Universidad de las Ciencias Informáticas

"Facultad 2"

Título: Plataforma de Gestión de Seguridad Informática. Módulo Pruebas de Intrusión.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO INFORMÁTICO

Autor: Lilibeth Abrahan Rodríguez

Tutor: Ing. Lilian Teresa Castro Mecias

La Habana, Junio, 2011

"Año 53 de la Revolución"

PENSAMIENTO

“***Sólo el hombre que alberga en su espíritu la fuerza de la nobleza forja el camino de sus mayores logros."***

José de Jesús Quintero.

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Telemática (TLM) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Lilibeth Abrahan Rodríguez	Ing. Lilian Teresa Castro Mecias
Autor	Tutor

Agradecimientos

AGRADECIMIENTOS

A mi mamá por siempre guiar todos mis pasos y todas las decisiones de mi vida con mucho amor y sabiduría para así poder obtener mis triunfos.

A toda mi familia por apoyarme y ayudarme en todo lo que han podido y siempre darme todo su cariño.

A mi novio que me dado siempre su apoyo y su amor incondicionalmente, dándome fuerzas para seguir a adelante y poder realizar mi sueño.

A mi tutora Lilian que siempre ha estado ahí dándome su ayuda y apoyo, corrigiendo todos mis errores y aclarando mis dudas.

A todos mis compañeros que han estado conmigo durante estos 5 largos años compartiendo todo lo bueno y malo de nuestras vidas.

A los que están conmigo en el laboratorio de proyecto que de una forma u otra me han ayudado muchísimo.

A mis amiguitas Carmen, Glenda y Yaily por siempre estar ahí cuando las necesito y por apoyarme en todo.

A mis profesores que desde primer año hasta ahora han hecho de mí toda una profesional.

A una persona que no puedo dejar de mencionar que tuvo gran significado en mi vida como estudiante y mi ayudó muchísimo en mis estudios al que tengo que agradecer parte de este triunfo a Omar Pimentel gracias.

A la revolución por permitirme estudiar en una universidad como esta y realizar mi sueño de ser una Ingeniera.

Gracias a todos por permitirme realizar este gran sueño.

DEDICATORIA

Le dedico esta tesis a mi mami por ser la luz de mis ojos y el aire que mueve todo en mi vida y por ser la persona que más confió en mí.

RESUMEN

La seguridad informática tiene como primicia lograr la confidencialidad, integridad y disponibilidad de la información. La comprobación de vulnerabilidades mediante pruebas de intrusión constituye una de las medidas de seguridad encaminadas a conseguir la disminución de riesgos de seguridad y por tanto altos niveles de seguridad en los sistemas informáticos antes de ser puestos en el entorno de producción. La presente aplicación ha surgido a partir de la necesidad de gestionar el proceso de pruebas de intrusión que se ejecutan a los proyectos de la Universidad antes de ser entregados al cliente, con el objetivo de garantizar calidad y fortaleza ante posibles vulnerabilidades, dicho proceso realizado en el Laboratorio de Seguridad Informática (LabSI). La aplicación brindará al usuario la posibilidad de que los datos de su proyecto y solicitud sean persistentes, ver las vulnerabilidades detectadas sin que necesariamente hayan terminado las pruebas, también obtener un reporte final con todas las vulnerabilidades encontradas y la posible forma de mitigación. Además facilitará a los especialistas el diseño de las pruebas y la conformación de dicho reporte final.

PALABRAS CLAVE

Pruebas de seguridad, Reporte, Vulnerabilidades.

Índice

ÍNDICE

Índice

Índice

Índice

Introducción

INTRODUCCIÓN

En la Universidad de las Ciencias Informáticas (UCI) se concentra en la actualidad la producción de software y servicios informáticos del país, organizada a través de una red de Centros que desarrollan sistemas en diferentes ramas del conocimiento para la exportación y la necesaria informatización de la sociedad cubana.

Como entidad desarrolladora de software es su responsabilidad el aseguramiento de la calidad de los sistemas antes de ser entregados al cliente. El Centro de Calidad para soluciones tecnológicas (CALISOFT) y el Centro de Telemática (TLM) implementan el proceso para la ejecución de un conjunto de pruebas que garanticen la calidad de los sistemas informáticos desarrollados en la institución, esas pruebas realizadas además de validar la calidad de un sistema también comprueban su seguridad mediante la detección de vulnerabilidades a través de Pruebas de intrusión en el Laboratorio de Seguridad Informática (LabSI).

Las pruebas de intrusión se fundamentan en comprobar los métodos de protección del sistema. En este tipo de análisis además de encontrar las vulnerabilidades presentes en las aplicaciones, se verifica que los mecanismos de protección incorporados funcionan. El objetivo es lograr acceso no autorizado al sistema siguiendo patrones de ataques ya conocidos. Una Prueba de intrusión permite al desarrollador obtener una visión clara de los puntos más débiles de la aplicación y en correspondencia tomar las medidas de seguridad adecuadas.

La seguridad, en el contexto de la informática, tiene como primicia lograr la confidencialidad, integridad y disponibilidad de la información. La comprobación de vulnerabilidades mediante pruebas de intrusión constituye una de las medidas de seguridad encaminadas a conseguir la disminución de riesgos de seguridad y por tanto altos niveles de seguridad en los sistemas informáticos antes de ser puestos en el entorno de producción.

El Laboratorio de Seguridad Informática del Centro de Telemática no cuenta con un sistema que gestione el proceso para la realización de las Pruebas de Intrusión que facilite la recopilación de información de la aplicación a probar, lo cual origina los siguientes inconvenientes:

- La información referente al proyecto obtenida en la reunión de inicio se realiza solamente mediante una entrevista, por lo que no hay persistencia de dicha información.

- La utilización de varias herramientas de seguridad, hace difícil la correlación de información para conformar el reporte final con las vulnerabilidades encontradas.
- Durante la ejecución de las pruebas el cliente no puede ver las vulnerabilidades hasta que se finalicen las mismas, por tanto solo se pueden resolver cuando finalice las pruebas en su totalidad.
- No se cuenta con un historial de las vulnerabilidades de seguridad encontradas en los proyectos comprobados que ayuden en la toma de decisiones.

Debido a dicha problemática el **problema a resolver** está determinado de la siguiente forma: ¿Cómo gestionar el proceso de pruebas de intrusión en el Laboratorio de Seguridad Informática (LabSI)?
Definiendo como **objeto de estudio**: Las pruebas de intrusión a sistemas informáticos y como **campo de acción**: El proceso de pruebas de intrusión a sistemas informáticos en el Laboratorio de Seguridad Informática (LabSI).

El **objetivo general** de esta investigación es: Desarrollar un sistema para la gestión de las pruebas de intrusión a sistemas informáticos en el Laboratorio de Seguridad Informática (LabSI).

Para cumplir con el objetivo general trazado se desarrollan las siguientes **tareas investigativas**:

- Selección y revisión bibliográfica que permita definir el estado del arte de la investigación.
- Procesamiento y análisis crítico de la información obtenida.
- Análisis del proceso definido para la ejecución de las pruebas de seguridad en el Laboratorio de Seguridad (LabSI).
- Realización del Modelado del Negocio resultante del estudio del proceso de pruebas de seguridad en el Laboratorio de Seguridad (LabSI).
- Diseño e Implementación del Sistema para la gestión de las Pruebas de Seguridad del proyecto LabSI.
- Validación de la propuesta de solución mediante la realización de pruebas de calidad.

En el desarrollo de la investigación se utilizaron los siguientes métodos:
Métodos Teóricos:

- Analítico - sintético: Este método permitirá analizar las teorías y documentos referentes a la ejecución de pruebas de intrusión, facilitando de esta forma la extracción de los elementos más importantes relacionados con el objeto de estudio.
- Modelación: Este método resultará útil para la realización del sistema debido a la selección de la metodología que se utilizará, en la mayoría de estas se hace muy necesaria la creación de varios modelos, pues estos permitirán una reproducción ampliada de la realidad, además de que posibilitará descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

Métodos Empíricos:

Observación: La observación científica estudia el proceso de pruebas de intrusión, se realiza de forma consciente y orientada al trabajo de los especialistas en el Laboratorio de Seguridad (LabSI).

La presente investigación está estructurado de la siguiente manera: Introducción, Desarrollo estructurado en 4 capítulos detallados a continuación:

Capítulo # I: Fundamentación Teórica.

Se realiza un estudio de varias aplicaciones de Gestión tanto en el ámbito nacional como internacional y se definen las herramientas y metodologías que se utilizarán en la propuesta de solución.

Capítulo # II: Características del Sistema.

El objetivo de este capítulo es obtener una visión más específica del sistema, donde se definirán los requisitos funcionales y no funcionales que regirán el desarrollo de la solución al problema. Además, se identificarán los conceptos más significativos del dominio del problema, sus relaciones y los atributos que lo componen.

Capítulo # III: Diseño del sistema.

En este capítulo se presentarán los diagramas de clases de diseño, donde se especificarán las responsabilidades de las clases y sus relaciones.

Capítulo # IV: Implementación y Prueba.

Introducción

En este capítulo se desarrolla la solución general de la aplicación, se definen los patrones de diseño, se presenta la arquitectura utilizada y la implementación de los componentes del sistema, así como las pruebas de funcionalidad para validar el funcionamiento del sistema.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

La evolución de las tecnologías para satisfacer necesidades que plantean los negocios hace que las aplicaciones, datos e información tengan un grado de exposición cada vez mayor, es por esto que la necesidad de la seguridad informática cada día va tomando una importancia más alta en el mundo de las tecnologías. En el siguiente capítulo se realiza un estudio del arte de aplicaciones de seguridad y se especifican las tecnologías empleadas en el desarrollo de la solución informática propuesta.

1.2 Seguridad Informática.

La seguridad informática es una disciplina que se especializa en proteger la integridad y la privacidad de la información almacenada en un sistema informático. La seguridad son aquellas reglas, técnicas y/o actividades destinadas a prevenir, proteger y resguardar lo que es considerado como susceptible de robo, pérdida o daño, ya sea de manera personal, grupal o empresarial.
Para garantizar la seguridad de aplicaciones aunque esta no es absoluta se le practican pruebas mediante herramientas y metodologías para minimizar los posibles riesgos a la infraestructura o a la información. (1)

1.3 Proyecto OWASP.

Open Web Application Security Project (OWASP), en español Proyecto de seguridad de aplicaciones web de código abierto. El objetivo principal de OWASP es determinar y contrarrestar las causas que hacen a un sistema Web inseguro. Es un proyecto de software libre que brinda artículos, herramientas, metodologías, documentación y tecnologías que pueden ser usadas de forma gratuita. OWASP ha producido varias guías que funcionan juntas para crear una base de conocimientos en seguridad de aplicaciones. La guía de desarrolladores cubre todos los controles de seguridad que los desarrolladores de software deben utilizar. Hay muchas formas diferentes de probar fallos de seguridad, y esta guía unifica el consenso de los principales expertos sobre cómo realizar esta comprobación rápida, exacta y eficientemente. (2)

1.3.1 Guía de Pruebas de OWASP.

Brinda a los usuarios una guía de las técnicas más comunes para realizar pruebas de intrusión a aplicaciones y servicios Web. Ayuda a comprender el porqué, cómo, qué y cuándo probar las aplicaciones web, brinda un listado de comprobación y con un informe de acciones específicas que deben ser atendidas. Además realiza un marco de desarrollo de pruebas para que se puedan crear programas de prueba y evaluar otros procesos. Estas herramientas pueden ser muy útiles, ya que encuentran muchas incidencias potenciales. Aunque ejecutar estas herramientas no toma mucho tiempo, cada uno de los problemas potenciales toma su período en investigar y verificar. Si el objetivo es encontrar y eliminar los fallos más serios tan rápidamente como sea posible, se tiene en consideración si el tiempo está mejor empleado usando herramientas automatizadas. Estas herramientas son ciertamente parte de un programa de seguridad de aplicaciones bien equilibrado, además pueden ofrecer soporte para procesos globales y producir código más seguro. (2)

1.4 Intrusión.

Como intrusión se entiende la realización de un acto no autorizado, como pueda ser el acceso a un sistema, la ejecución de programas no autorizados o el ataque a una red informática. También se considera intruso a toda aquella persona o entidad que intenta acceder a un sistema informático sin autorización. (3)

En el contexto de la seguridad informática se utiliza el término con fines éticos aunque se emplean herramientas utilizadas por los intrusos o hackers, la diferencia consiste en que en este caso el objetivo es detectar vulnerabilidades previa autorización de los propietarios del sistema.

1.4.1 Pruebas de Intrusión.

Una prueba de intrusión es el proceso para evaluar la seguridad en sistemas. Los sistemas y políticas de seguridad son analizados exhaustivamente con el fin de encontrar fallos de seguridad, tanto en el diseño, como en la implementación de la aplicación. Este proceso se realiza con herramientas de seguridad automatizadas que hacen metódicamente esta tarea. También se pueden hacer estas pruebas manual mente cuando se trata de alguna aplicación con características específicas, siendo realizada por los especialistas.

