que el software cumple con los requisitos de seguridad debemos aplicar 1) Pruebas Funcionales y 2) Pruebas basadas en riesgos. Las pruebas funcionales tratan de probar la aplicación de software con requisitos funcionales, estos requisitos definen el comportamiento funcional de la aplicación para un estado específico. Las pruebas basadas en riesgos se enfocan en los estados o comportamientos que no debe presentar una aplicación. Durante la aplicación de esta prueba se crean planes de prueba para componentes específicos de software que requieren seguridad. Una vez que se tenga toda la información sobre amenazas de seguridad, vulnerabilidades, y las acciones para enfrentarlas, entonces se puede proceder a aplicar este tipo de pruebas. (McGraw, 2006)

Pruebas dirigidas	Pruebas realizadas por el equipo de pruebas de TI y el equipo de pruebas de penetración.
Pruebas externas	Pruebas realizadas en servidores externos y firewalls
Pruebas internas	Pruebas para verificar las amenazas internas por el usuario autorizado.
Prueba ciega	Se examinan todas las acciones y procedimientos que un atacante real puede realizar.
Prueba de doble ciego	Pruebas ciegas realizadas por desarrolladores que los demás integrantes del equipo desconocen.

Tabla 3 Tipos de pruebas

4. COMPROMISO CON LA SEGURIDAD

Existen organizaciones a nivel mundial que están comprometidas en aumentar la conciencia sobre la seguridad de las aplicaciones web al identificar algunos de los riesgos más críticos que enfrentan las empresas o instituciones. Entre las más importantes e influyentes podemos nombrar a OWASP, SANS y WASC. Podemos decir que gracias a su trabajo y empeño nos muestran el camino para cumplir con estándares de seguridad en el desarrollo de software web para así enfrentar las amenazas que se presentan hoy en día en la red.

OWASP - Open Web Application Security Project

Es un proyecto de código abierto que se dedica a documentar y combatir las causas que hacen que un software sea inseguro. La comunidad OWASP está formada por empresas, organizaciones educativas y particulares de todo mundo. Juntos constituyen una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías que se liberan y pueden ser usadas gratuitamente por cualquiera.

Entre sus proyectos más importantes se encuentra OWASP Top 10 de Riesgos de seguridad de aplicaciones en el cual documenta los diez riesgos de seguridad más importantes en aplicaciones web. Esta lista se publica y actualiza cada tres años por dicha organización. El objetivo de este proyecto según la OWASP top 10, es crear

conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones. Para la versión del año 2017 se presentan los siguientes riesgos: (owasp, 2017)

*A1 – Inyección. S*e producen cuando se envían datos no confiables a un intérprete como parte de un comando o consulta.

A2 – Autenticación rota y gestión de sesión. Mala implementación en la gestión de sesiones permitiendo robo de identidad.

A3 – Cross-Site Scripting (XSS). Inyección código malicioso en las aplicaciones web permitiendo secuestrar sesiones, deshacerse de los sitios web o redirigir al usuario a sitios maliciosos.

A4 – Control de Acceso Roto. Mala implementación de restricciones a los usuarios permitiendo que accedan a páginas no autorizadas.

A5 – Configuración errónea de seguridad. Mala configuración de los sistemas que soportan la aplicación, software desactualizado.

A6 – Exposición de datos sensibles. Los datos sensibles merecen una protección adicional así como precauciones especiales cuando se intercambian con el navegador.

A7 – Protección de ataque insuficiente. La protección contra ataques debe ir más allá de la validación de entrada básica e implica la detección automática, registro, respuesta e incluso bloqueo de intentos de explotación.

A8 – Falsificación de Solicitudes entre Sitios (CSRF). Permite al atacante forzar al navegador de la víctima a generar peticiones que la aplicación vulnerable cree que son peticiones legítimas de la víctima.

A9 – Uso de componentes con vulnerabilidades conocidas. Las aplicaciones y las API que utilizan componentes con vulnerabilidades conocidas pueden socavar las defensas de las aplicaciones y permitir varios ataques e impactos.

A10 – APIs desprotegidas. Las aplicaciones modernas a menudo implican aplicaciones ricas de cliente y API, como JavaScript en el navegador y aplicaciones móviles, que se conectan a una API de algún tipo (SOAP / XML, REST / JSON, RPC, GWT, etc.).

SANS - SysAdmin Audit, Networking and Security Institute

Es una institución líder a nivel mundial en capacitación en seguridad de la información en el mundo la cual tiene un proyecto llamado CWE/SANS Top 25 que se encarga de clasificar vulnerabilidades de Aplicaciones, en términos de su importancia relacionada con la frecuencia con la que se producen en las aplicaciones y el peligro de los ataques que pueden desencadenar Common Weakness Enumeration (CWE) es un diccionario de las debilidades de software comunes que pueden ocurrir en la arquitectura, diseño, código o implementación del software que puede conducir a vulnerabilidades de seguridad explotables. CWE fue creado para:

- Servir como un lenguaje común para describir las debilidades de seguridad del software.
- Servir como un estándar de medición para herramientas de seguridad de software dirigidas a estas debilidades.
- Proporcionar un estándar común de referencia para la identificación de

> debilidades, mitigación y esfuerzos de prevención. Las debilidades del software son fallas, errores, vulnerabilidades y otros errores en la implementación del software, el código, el diseño o la arquitectura que, si no se resuelven, podrían resultar en sistemas y redes vulnerables al ataque.

Las debilidades del software podrían ser por ejemplo: desbordamientos de búfer, problemas de estructura y validez; Manipulación de elementos especiales comunes; Errores de canal y ruta; Errores de la interfaz de usuario; Trayectoria y errores de equivalencia; Errores de autenticación; Errores de gestión de recursos; Verificación insuficiente de los datos; Evaluación e inyección de código; Y aleatoriedad y previsibilidad. (Systems, 2015)

WASC - Web Application Security Consortium

El Consorcio de Seguridad de Aplicaciones Web (WASC) es una organización sin fines de lucro compuesto por un grupo internacional de expertos, profesionales de la industria y representantes de organizaciones que producen código abierto y ampliamente acordados estándares de seguridad para la World Wide Web. Como una comunidad activa, WASC facilita el intercambio de ideas y organiza varios proyectos de la industria. WASC constantemente publica información técnica, artículos aportados, directrices de seguridad y otra documentación útil. Empresas, instituciones educativas, gobiernos, desarrolladores de aplicaciones, profesionales de la seguridad y vendedores de software de todo el mundo utilizan los conceptos aportados por esta organización como apoyo en los desafíos presentados por la seguridad de las aplicaciones web.

La clasificación de amenazas WASC v2.0 es un esfuerzo cooperativo para aclarar y organizar las amenazas a la seguridad de un sitio web. Los miembros del Consorcio de Seguridad de Aplicaciones Web han creado este proyecto para desarrollar y promover la terminología estándar de la industria para describir estos problemas. Los desarrolladores de aplicaciones, los profesionales de seguridad, los proveedores de software y los auditores tendrán la capacidad de acceder a un lenguaje y definiciones coherentes para problemas relacionados con la seguridad web.

En la clasificación de amenazas podemos encontrar: Abuso de funcionalidad; Fuerza bruta; Desbordamiento de búfer; Spoofing de contenido; Credencial / Sesión de Predicción; Scripting entre sitios; Cross-Site Solicitud de Falsificación; Negación de servicio; Huellas dactilares; Formato de cadena; Contrabando de respuesta HTTP; División de respuestas HTTP; Contrabando de solicitudes HTTP; División de solicitud HTTP; Desbordamientos de enteros; Inyección LDAP; Inyección de Comando de Correo; Inyección de Byte Nulo; Comando de OS; Camino Traversal; Ubicación de recursos predecibles; Inclusión de archivos remotos (RFI); Desvío de enrutamiento; Sesión de fijación; Abuso de SOAP Array; Inyección de SSI; Inyección SQL; Abuso del redirector de URL; Inyección XPath; Error de atributo de XML; XML Entidades externas; Expansión de entidades XML; Inyección XML; Inyección XQuery.(WASC, 2011)

5. CONCLUSIONES

Al momento de desarrollar aplicaciones Web es importante tomar en cuenta el tema de la seguridad, a través de la implementación de patrones de desarrollo seguro, para poder mitigar el acceso no autorizado y la pérdida de información.