Estas pruebas ofrecen a la empresa no solo la tranquilidad de saber que sus sistemas están seguros, sino también la posibilidad de cumplir con las regulaciones de seguridad más estrictas establecidas por gobiernos y entidades reguladoras. (4)

Asegurándose de validar que al menos los 6 conceptos básicos en estas pruebas son cubiertos:

- Confidencialidad: los datos tienen que ser legibles únicamente para los usuarios autorizados.
- Integridad: la seguridad de que una información no ha sido alterada, borrada, reordenada, copiada con información modificable sólo por las personas autorizadas.
- Autentificación: detecta y comprueba la identidad mediante el examen de las credenciales del usuario y la validación de las mismas consultando a una autoridad determinada.
- Disponibilidad: garantizar el correcto funcionamiento de los sistemas de información.
- Negación de culpa: de información constituye la garantía de que ninguna de las partes involucradas pueda negar en el futuro una operación realizada.

1.5 Herramientas de Intrusión.

Son herramientas software que tienen como objetivo buscar los puntos vulnerables en tecnologías y sistemas. Una vez detectados los puntos débiles, estas herramientas pueden aprovecharlos para diversos fines, como pueden ser robo de información, elevación de privilegios, provocar una denegación de servicio, etc. Estas herramientas no deben considerarse siempre como algo negativo. Si se utilizan de forma éticamente correcta, pueden servir para descubrir los puntos débiles de un sistema y poder asegurarlos posteriormente. (5)

A continuación se presentan algunas distribuciones para la realización de Pruebas de Intrusión.

1.5.1 Livecd de OWASP.

Es una distribución que se centra en ofrecer los recursos del OWASP desde uno CD auto arrancable, tanto aplicaciones como documentos. El objetivo general de este proyecto es hacer que las herramientas de seguridad de aplicaciones y la documentación que se utilizan para realizar pruebas de penetración sean de fácil acceso. (6)

1.5.2 Backtrack Penetration Testing Distribution.

Es una distribución de Linux que permite la realización de completos análisis informáticos y pruebas de seguridad. Fue pensada y diseñada para la auditoría de seguridad y relacionada con la seguridad informática en general. Actualmente tiene una gran popularidad y aceptación en la comunidad que se mueve en torno a la seguridad informática. Los análisis que se realizan con esta aplicación son numerosos porque incluye una larga lista de herramientas de seguridad listas para usar, entre las que destacan scanner de puertos y vulnerabilidades, archivos de exploits[1], sniffers[2], herramientas de análisis forense y herramientas para la auditoría Wireless. Backtrack le ofrece al usuario una extensa colección de herramientas completamente usables desde un Live DVD o un Live USB por lo que no requiere una instalación para poder usarse. (7)

1.5.3 Samurai Web Testing Framework.

Samurai Web Testing Framework es un entorno de trabajo basado en GNU/Linux Ubuntu, que ha sido pre-configurado para llevar a cabo pruebas de penetración a aplicativos Web.

Este LiveCD, que además puede ser instalado como sistema operativo por defecto en el disco duro, contiene un repertorio bastante amplio en cuanto a herramientas de libre uso y distribución se refiere. Estas herramientas están destinadas a realizar pruebas y ataques sobre aplicaciones Webs.

Al igual que la distribución BackTrack, Samurai Web Testing Framework divide las herramientas por grupos según una metodología.

Comienza por la etapa de reconocimiento, para ello hace uso de las siguientes herramientas: Fierce domain Scanner y Maltego. Para el mapeo del sistema objetivo incluye la herramienta WebScarab y Ratproxy. Para el descubrimiento de vulnerabilidades incluye w3af y burp, finalmente para el proceso de explotación utiliza BeEF, AJAXShell y otras más.

[1] Pieza de software, un fragmento de datos, o una secuencia de comandos.

[2] Captura todos los datos que pasan a través de una tarjeta de red.

Samurai Framework también incluye una wiki pre configurado y lista para ser usada como bitácora de recolección de la información que se va generando a medida que avanza en el proceso. (8)

1.6 Pruebas de intrusión en el Laboratorio de Seguridad (LabSI)

La guía de pruebas de OWASP y la lista de chequeo que proporciona AppSecTestingChecklist, se emplean en la realización de las pruebas a las aplicaciones que solicitan el servicio al Laboratorio de Seguridad. A continuación se describen una representación de las pruebas y herramientas asociadas a estas.

1.6.1 Pruebas de gestión de configuración de la aplicación.

En esta prueba se utilizan escáneres de vulnerabilidades con el objetivo de determinar si un sistema es vulnerable a ataques conocidos. La valoración de vulnerabilidades representa un caso esencial del proceso de la detención de intrusos. Estos sistemas que realizan valoraciones de vulnerabilidades buscan servicios y configuraciones con vulnerabilidades conocidas. (9)
Entre ellos los más usados son:

- Nikto: es un scanner de código abierto (Licencia GPL) que realiza exhaustivos pruebas de penetración a aplicaciones web.(10)
- Wapiti: es un escáner de vulnerabilidades para aplicaciones web, licenciado bajo la GPL v2, que busca fallos XSS[3], inyecciones SQL y XPath[4], inclusiones de archivos (local y remota), ejecución de comandos, inyecciones LDAP[5], y inyecciones CRLF[6].(11)
- Wikto: realiza un escáner en el entorno de un servidor Web para encontrar vulnerabilidades, extracción de directorios y vínculos.(12)

1.6.2 Descubrimiento de aplicaciones y gestión de configuración de la infraestructura

Escáner de puertos y servicios:

[3] Cross Site Scriptting. Vulnerabilidad que afecta el código JavaScript de la página.

[4] XML Path Language. Se utiliza para direccionar partes de un documento XML

[5] Protocolo compacto de acceso a directorios.

[6] Se refiere a la combinación de dos códigos de control: CR (retorno de carro) y LF (salto de línea).

Se utilizan para detectar qué servicios comunes está ofreciendo la máquina y posibles vulnerabilidades de seguridad según los puertos abiertos, detecta si un puerto está abierto, cerrado, o protegido por un cortafuego. También puede llegar a detectar el sistema operativo que está ejecutando la máquina según los puertos que tiene abiertos. Es usado por administradores de sistemas para analizar posibles problemas de seguridad, pero también es utilizado por usuarios malintencionados que intentan comprometer la seguridad de la máquina o la red. (13)

Existen varios programas escaneadores de puertos por la red:

- Nmap: es una herramienta para administradores de sistemas interesados en el escaneo de grandes o pequeñas redes para determinar los equipos que se encuentran activos y cuáles son sus servicios. Es un escáner de puertos muy potente, puede determinar si un servidor o máquina está en uso y que servicios ofrece. (14)

- OpenVas: es una herramienta que permite identificar las vulnerabilidades de un equipo o servidor desde otra PC remota, mostrando cuales son las posibles vulnerabilidades a las que puede estar expuesto el equipo analizado durante un ataque remoto, la gravedad de esas vulnerabilidades y unas recomendaciones para su solución. (15)

- Nessus: trabaja en modo cliente-servidor. El servidor es quien se encarga de realizar todas las auditorías y el cliente es la interfaz por donde se le dan las indicaciones correspondientes, como por ejemplo IP o dominio de la máquina a escanear, tipo de auditoría y modo de informe. (16)

1.6.3 Pruebas de Validación de Datos.

La debilidad más común en la seguridad de aplicaciones web, es la falta de una validación adecuada de las entradas procedentes del cliente o del entorno de la aplicación. Se deben comprobar todas las formas posibles de validación de entradas, para poder comprender si la aplicación es lo suficientemente resistente ante cualquier tipo entrada de datos. (17)

- Sqlmap: es una herramienta desarrollada en python[7] para realizar inyección de código sql automáticamente. Su objetivo es detectar y aprovechar las vulnerabilidades de inyección SQL en aplicaciones web. (18)

- Sqlbrute: es una herramienta para forzar la salida bruta de datos de bases de datos utilizando vulnerabilidades de inyección SQL ciega. Es compatible con el tiempo y error basado en tipos de explotación desde el Microsoft SQL Server y error basado en la explotación de Oracle. (19)

1.7 Sistemas de Gestión de la Seguridad Informática

Existe en el mundo variedad de aplicaciones que gestionan diversas informaciones diariamente. Estos sistemas cumplen con la función de informatizar ciertas tareas que habitualmente se hacen en forma manual y que resultan engorrosas o insuficientes. Surgen a partir de la necesidad de las empresas modernas de contar con medios más rápidos para el procesamiento de datos, de manera confiable, en tiempo y forma. Los sistemas de gestión orientados a la seguridad informática de forma general capturan información que faciliten a especialistas de la seguridad informática la toma de decisiones en cuanto a eventos de seguridad se refiere. (20)

1.7.1 Open Source Security Information Management (OSSIM)

AlienVault OSSIM es una solución completa para la gestión de la seguridad que incluye un número de funciones que permiten detectar y caracterizar un ataque, así como un sistema completo de gestión inteligente de la seguridad.
La versión profesional de AlienVault ofrece todas las funciones necesarias para la gestión de la seguridad en entornos profesionales.
Las principales características son:

- Alto volumen de almacenamiento de log firmados digitalmente.
- Soporte a ambientes multiplataforma y multinivel.
- Diseñado para altos volúmenes de registros.
- Alta disponibilidad y continuidad.

[7] Lenguaje de programación.

Su objetivo principal es proporcionar una recopilación completa de herramientas que, al trabajar la red junto a los administradores obtienen una vista detallada sobre cada aspecto de sus redes, los dispositivos físicos de acceso y el servidor. Además de obtener el máximo provecho de conocidas herramientas de código abierto. OSSIM proporciona un motor de correlación, visualización y presentación de informes y herramientas de manejo de incidentes, sobre la base de un conjunto definido de activos como anfitriones, redes, grupos y servicios. Toda esta información puede ser restringida por la red o el sensor con el fin de ofrecer la información necesaria para usuarios específicos. (21)

Aunque OSSIM permite gestionar la seguridad, no se considera su aplicación en LabSI porque está destinada para la administración de eventos de seguridad en una infraestructura determinada, tales como detección de intrusos, disponibilidad de hosts y servicios, etc.

Sin embargo LabSI tiene como objetivo fundamental la inclusión de la Seguridad durante el proceso de desarrollo de software.

1.7.2 Sistema de Gestión de Reportes de vulnerabilidades.

El laboratorio de Seguridad Informática (LabSI) cuenta un Sistema de Gestión de Reportes de vulnerabilidades, el cual gestiona parte del proceso de pruebas de intrusión, las vulnerabilidades encontradas y los reportes dados por las pruebas que se realicen en el proyecto LABSI; este sistema permite gestionar de forma particular los resultados arrojados por las auditorías a las aplicaciones probadas por el proyecto, planificar las pruebas y adicionar usuarios. (22)

Sin embargo el Sistema tiene como inconveniente que los resultados eran almacenados en el mismo formato que arroja la herramienta utilizada, por ejemplo, si en una prueba se utilizaban 5 herramientas cuyos reportes tenían los formatos xml, txt y html, el especialista en el momento de entregar el reporte final de vulnerabilidades debía consultar cada uno de ellos y conformar un documento de texto con toda esa información. Tampoco permite gestionar el proceso inicial de las pruebas desde realizar la solicitud de las pruebas, diseñar las pruebas, ni realizar un reporte final de vulnerabilidades encontradas.

Capítulo I: Fundamentación Teórica

1.8 Tecnologías utilizadas en el desarrollo de la propuesta.

1.8.1 RUP como Metodología de desarrollo

RUP (Rational Unified Process) constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos, permitiendo adaptarse a las necesidades de cualquier proyecto. Dicha metodología facilita el mantenimiento y soporte de la aplicación, al contar con artefactos por cada una de las fases durante el desarrollo. (23)

Se caracteriza por ser iterativo e incremental, de esta forma en cada iteración se puede llevar al cliente una versión del producto, con el fin de obtener mejoras para el software, alcanzando así la mejor calidad durante el ciclo de vida del proyecto y si existiera algún cambio no afectaría a las demás iteraciones.

Centrada en la arquitectura y guiado por casos de usos, esto permite que exista una trazabilidad entre el caso de uso y los demás artefactos vinculados a él. Incluye artefactos y roles, que son diseñados durante las diferentes fases, describiendo las características del software hasta lograr la entrega del producto, permitiendo reconocer los problemas, para así corregirlos de forma temprana.

1.8.2 Lenguaje Unificado de Modelado

UML (Lenguaje Unificado de Modelado) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software en cada una de las etapas por las que tiene que pasar. Está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. (24)

A través de UML es posible establecer los requerimientos y estructuras necesarias para un sistema de software, antes del código. Incluye aspectos conceptuales como los procesos de negocio y funciones del sistema, y aspectos concretos como son expresiones de lenguajes de programación, componentes reutilizables y esquemas de bases de datos lo que es de gran importancia para el modelado del sistema.

1.8.3 Herramienta de Modelado Visual Paradigm

Es una herramienta CASE (Computer Aided Software Engineering) que facilita el modelado de artefactos en el Proceso de Desarrollo de Software mediante el Lenguaje de modelado UML, soportando el ciclo de vida completo del desarrollo del software. Ofrece la posibilidad de documentar todo el trabajo sin

necesidad de emplear otras herramientas en varios formatos, como PDF u otras. Ayuda a una rápida construcción de aplicaciones de calidad. Permite además diseñar los diferentes tipos de diagramas de clases y generar códigos a partir de dichos diagramas. Es una herramienta multiplataforma. (25)
Soporta ingeniería inversa, importa proyectos de Rational Rose, genera informes, edita detalles de casos de uso y genera bases de datos, permitiendo la transformación de diagramas de entidad-relación. Dicha herramienta soporta varios usuarios trabajando sobre el mismo proyecto y permite control de versiones, por estas razones se considera colaborativa.