Desde que comienza el desarrollo de software en su fase de requerimientos, se debe pensar en la seguridad del mismo, para esto se dispone de un modelo de desarrollo seguro, que sirve como guía.

A medida que evoluciona el desarrollo de aplicaciones Web evolucionan los ataques a los mismos, por lo tanto es importante para poder mitigar los posibles ataques, tener conocimiento del funcionamiento de los mismos.

REFERENCIAS

(LujanWeb, 2002) Luján Mora, S. Programación de aplicaciones web: historia, principios básicos y clientes web 2002

(OWASP, 2005) OWASP Una Guía para Construir Aplicaciones y Servicios Web Seguros Edición 2.0 Black Hat 2005

(Higuera, 2014) Higuera, J. R. B. Assessment methodology of web applications automatic security analysis tools for adaptation in the development life cycle *Universidad Nacional de Educación a Distancia , España*, 2014

(CheswickW.R, 2003) Cheswick W. R. Bellovin S. M., R. A. Firewalls And Internet Security Repelling The Wily Hacker. 2 Rev Ed. *Pearson Education 2003. ISBN: 9780201634662. ISBN-10: 020163466X.*, 2003

(Neville-Neil, 2007) Neville-Neil, G. V. Building Secure Web Applications *Queue, ACM*, 2007, *5*, 22-26

(Near, 2016) Near, J. P. & Jackson, D. Finding Security Bugs in Web Applications Using a Catalog of Access Control Patterns *Proceedings of the 38th International Conference on Software Engineering, ACM*, 2016, 947-958

(MarkDowd, 2006) Mark Dowd John McDonald, J. S. The Art of Software Security Assessment - Identifying and Preventing Software Vulnerabilities ISBN-10: 0-321-44442-6 , ISBN-13: 978-0-321-44442-4 *Addison Wesley Professional*, 2006

(Mavromoustakos, 2016) Mavromoustakos, S.; Patel, A.; Chaudhary, K.; Chokshi, P. & Patel, S. Causes and Prevention of SQL Injection Attacks in Web Applications *Proceedings of the 4th International Conference on Information and Network Security, ACM*, 2016, 55-59

(Zhang, 2011) Zhang, K. X.; Lin, C. J.; Chen, S. J.; Hwang, Y.; Huang, H. L. & Hsu, F. H. TransSQL: A Translation and Validation-Based Solution for SQL-injection Attacks *2011 First International Conference on Robot, Vision and Signal Processing*, 2011, 248-251

(Antunes, 2009) Antunes, N. & Vieira, M. Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing*, 2009, 301-306

(Weiss, 2012) Weiss, A. How to Prevent SQL Injection Attacks. Recuperado de http://www.esecurityplanet.com/hackers/how-to-prevent-sql-injection-attacks.html, 25 de Junio de 2017, 2012

(Karam2009) Karam, O. & Peltsverger, S. Teaching with Security in Mind *Proceedings of the 47th Annual Southeast*

Regional Conference, ACM, 2009, 68:1-68:4

(Gutierrez, 2005) Gutierrez, C.; Fernandez-Medina, E. & Piattini, M. Web Services Enterprise Security Architecture: A Case Study *Proceedings of the 2005 Workshop on Secure Web Services, ACM*, 2005, 10-19

(Token, 2006) Web Services Security Kerberos Token Profile 1.1 *OASIS*, 2006

(C.Gutierrez, 2005) C. Gutiérrez E. Fernández-Medina, M. P. Seguridad en Servicios Web *Universidad de Castilla-La Mancha, España.Informe Técnico UCLM DIAB-05-01-2*, 2005