1.8.4 PHP5 como lenguaje de programación

PHP HipertextPreprocesor es un lenguaje del lado del servidor con una gran librería de funciones y mucha documentación, es rápido, gratuito e independiente de la plataforma, lo que significa que puede ser ejecutado en diferentes sistemas operativos como son: Windows, Linux, Mac OS X, entre otros. Es ampliamente utilizado en el mundo por los desarrolladores debido a su simplicidad y potencia, soporte para *Unicode (PHP* 6*).* (26)

Es un lenguaje de programación interpretado, de código abierto y de alto nivel, procesa la información de formularios, genera páginas con contenidos dinámicos. La utilización de este lenguaje de programación permite flexibilidad y robustez en el desarrollo de aplicaciones. Posee soporte para la Programación Orientada a Objetos y el manejo de excepciones, etc.

1.5.5 Framework de desarrollo.

Fig. 1 Zend Framework 1.11

Zend Framework es utilizado para el desarrollo de aplicaciones Web y servicios Web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. Zend Framework tiene como ventaja que es desarrollado por Zend que es la empresa que respalda comercialmente a PHP, tiene como principales características (27):

- Trabaja con el Modelo Vista Controlador (MVC)
- Cuenta con módulos para manejar archivos PDF.
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad.
- Completa documentación y pruebas de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.

Es seleccionado debido a que es un framework para el desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios Web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5.

1.8.6 PostgreSQL como Gestor de Bases de Datos

PostgreSQL es un sistema de bases de datos objeto-relacional. Es libre y su código fuente completo está disponible. Es una de las bases de datos más potentes y robustos del mercado, ofreciendo control de concurrencia multi-versión, soportando casi toda la sentencia SQL (incluyendo subconsultas, transacciones distribuidas, y tipos y funciones definidas por el usuario), además de un amplio conjunto de enlaces con lenguajes de programación como (C, PHP, C++, C#, Python, Java, Perl, entre otros). PostgreSQL funciona bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Dentro de sus características se destacan el trabajo de múltiples procesos dentro de una misma tabla sin que ocurran bloqueos, amplia variedad de tipos nativos como: números de precisión arbitraria, texto de largo ilimitado, figuras geométricas, direcciones IP (IPv4 e IPv6), direcciones MAC, arreglos, entre otros; llaves foráneas, disparadores (*triggers*), vistas, y herencia de tablas. (28)

Es utilizado teniendo en cuenta su eestabilidad, facilidad de administración e implementación de estándares. Además de que soporta ACID (es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.), o lo que es lo mismo, la

realización de transacciones seguras; también, vistas, uniones, claves extranjeras, procedimientos almacenados, triggers[8], etcétera.

1.8.7 Zend Studio 8.0

IDE (Integrated Development Environment) de Desarrollo PHP. Zend Studio o Zend Development Environment es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Tiene las siguientes características (29):

- Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- Detección de errores de sintaxis en tiempo real.
- Funciones de depuración: Botón de ejecución y traza, marcadores, puntos de parada (*breakpoints*), seguimiento de variables y mensajes de error del intérprete de PHP.
- Soporte para gestión de grandes proyectos de desarrollo.
- Manual de PHP integrado.
- Soporte para navegación en bases de datos y ejecución de consultas SQL.

Además de ser un estándar de la arquitectura del proyecto fue seleccionado para maximizar la productividad teniendo en cuenta que permite crear y mantener el código más rápido, resolver los problemas de aplicación de forma rápida

1.9 Conclusiones

En este capítulo se realizó un estudio del arte de los Sistemas de Gestión de Seguridad así como los términos o conceptos fundamentales tratados en la investigación por ejemplo Seguridad Informática, Proyecto OWASP, Pruebas de Intrusión, entre otros. Se definieron las principales herramientas utilizadas para las pruebas de intrusión. Se realizó una selección de las herramientas a utilizar para el desarrollo del proyecto definiendo como metodología de desarrollo a RUP, el Visual Paradigm para desarrollar los

[8]Un ***trigger*** es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación.

diagramas UML. Como gestor de base de datos PostgreSQL, Zend Studio como plataforma de desarrollo y PHP como lenguaje de programación para la implementación del proyecto.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción.

El presente capítulo abarca un estudio del problema planteado para diseñar una aplicación que resuelva los aspectos relacionados con la gestión del proceso de pruebas de seguridad. Se realiza el modelo de negocio, un análisis de los requerimientos necesarios para la realización de la aplicación; y se describen y confeccionan, los diagramas de casos de usos para dicha aplicación.

2.2 Objeto de automatización.

La Plataforma de Gestión de Pruebas de Seguridad tiene como objetivo gestionar todo el proceso de pruebas en LabSI. Como resultado se debe desarrollar una aplicación web que gestione la información durante todo el proceso, desde que se realiza la solicitud de pruebas, se diseñan las mismas con las herramientas definidas hasta la concepción del reporte final de vulnerabilidades generadas por estas herramientas con el fin de lograr su automatización y garantizar la persistencia de la información sobre las aplicaciones a las que se le realizan las auditorias.

2.3 Información que se maneja.

La información procesada en el sistema es de carácter confidencial, debido a que se registran vulnerabilidades encontradas en los proyectos en el proceso de pruebas. Por tanto esta información solo deber ser vista por el líder del proyecto y el especialista encargado de las pruebas.

2.4 Propuesta de sistema.

Se desea realizar una aplicación web para la gestión de las pruebas de intrusión realizadas en LabSI, que cuente con una interfaz amigable la cual permita adicionar proyectos, realizar solicitudes de pruebas a estos que contengan los aspectos definidos por los especialistas del laboratorio, buscar el proyecto, en caso de ser especialista, para conocer sus datos, vulnerabilidades encontradas y el resto de la información

relacionada con el proyecto, el diseño de pruebas de seguridad con sus herramientas correspondientes para el análisis de los sistemas, pueden ser diseñadas tanto manual como automáticamente a consideración del encargado de las pruebas. Además mostrar las vulnerabilidades encontradas durante las pruebas, obtener estadísticas sobre vulnerabilidades encontradas en los proyectos, conformar un informe final de vulnerabilidades de los reportes arrojados por las herramientas que realizan las pruebas.

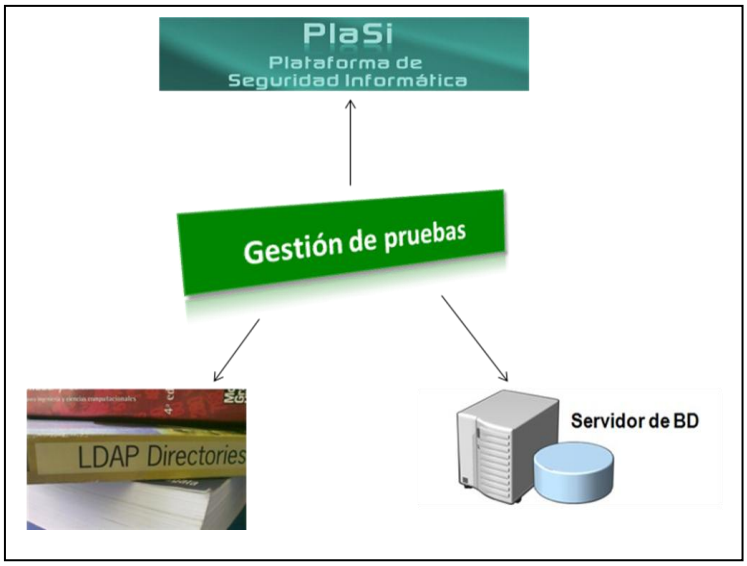

Figura 1. Propuesta del sistema (Gráfico)

2.5 Modelo del Negocio.

Organiza mejor las ideas para desarrollar un sistema bien organizado y estructurado. Proporciona una herramienta para identificar las relaciones estratégicas necesarias que garantizaran la sustentabilidad de la aplicación que se desarrollará. (30)

2.5.1 Actores del Negocio.

Actores del negocio	Justificación
Cliente	Solicita las pruebas al líder del proyecto de LABSI y una vez finalizada las mismas obtiene el reporte final de Vulnerabilidades encontradas.

Líder de Proyecto LabSI	Se encarga de proponer los especialistas para realizar las pruebas.

Tabla 1. Actores del Negocio

2.5.2 Trabajadores del negocio.

Trabajadores del negocio	Justificación
Especialista de Seguridad	Diseña las pruebas en el laboratorio de seguridad, conforma el informe final de vulnerabilidades.

Tabla 2. Trabajadores del Negocio

2.5.3 Diagrama de caso de uso del Negocio.

El diagrama de Caso de Uso del Negocio describe los procesos de un negocio, vinculados al campo de acción, y cómo se benefician e interactúan los socios y clientes en estos procesos asociados a un grupo de tareas lógicamente relacionadas que se llevan a cabo en una determinada secuencia y manera empleando los recursos de la organización para dar resultados en apoyo a sus objetivos. (31)

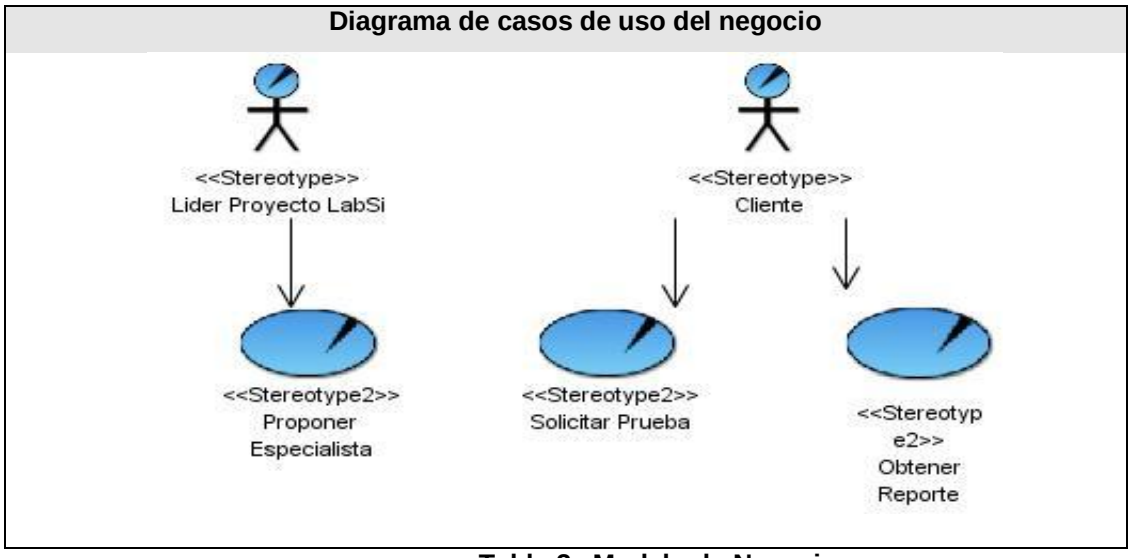

Tabla 3. Modelo de Negocio

2.6 Especificación de Requerimientos del sistema.

Los requerimientos son la condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

2.6.1 Requisitos Funcionales.

RF1 Autenticar Usuario.

RF2 Gestionar Permisos de usuarios.

RF2.1 Asignar Permisos de usuarios.

RF2.2 Modificar Permisos de usuarios.

RF2.3 Eliminar Permisos de usuarios.

RF3 Registrar solicitud de pruebas del proyecto.

RF4 Adicionar Proyecto.

RF5 Modificar Proyecto.

RF6 Buscar Proyecto.

RF7 Diseñar Pruebas.

RF8 Generar Reporte Final de Vulnerabilidades.

RF9 Mostrar Historial de Proyectos a los que se le han realizados pruebas.

RF10 Mostrar vulnerabilidades encontradas en el sistema.

RF11 Calcular estadísticas de Vulnerabilidades.

2.6.2 Requisitos No Funcionales.

Los requerimientos no funcionales especifican las propiedades o condiciones que el producto debe tener. (32)

Estos requerimientos están dados en las siguientes categorías:

Usabilidad

- El sistema brindará al cliente el manual de usuario y manual de instalación donde se deberá sugerir claramente las funcionalidades del sistema y la acción que se debe realizar en cada una para así lograr una familiarización del usuario con el nuevo sistema.

Apariencia o interfaz externa

- La interfaz será amigable y sencilla para el fácil manejo de los usuarios. No debe tener animaciones ni imágenes pesadas que comprometan la rapidez de la aplicación.
- El diseño deberá ser atractivo con colores que no resulten agotadores a la vista como verde, blanco y negro.
- Para nombrar los menús, botones y cajas de diálogo se solicitó opinión del cliente.

Rendimiento

El tiempo de respuesta del sistema en su caso más crítico, a la hora de generar un Informe final de vulnerabilidades estará determinado por la cantidad de vulnerabilidades que se tengan en la BD.

Seguridad

- Garantizar que la información esté disponible únicamente para las personas que estén en el dominio UCI, y que tengan los permisos asignados por el administrador del sistema, los usuarios solo podrán conocer los datos a los que tienen permiso, otorgados solamente por el administrador del sistema.
- No se podrá acceder al sistema con una url directamente, siempre habrá que autenticarse.
- Solo los usuarios con derechos de administración podrán realizar las funciones administrativas.
- Las contraseñas deberán ser personales con el objetivo de autenticar el acceso y no será visible al ser introducida.

Soporte

- Se entregara la documentación resultante de aplicar la metodología de RUP así como el Manual de instalación para facilitar el mantenimiento del sistema.

2.7 Modelo de Caso de Uso del Sistema

2.7.1 Definición de Actores del Sistema

Actores	Justificación
Administrador del Sistema	Gestiona los permisos a usuarios del sistema y el control de acceso en dependencia del rol que desempeñen los usuarios.
Especialista de Seguridad	Es el encargado de gestionar el diseño de las pruebas, conformar los reportes antes de entregarlos al cliente. Además de solicitar estadísticas de vulnerabilidades que aportan datos para el laboratorio LabSI.
Cliente	Realiza las solicitudes de las pruebas. Puede buscar su proyecto y consultar las vulnerabilidades encontradas a su proyecto una vez empezadas las pruebas.

Tabla 4. Trabajadores del Negocio

2.7.2 Diagrama de Casos de Uso del Sistema.

Los Diagramas de Caso de Uso modelan la vista de los Casos de Uso del Sistema, es decir modelan el contexto del sistema, subsistema o clase, y el modelado de los requisitos para mostrar cómo reacciona el sistema a situaciones que se produzcan. Una vez confeccionado el diagrama de casos de uso del negocio y definidos los requisitos funcionales, se obtuvo el siguiente Diagrama de Caso de Uso del Sistema. (33)

DIAGRAMA DE CASOS DE USO DEL SISTEMA

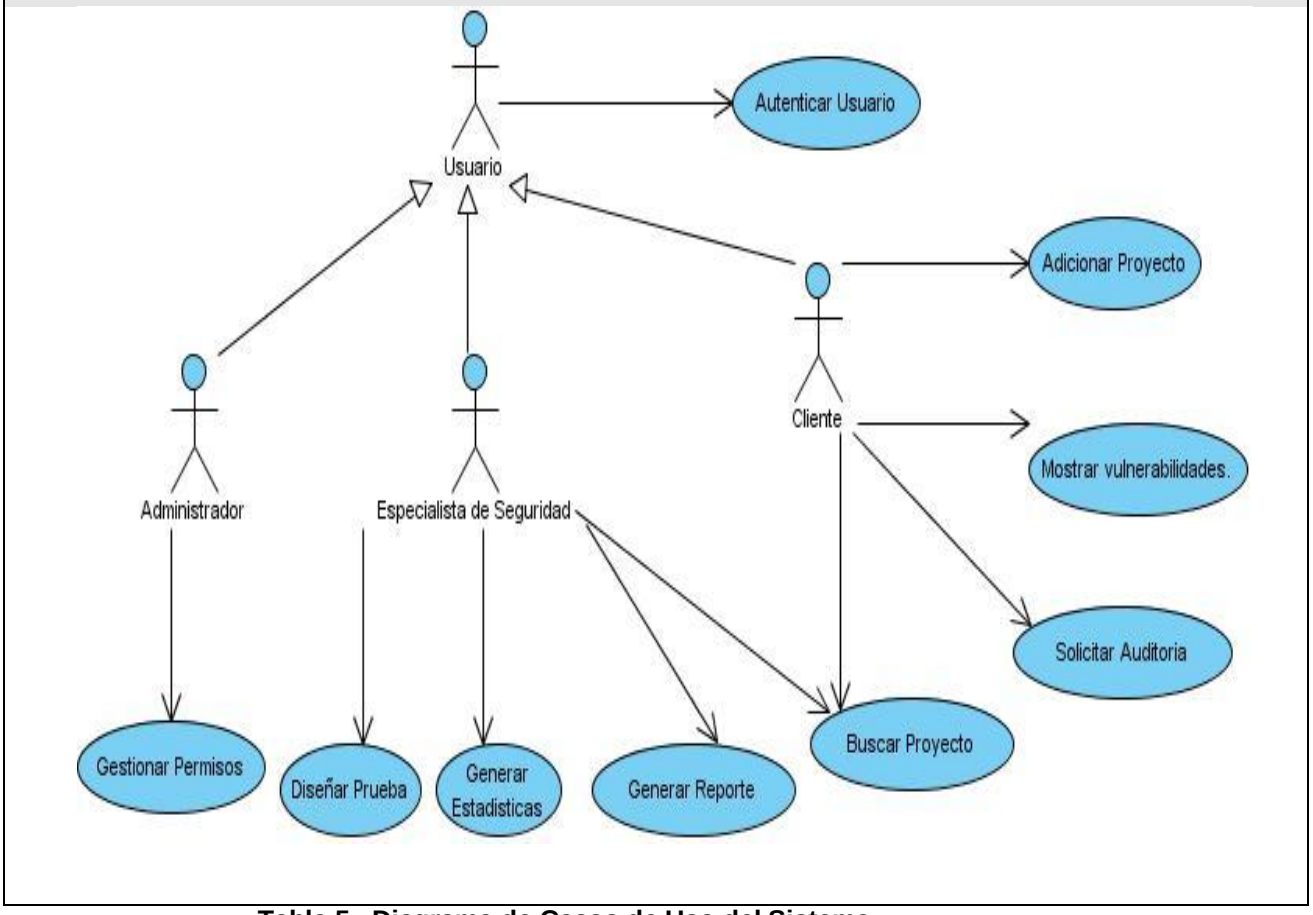

Tabla 5. Diagrama de Casos de Uso del Sistema.

2.7.3 Descripción de Casos de Uso.

Los casos de uso describen en forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista de un usuario, permiten definir los límites del sistema y las relaciones entre la aplicación y el entorno. Es una especificación simple y consistente de cómo interactúan los actores y los casos de uso (el sistema). Se concentra en el comportamiento externo del sistema e ignora cómo se hacen realmente las cosas dentro del sistema. El lenguaje y la terminología son los mismos que los usados por el cliente/usuario del sistema (33)

A continuación la descripción de los casos de uso del sistema:

Caso de Uso:	Diseñar Pruebas
Actores:	Especialista de Seguridad

Resumen:	El objetivo de este caso de uso es diseñar las pruebas que se le realizarán al sistema a partir de sus características y con las herramientas necesarias.
Precondiciones:	Debe estar realizada la solicitud de pruebas.
Referencias	RF6
Prioridad	Alta

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista selecciona la opción de diseñar pruebas.	2. Se muestra la lista de proyectos que se encuentran registrados en el sistema.
3. Selecciona el proyecto al que se le realizará la prueba y se presiona el botón generar lista de chequeo.	4. Genera la lista de chequeo automáticamente con un conjunto de pruebas y sus herramientas correspondientes.
Prototipo de Interfaz	
Flujo alterno "Pruebas Manuales"	
5. Si selecciona la opción adicionar pruebas de forma Manual.	6. Muestra un listado de pruebas que podrá agregar.
7. Selecciona las pruebas que considera necesarias y selecciona el botón de Adicionar.	8. Las pruebas seleccionadas se agregan a la lista de pruebas ya diseñadas.
	9. Se muestra en mensaje "Se ha añadido correctamente la prueba".
Prototipo de Interfaz	
Flujo alterno	
Acción del Actor	**Respuesta del Sistema**

1. Selecciona el botón Generar lista de chequeo sin haber seleccionado el proyecto al que se le realizará la prueba.	2. Se muestra un mensaje de error "Debe seleccionar el proyecto".

Tabla 6. Diseñar Pruebas

Caso de Uso:	**Generar Reporte**
Actores:	Especialista De Seguridad
Resumen:	El caso de uso inicia cuando se requiere entregar al cliente un informe final con todas las vulnerabilidades encontradas en su sistema una vez terminada las pruebas.
Precondiciones:	Se deben haber terminado las pruebas.
Referencias	RF7, RF8
Prioridad	Alta

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción de generar reporte.	2. Muestra una interfaz donde se selecciona el nombre de todos los proyectos registrados en el sistema.
3. Selecciona el proyecto y el sistema al que se le generara el reporte final y presiona el botón Generar Reporte.	4. Muestra las vulnerabilidades del sistema seleccionado.
Prototipo Interfaz	

Flujo Alterno "Adicionar Vulnerabilidades "	
5. Selecciona adicionar nuevas vulnerabilidades.	6. Muestra los campos para adicionar vulnerabilidades.
7. Adiciona las vulnerabilidades al sistema.	

Flujo alterno "Vulnerabilidades Existentes"	
Acción del Actor	**Respuesta del Sistema**
1. Agrega vulnerabilidades que ya se generaron en el reporte.	2. Muestra en mensaje de error.
3. Retorna al paso 5 de la sección Adicionar Vulnerabilidades.	4. Termina el caso de uso.

Tabla 7. Generar Reporte

Caso de Uso:	**Generar Estadísticas**
Actores:	Especialista De Seguridad
Resumen:	El caso de uso inicia cuando se solicita un informe estadístico de vulnerabilidades. Estas son generadas y calculadas a partir de los archivos existentes en la base de datos. El caso de uso termina cuando se generan las estadísticas.
Precondiciones:	Que se hayan realizado pruebas.
Referencias	RF11

Prioridad	Media
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción de generar estadísticas.	2. El sistema muestra la interfaz para Generar estadísticas.
3. Escoge la estadística deseada. • Estadísticas por periodo de tiempo • Estadísticas por proyectos	4. Si selecciona la opción Estadísticas por periodo de tiempo
Sección "Estadísticas por periodo de tiempo"	
3. Selecciona el intervalo de fecha deseado.	4. Muestra las estadísticas de vulnerabilidades en el periodo de tiempo especificado.
Sección "Estadísticas por proyecto"	
5. Selecciona el proyecto deseado.	6. Muestra las estadísticas de vulnerabilidades del proyecto seleccionado.
	7. Termina el caso de uso.

Tabla 8. Generar Estadísticas

Caso de Uso:	**Autenticar Usuario**
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario debe autenticarse para acceder al sistema después de tener asignados privilegio de acceso al mismo, se comprobarán sus datos. El caso de uso termina con la entrada del usuario al sistema.
Precondiciones:	El usuario debe tener permisos de acceso.
Referencias	RF1
Prioridad	Alta.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. Accede a la interfaz del sistema para autenticarse.	2. Se muestra un formulario para que el usuario entre los datos.
2. Ingresa las credenciales pedidas en el formulario de autenticación (Usuario y Contraseña).	3. Comprueba los datos y lo autentica en el sistema.
	4. Muestra la aplicación.
	5. Termina el caso de uso.

Prototipo Interfaz

Flujo alternativo "Campos vacios"

Acción del Actor	Respuesta del Sistema
	1. Existen campos vacios; se muestra un mensaje de error
2. Retorna al paso 1 del flujo normal de eventos.	
3. Termina el caso de uso.	

Prototipo de Interfaz

Flujo alterno "Datos incorrectos"

Acción del Actor	**Respuesta del Sistema**
1. Accede a la interfaz del sistema para autenticarse.	2. Se muestra un formulario para que el usuario entre los datos.
3. Ingresa las credenciales pedidas en el formulario de autenticación (Usuario y Contraseña).	4. Comprueba los datos del actor y lo autentica en el sistema.
	5. Muestra un mensaje diciendo que los datos son incorrectos.

Prototipo de Interfaz

Tabla 9. Autenticar Usuario

Caso de Uso:	**Adicionar Proyecto**
Actores:	Líder
Resumen:	El CU inicia cuando el usuario selecciona la opción de adicionar proyecto e ingresa los datos de su proyecto. El caso de uso termina cuando se adiciona la información del proyecto.
Precondiciones:	El líder debe haberse registrado.
Referencias	RF5.
Prioridad	Media.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción Adicionar Proyecto.	2. Se muestra una interfaz para que el usuario llene los campos requeridos nombre del proyecto, Centro, líder y nivel de seguridad que necesita.
2. Selecciona el botón Adicionar.	3. Muestra un cuadro de dialogo indicando que “El proyecto ha sido Adicionado correctamente”.

	4. Termina el caso de uso.

Prototipo Interfaz

Flujo alternativo "Campos vacios"

Acción del Actor	Respuesta del Sistema
1. Selecciona la opción Adicionar Proyecto.	2. Se muestra una interfaz para que el usuario llene los campos que se piden del proyecto.
	3. Muestra "Existen campos requeridos".
2. Retorna al paso 1 del flujo normal de eventos.	
3. Termina el caso de uso.	

Prototipo de Interfaz

Tabla 28. Descripción de casos de uso (Adicionar Proyecto)

En el *Anexo1* se continúa la descripción de los Casos de Uso del Sistema definidos en el diagrama.

2.8 Conclusiones

Durante este capítulo se aborda lo relacionado con la descripción de la solución propuesta para resolver el problema, así como las características que debe cumplir sistema, así como sus principales funcionalidades, con el objetivo de brindar una mayor comprensión del proceso y contar con la base necesaria para la realización del diseño del sistema a realizar.