(Saltzer, 1984) Saltzer, J. H.; Reed, D. P. & Clark, D. D. End-to-end Arguments in System Design *ACM Trans. Comput. Syst., ACM*, 1984, *2*, 277-288

(Daud, 2010) Daud, M. I. Secure Software Development Model: A Guide for Secure Software Life Cycle *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol I, IMECS 2010, March 17-19,2010, Hong Kong*, 2010

(Hope, 2007) Hope, P. & White, P. Software Security Requirements – the foundation for security,Cigital Inc, http://www.cigital.com, 2007

(Criteria, 2005) Criteria, C. Part 2: Security functional requirements August 2005 Version 2.3 Recuperado de https://www.commoncriteriaportal.org/files/ccfiles/ccpart2v2.3.pdf, 5 de Agosto de 2017, 2005

(Mead, 2008) Mead, J. H. A. S. J. B. R. J. E. G. M. N. R. Software Security Engineering: A Guide for Project Managers. ISBN 10: 032150917X ISBN 13: 9780321509178 *Addison-Wesley Professional*, 2008

(McGraw, 2006) McGraw, G. R. Software Security: Building Security In. ISBN-13: 978-0321356703 - ISBN-10: 0321356705 *Addison-Wesley Professional*, 2006

(owasp, 2017) OWASP Recuperado de https://www.owasp.org/index.php/Top_10_2017-Top_10, 15 de Junio de 2017, 2017

(Systems, 2015) Systems, W. R. THE CWE/SANS TOP 25 SECURITY VULNERABILITIES, 2015

(WASC, 2011) v2.0 null, T. W. T. C. Recuperado de http://projects.webappsec.org/w/page/13246978/Threat%20Classification, 23 de Junio de 2017, 2011