CAPÍTULO III: DISEÑO DEL SISTEMA

3.1 Introducción.

En este capítulo se hace referencia a los aspectos correspondientes al diseño, en función de la propuesta del sistema y teniendo en cuenta las funcionalidades seleccionadas. Se define la arquitectura para la implementación y se especifican los patrones de diseño y son confeccionados los diagramas de clases del diseño además se realiza la propuesta de interfaz de usuario para el sistema.

3.2 Patrón de arquitectura.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (34)

3.2.1 Modelo-Vista-Controlador (MVC)

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos (34):

- Modelo: encapsula los datos y la funcionalidad de la aplicación.
- Vista: maneja la visualización de la información desplegando la información contenida en el modelo (pueden existir varias vistas).
- Controlador: está asociado a cada vista, recibe entradas que traduce en invocaciones de métodos del Modelo o de Vista. El usuario interactúa con el sistema solamente vía controladores.

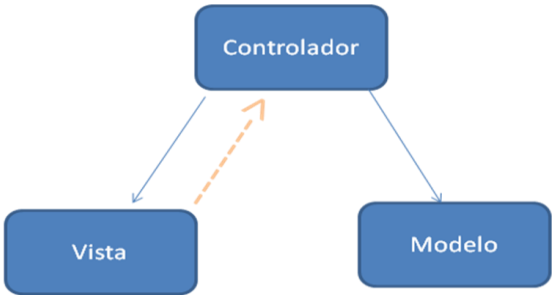

Figura 2. Modelo-Vista-Controlador (MVC)

Este patrón fue seleccionado para tener un mejor soporte de vistas múltiples, dado que la vista se encuentra separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. En este caso múltiples páginas de una aplicación de Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes. También tiene facilidad de adaptación al cambio ya que los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocio. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.
A continuación una muestra de cómo se utiliza el patrón modelo-vista-controlador para la estructuración de la plataforma y sus módulos.

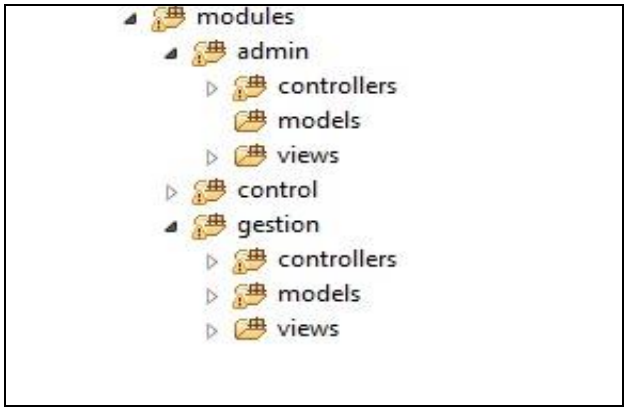

Tabla 11 Patrón Aplicado para la Arquitectura

3.3 Patrón de diseño.

Un patrón de diseño está relacionado con los aspectos del diseño de los subsistemas. Es una solución estándar para un problema común de programación y una técnica para flexibilizar el código. (35)

3.3.1 Singleton

Este patrón fue seleccionado porque se crea a nivel de objetos con el propósito de garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma. Cuando debe haber únicamente una instancia de una clase y debe ser claro su acceso para los clientes. Permite refinamientos en las operaciones y en la representación, mediante la especialización por herencia. (35)

A continuación como está definido este patrón en la aplicación:

```
public static function getInstance()
{
    if ( ! isset(self::$_instance)) {
        self::$_instance = new self();
    }
    return self::$_instance;
}
```

Tabla 12 Definición del patrón Singleton.

```
public function dameaplicacionporproyecto($id_proyecto)
{
    $con = Doctrine_Manager::getInstance ()->getCurrentConnection ();
        $sql = "select id,aplicacion_audit from
                solicitud
                where id_proyecto = '$id_proyecto'";
                $stmt = $con->prepare ( $sql );
        $stmt->execute ();
        $results = $stmt->fetchAll ( PDO::FETCH_ASSOC );
    return $results;
}
```

Tabla 13 Donde se utiliza el patrón Singleton en la implementación del sistema.

3.4 Patrones de GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP) una de las cosas más complicadas es el diseño orientado a objetos, consiste en elegir las clases adecuadas y decidir como estas clases deben interactuar. Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Para el desarrollo del sistema fueron seleccionados los siguientes patrones: (36)

3.4.1 Bajo Acoplamiento

Se utiliza para lograr que existan pocas dependencias entre las clases, deben estar lo menos ligadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima

repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (36)

3.4.2 Alta Cohesión

Se utiliza para garantizar que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Para lograr un buen diseño se deben de crear "paquetes de servicio" o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia). (36)

3.4.3 Experto

Se utiliza como principio básico de asignación de responsabilidades. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento), por lo que los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener. (36)

3.5 Diagrama de Clases del Diseño.

3.5.1 Diagrama de clase de diseño Autenticar Usuario

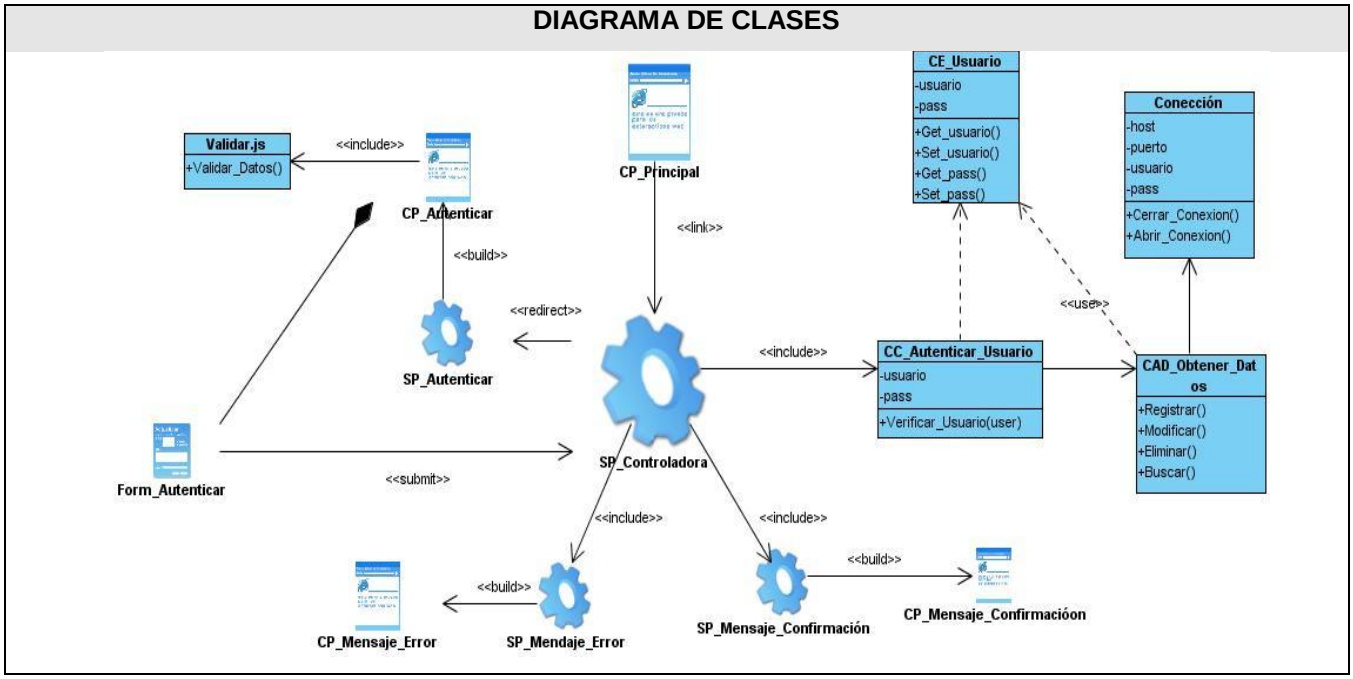

Tabla 14. Diagrama de clase de diseño Autenticar Usuario

3.5.2 Diagrama de clase de diseño Gestionar Permisos a Usuario

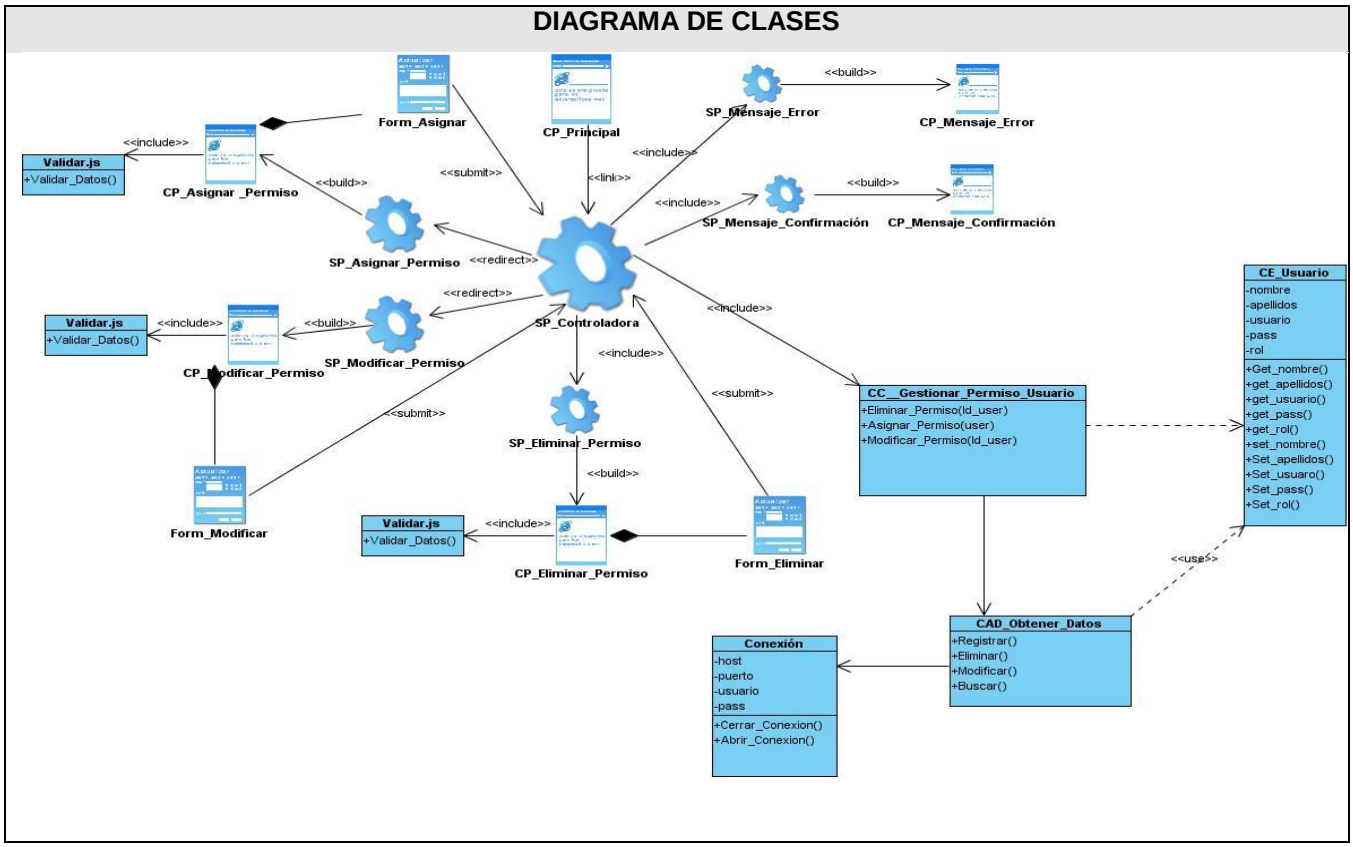

Tabla 15. Diagrama de clase de diseño Gestionar Permisos a Usuario

3.5.3 Diagrama de clase de diseño Solicitar Auditoria

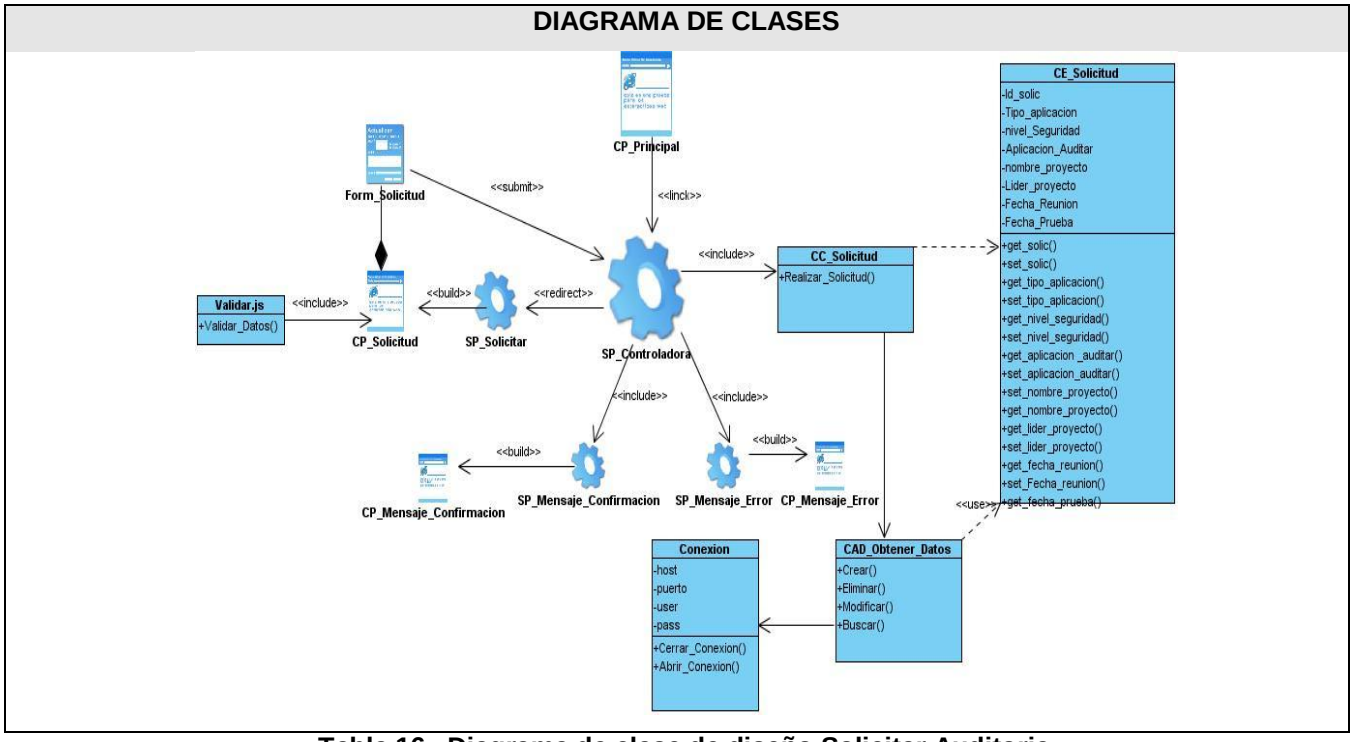

Tabla 16. Diagrama de clase de diseño Solicitar Auditoria

3.5.4 Diagrama de clase de diseño Buscar Proyecto

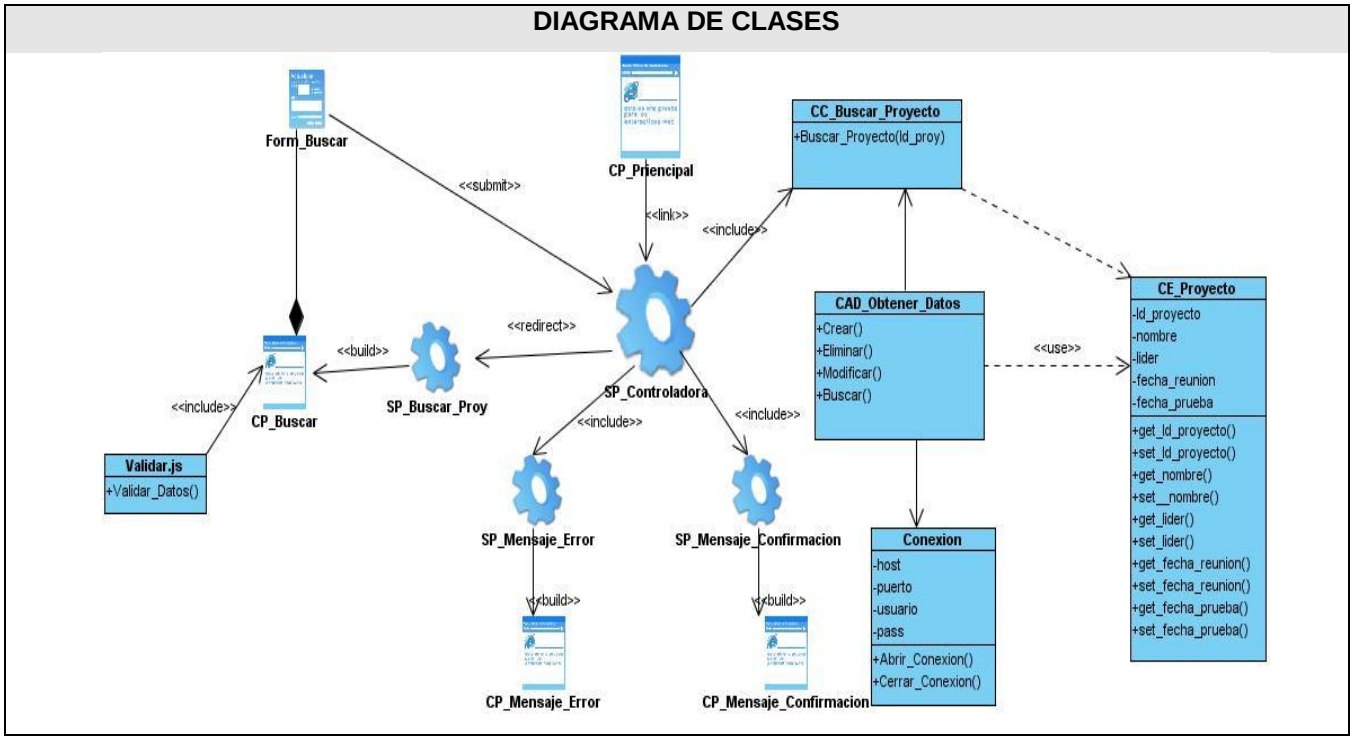

Figura 17. Diagrama de clase de diseño Buscar Proyecto

3.5.5 Diagrama de clase de diseño Diseñar Prueba

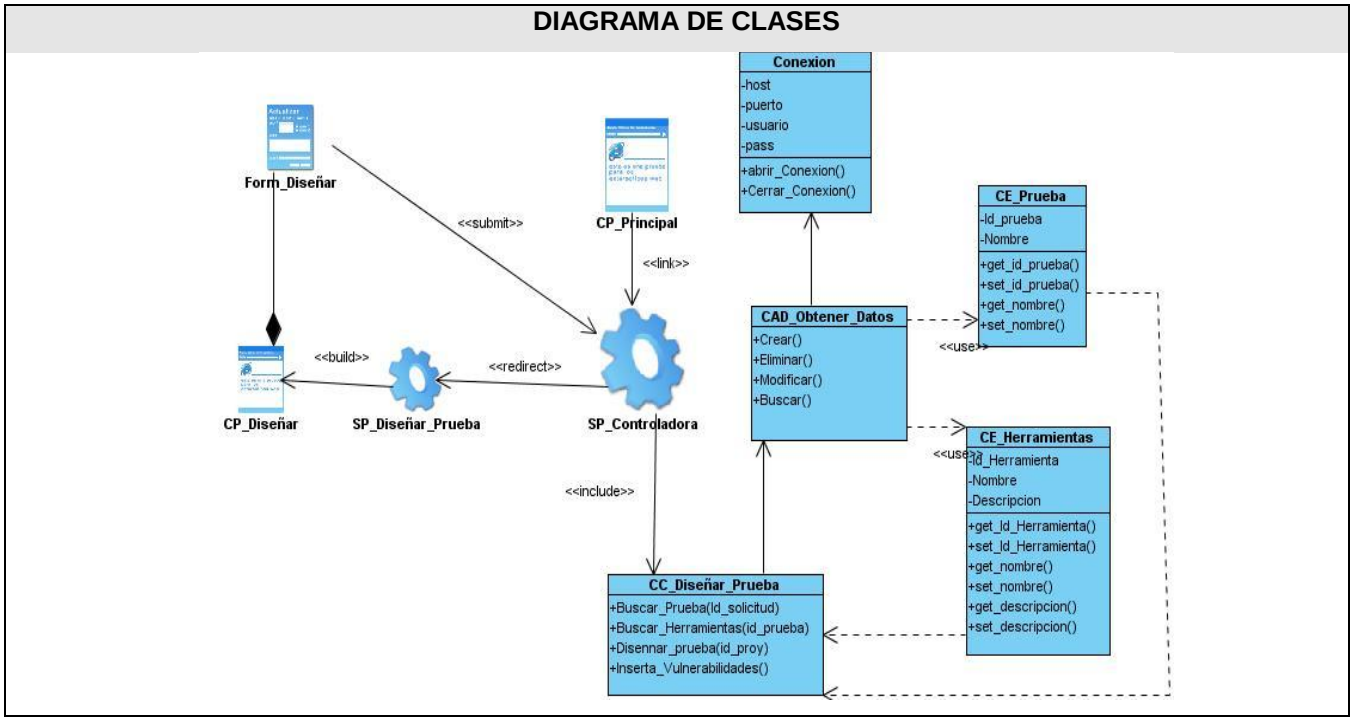

Tabla 18. Diagrama de clase de diseño Diseñar Prueba

3.5.6 Diagrama de clase de diseño Generar Reporte

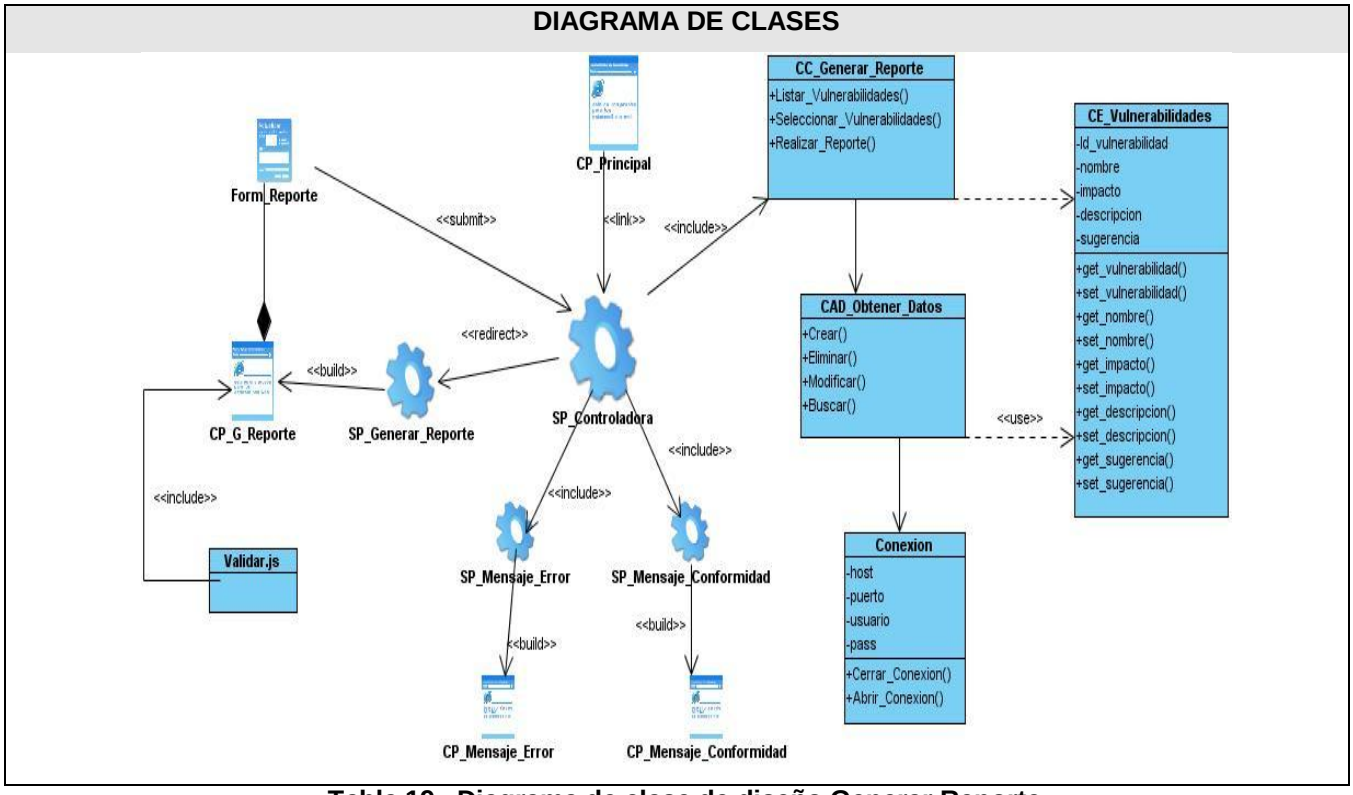

Tabla 19. Diagrama de clase de diseño Generar Reporte

3.5.7 Diagrama de clase de diseño Mostrar Vulnerabilidades

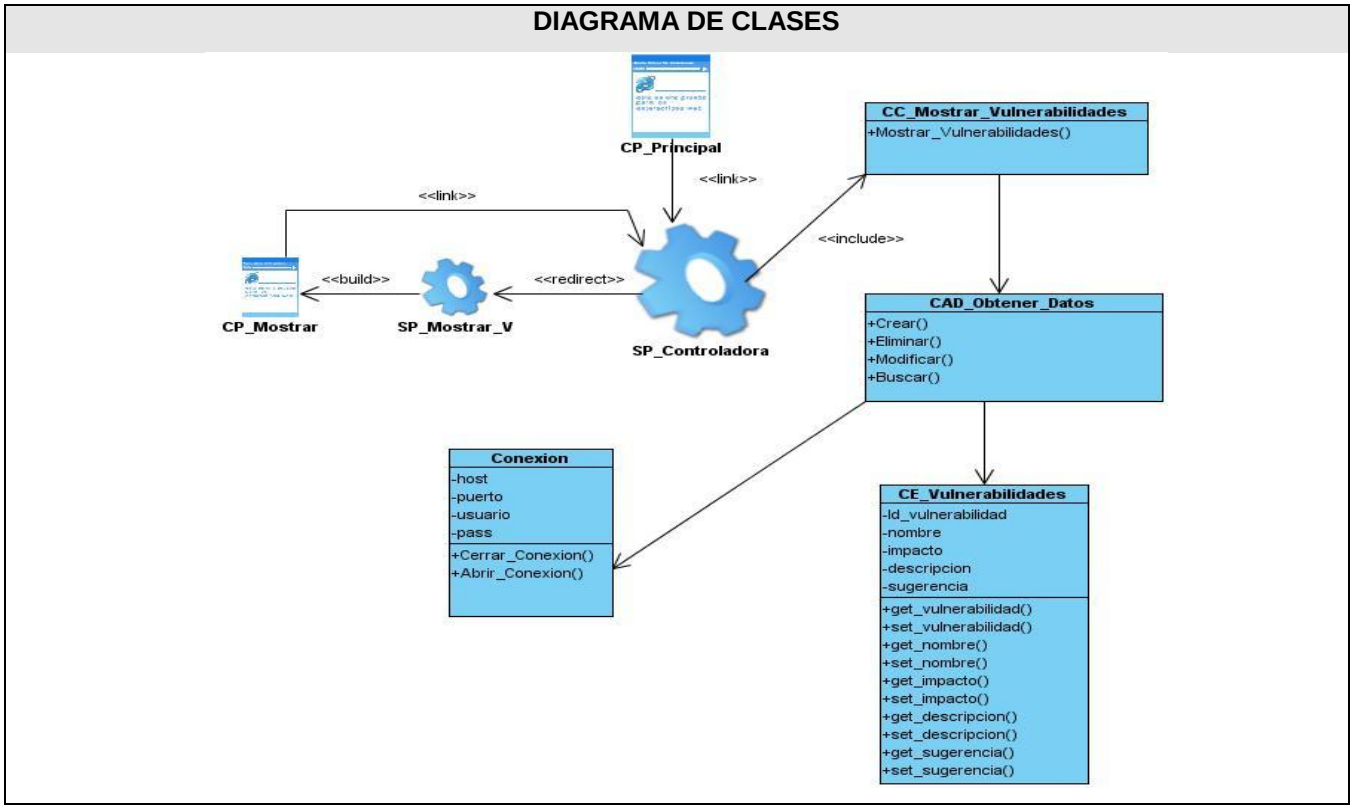

Tabla 20. Diagrama de clase de diseño Mostrar Vulnerabilidades

3.5.8 Diagrama de clase de diseño Generar Estadísticas

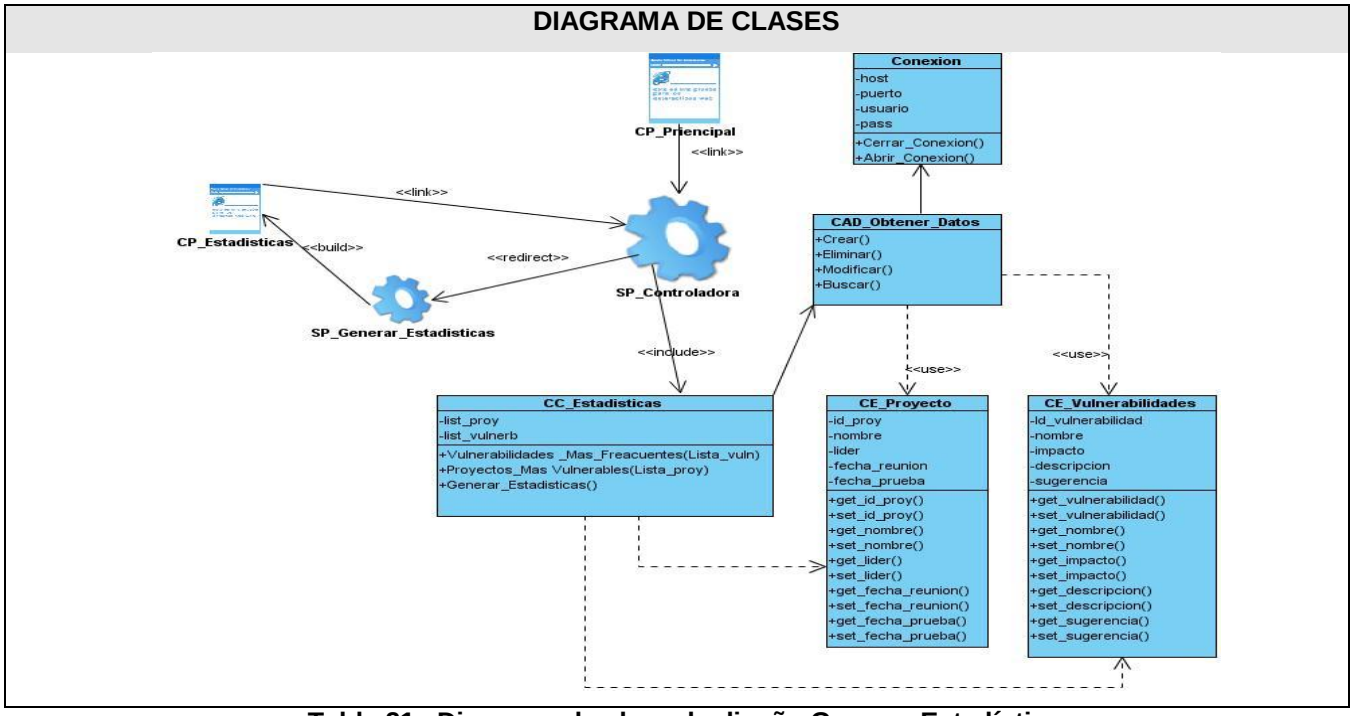