ANEXO 7. OPERACIONALIZACIÓN DE VARIABLES

Variable	Dimensión	Indicador	ítem
CODIFICACIÓN SEGURA (Mark Dowd, 2006)	Desarrollo Web	Lenguaje de Programación	3. ¿Cuál o cuáles lenguajes de programación utiliza la institución para desarrollar aplicaciones web?
			4. ¿Las librerías JavaScript utilizadas en los desarrollos web de la Institución, provienen de fuentes confiables o de sitios oficiales?
		Servicios Web	5. ¿Las aplicaciones web de su Institución, intercambian datos entre ellas por medio de Servicios Web (WS)?
			6. ¿Los Servicios Web se basan en estándares y protocolos abiertos?
			7. En caso de ser positiva la respuesta anterior, indique ¿Cuáles son utilizados por las aplicaciones web de su Institución?
		Sistemas de gestion de base de datos	13. ¿Indique cuáles de los siguientes sistemas de gestión de bases de datos se utilizan para el desarrollo de aplicaciones web?
			15. ¿En el sistema de gestión de base de datos se aplican reglas de integridad (llaves primarias y foráneas)?
		Configuracion del entorno de desarrollo	21. ¿Alguno de los programas que soportan su aplicación web, está desactualizado? (Sistema Operativo, servidor Web/App, motor de Base de Datos, API y todos los componentes de bibliotecas)
			22. ¿En los servidores se habilitan o instalan funciones innecesarias, como por ejemplo, puertos, servicios, páginas, cuentas, privilegios?
			23. ¿Se tiene definido los roles de las personas que ingresan con permisos de administrador a los servidores de aplicaciones y bases de datos?
			24. ¿Se tiene control sobre los permisos establecidos para las carpetas y ficheros de las aplicaciones web?
			25. ¿Su manejo de errores revela rastros de pila u otros mensajes de error demasiado informativos a los usuarios?
			26. ¿Cuenta con una arquitectura de aplicación robusta, que proporciona una separación efectiva y segura entre los componentes?
			28. ¿Se cuenta con un control de versiones de los aplicativos web, el cual permita identificar los cambios que ha sufrido la aplicación durante el ciclo de vida o que permita volver a una versión anterior en caso de alguna falla?
	Vulnerabilidad Software	Control de autenticación	8. ¿La autenticación de las aplicaciones web, garantiza un correcto funcionamiento en las sesiones y evita la suplantación?
			9. ¿Se controla el inicio, duración y cierre de la sesión que utiliza el usuario?
			10. ¿Las aplicaciones web solicita de forma periódica el cambio de la contraseña, comprobando la longitud y seguridad de los datos ingresados por el usuario?
			11. ¿Las contraseñas se almacenan con un algoritmo diseñado específicamente para la protección por contraseña, como por ejemplo bcrypt, PBKDF2 o scrypt?
			12. ¿Existe una autenticación mutua entre el cliente que accede a los servicios y el servidor que proporciona el servicio?
			16. ¿Se cuenta con usuarios distintos en la base de datos para cada desarrollo web, así como sus propios privilegios?
			17. ¿Las credenciales de acceso a la base de datos se almacenan en alguna parte del desarrollo?
		Validacion de datos	14. ¿La información almacenada en la base de datos, está validada mediante el uso de constraints (restricciones aplicadas a los objetos de una base de datos: default, not null, unique,check) restringiendo de esta forma la entrada para cada campo?
			20. ¿Los formularios de las aplicaciones web, tienen establecidos un identificador aleatorio único para cada usuario que inicie sesión? (donde el sistema revisa si el formulario es válido, comparando el identificador generado en la sesión antes de enviar una petición).
			31. ¿La validación de los datos de entrada (longitud, tipo de dato, rango de valores aceptados, sintaxis) de las aplicaciones, se realizan del lado del servidor así como del cliente?
			33. ¿Se cuenta con algoritmos criptográficos para proteger los datos sensibles de la Institución?
			34. ¿Los datos considerados sensibles (contraseñas, números de tarjetas de crédito, información personal, pagos registrados en la Institución) tienen algún nivel de protección adicional?
			35. ¿Los formularios que solicitan el ingreso de datos sensibles, tienen deshabilitado la opción de autocompletado y almacenamiento en cache?
			30. ¿Las interfaces de acceso público están protegidas mediante encriptación de datos, token de seguridad o algún tipo de comprobación de control sobre las mismas?
		Mecanismos de proteccion de la aplicación	36. ¿Las aplicaciones tienen protección de ataques, que permite identificarlos, bloquearlos y proporcionar algún detalle, como por ejemplo, si se está utilizando la aplicación de una forma que un usuario común nunca haría? (tiempo demasiado alto, entradas atípicas, patrones de uso inusuales, peticiones repetidas)
			37. ¿En caso de un comportamiento atípico, las aplicaciones de su Institución, bloquean solicitudes, direcciones IP o rangos de direcciones IP?
			38. ¿Las aplicaciones de su Institución, bloquean las cuentas de los usuarios que le dan mal uso a las mismas?
		Validacion por perfiles o roles	18. ¿Los permisos de acceso de los usuarios a la base de datos, son los adecuados de acuerdo a los privilegios que debería tener?
			19. ¿Las aplicaciones web, garantizan que el usuario esté autorizado mediante un rol, perfil o algún tipo de comprobación de control de acceso, para interactuar con las funcionalidades de la aplicación o acceder a los datos de la misma?
		Prevencion de ataques	32. ¿Los aplicativos web desarrollados en su Institución, poseen controles para evitar ataques de inyección?
			29. ¿Las aplicaciones web tienen implementado algún framework o librería anti-XSS que ayude a disminuir los inconvenientes con este tipo de ataques?
	Política de desarrollo seguro	Normativa institucional	1. ¿La Institución cuenta con políticas para el desarrollo seguro de aplicaciones web?
			2. ¿La Institución tiene establecida una arquitectura de desarrollo de software web documentada y estandarizada?
		Mantenimiento de arquitectura de desarrollo	27. ¿Se establecen ventanas de mantenimiento periódicas sobre la arquitectura que soporta las aplicaciones web? (para verificar que las configuraciones están correctas, se realice la revisión de logs de seguridad y actualización de software, entre otras).