Tabla 21. Diagrama de clase de diseño Generar Estadística

3.6 Diseño de la Base de Datos.

3.6.1 Modelo lógico de la Base de Datos.

Modelo lógico de la Base de Datos.

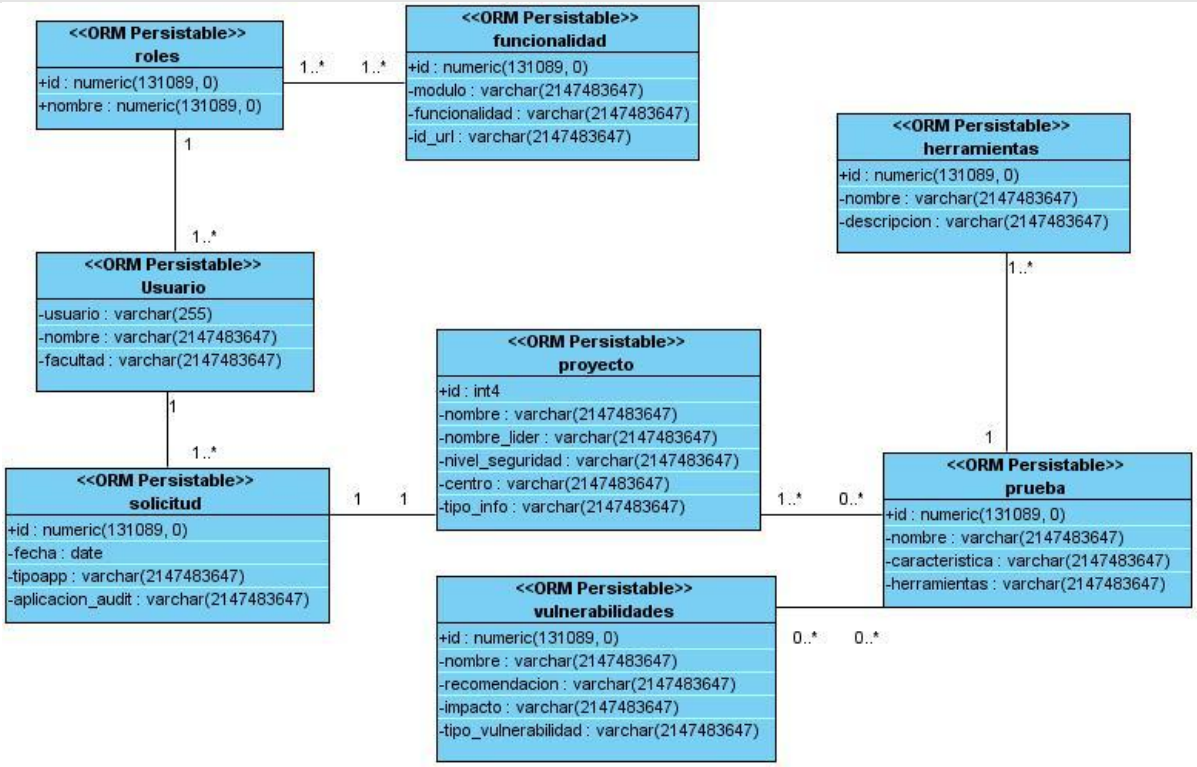

Tabla 22. Modelo lógico de la Base de Datos.

3.6.2 Modelo Físico de la Base de Datos.

Modelo físico de la Base de Datos.

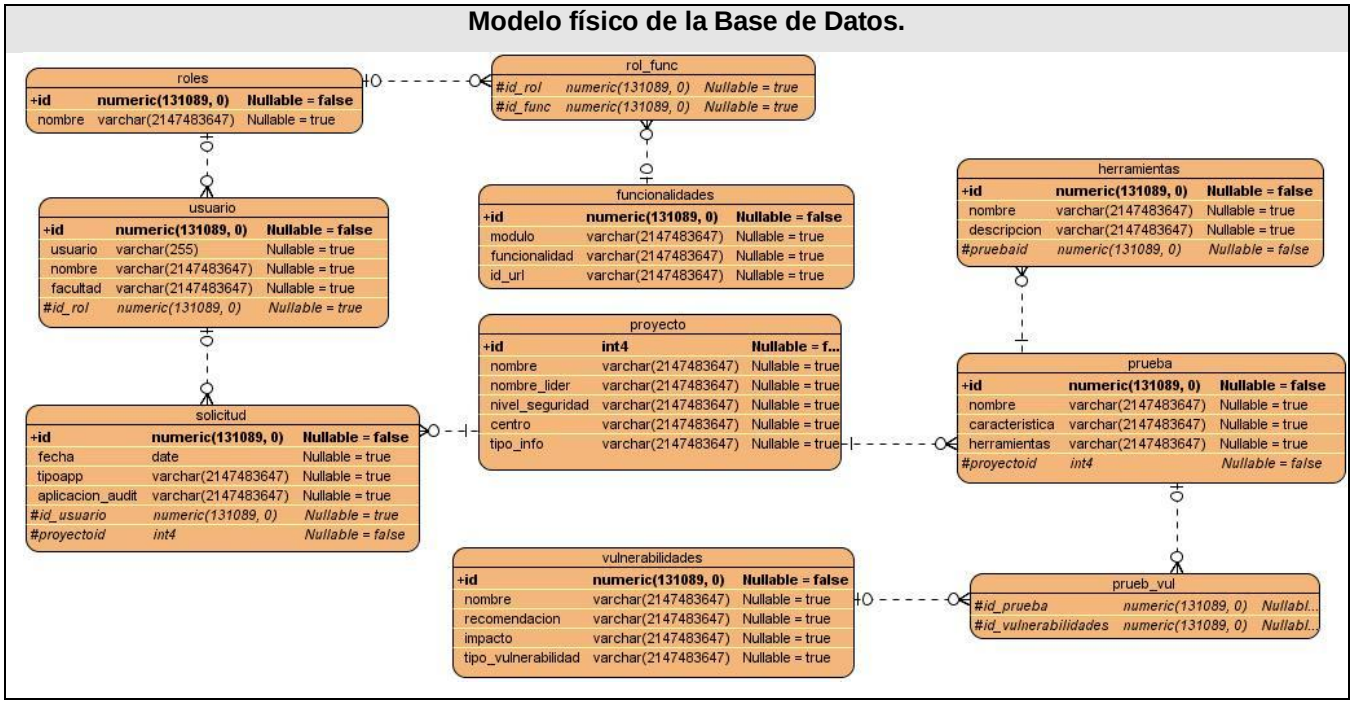

Tabla 23. Modelo físico de la Base de Datos.

3.7 Conclusiones.

Como parte del desarrollo de este capítulo se realizó la descripción del diseño de la plataforma a partir de las funcionalidades definidas. Fueron definidos los patrones de arquitectura y diseño para realizar el sistema, se realizaron los diagramas de clase de diseño y el diseño lógico y físico de la base de datos.

CAPÍTULO IV: IMPLEMENTACION Y PRUEBA

4.1 Introducción

Una vez obtenido el diseño en el capitulo anterior, la implementación se simplifica a implementar el sistema en término de componentes, es decir, ejecutables, código fuente y scripts. Aquí se desarrolla la arquitectura y el sistema como un todo. Además se implementan las clases del diseño como componentes que contienen código fuente. Se presentan los diagramas de componentes y despliegue los que se conforman el Modelo de Implementación.

4.2 Diagramas de Componentes

El modelo de implementación describe cómo los elementos del modelo de diseño (las clases), se implementan en términos de componentes. También estructuran el modelo de implementación en términos de subsistemas de implementación y muestra las relaciones entre los elementos de implementación. Estos diagramas muestran la estructura de alto nivel del modelo de implementación. Representan las dependencias entre componentes de software, incluyendo componentes de código fuente, de código binario y ejecutable. (24)

A continuación el diagramas de componentes del sistema.

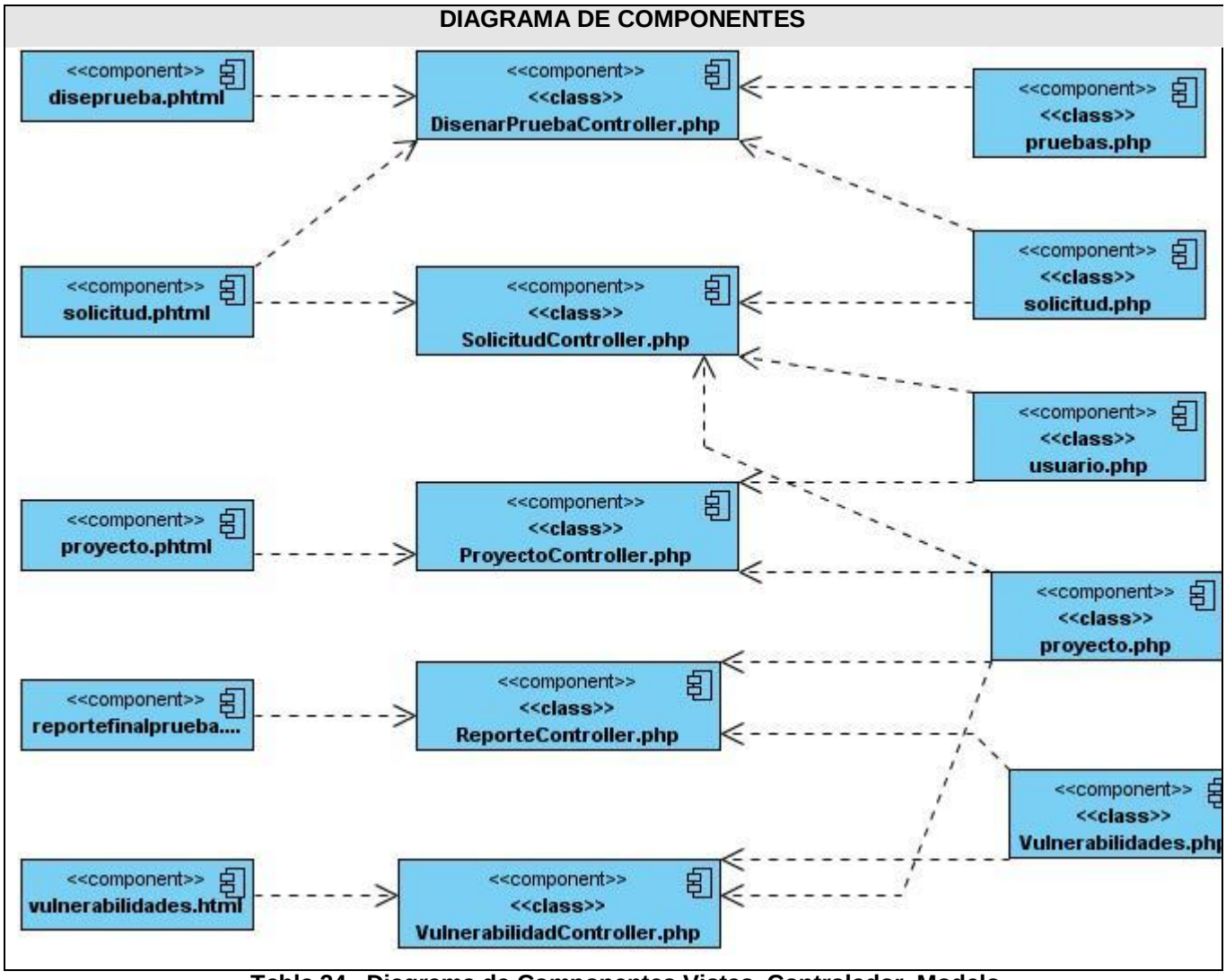

Tabla 24. Diagrama de Componentes Vistas, Controlador, Modelo.

4.2.1 Descripción de Componentes del sistema.

Vistas: Son las clases donde se realiza la implementación de las interfaces del sistema las que se relacionan directamente con el usuario.

- diseprueba.phtml
- solicitud.phtml
- buscar.phtml
- reportefinalprueba.phtml

- vulnerabilidades.phtml

Controladores: Son las clases que atienden las peticiones que hace el usuario a través de las vistas y las envía al modelo o clases manager para que este ejecute las funcionalidades y haga las peticiones a la base de datos.

- DisenarPuebaController.php
- SolicitudController.php
- ProyectoController.php
- Reporte Controller.php
- VulnerabilidadController.php

Modelos: En estas clases se implementan las funcionalidades referidas en los requerimientos del sistema que son ejecutadas en dependencia de las peticiones formuladas desde la vista y responde a instrucciones de cambiar el estado desde el controlador.

- prueba.php
- solicitud.php
- proyecto.php
- vulnerabilidades.php
- usuario.php

4.3 Diagrama de Despliegue

El diagrama de Despliegue describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Mostrando así las relaciones físicas entre los componentes de hardware y de software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software, que no son más, que los procesos que se ejecutan en ellos. A continuación el diagrama de despliegue de la Plataforma de Gestión de Seguridad Informática.

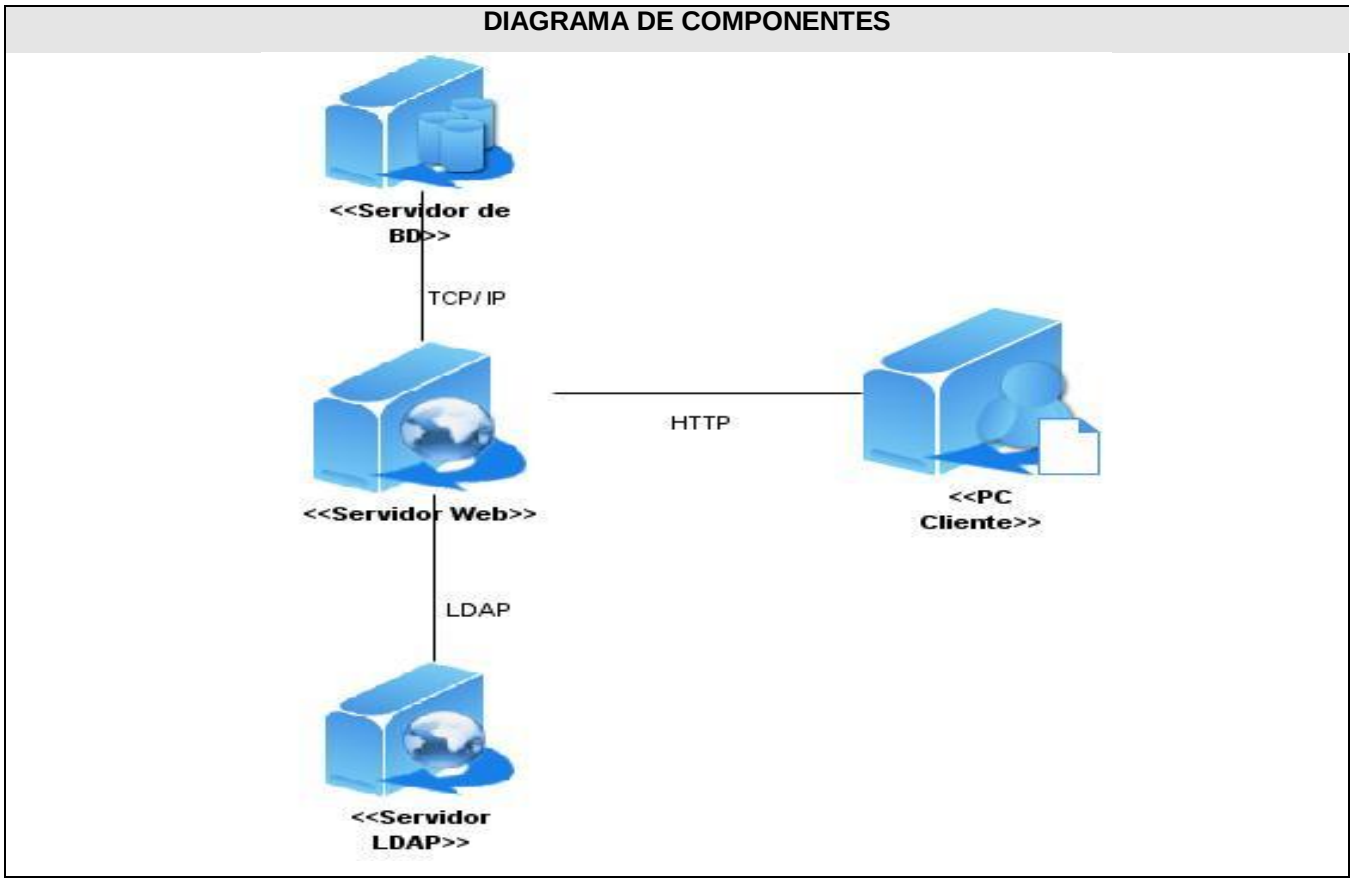

Tabla 25. Diagrama de Despliegue.

4.3 Pruebas al Software

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas para la garantía de la calidad. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Además, se verifica la interacción de componentes, que todos los requisitos se han implementado correctamente. También identifica y asegura que los defectos identificados se han rectificado antes de entregar la aplicación al cliente. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces. (37)

4.3.1 Pruebas de Caja Negra

Las pruebas de caja negra o pruebas funcionales se centran en las funcionalidades que se espera que el sistema debe realizar; el probador le suministra datos de entrada y espera que el sistema los procesé sin importar lo que pueda estar ejecutando por dentro, luego estudia los resultados obtenidos de las prueba y los evalúa. Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (general: teclado, pantalla, ficheros, canales de comunicaciones, etc.)(38)

4.3.1.1 Casos de Pruebas

- **Caso de Uso Buscar Proyecto**

Escenario	Descripción	Proyecto	Respuesta del sistema	Flujo central
EC 1.1 Proyecto seleccionado	El usuario selecciona el proyecto que quiere mostrar.	Montetra	Se muestran los datos del proyecto que ha seleccionado.	1. El usuario selecciona el proyecto que quiere mostrar. 2. El usuario selecciona la opción mostrar. 3. El sistema muestra la interfaz con los datos del proyecto seleccionado.
EC 1.2 Proyecto no seleccionado	El usuario no selecciona el proyecto que quiere mostrar.		Se muestra un cuadro de dialogo indicando que el proyecto debe ser seleccionado.	1. El usuario selecciona la opción mostrar. 2. El sistema muestra un cuadro de diálogo “Debe seleccionar un proyecto.”

Tabla 26. Caso de Prueba (Buscar Proyecto).

4.3.1.2 Resultado de la Prueba de Caja Negra.

Entorno de Pruebas

Los resultados se obtuvieron en el siguiente escenario:

Microprocesador	Core Duo
RAM	1 Gb
Disco Duro	120 Gb
Sistema Operativo	Windows XP

Después de haber realizado las pruebas de funcionalidades y analizado los casos de prueba se obtuvieron los siguientes resultados.

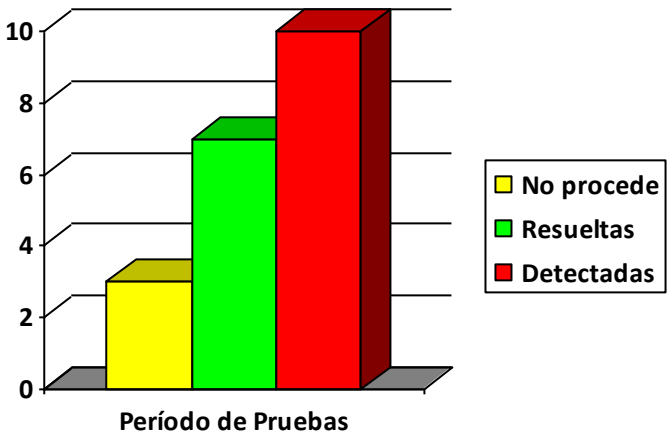

Tabla 27. Resultados esperados de las pruebas

4.3.2 Pruebas de Integración

Las pruebas de integración se realizan durante la construcción del sistema, englobando a todos los módulos que en él se implementan y terminan probando el sistema como conjunto. Las pruebas finales de integración cubren todo el sistema y la especificación de requisitos del usuario. (37)

En esta plataforma se han implementado dos módulos, el de administración y el de gestión de pruebas, y se ha comprobado que al integrarlos funcionen como un solo sistemas correctamente para lo que fueron implementados.

4.3.3 Pruebas de Aceptación del Cliente

En las pruebas de aceptación al cliente se realizan un conjunto de pruebas funcionales sobre el sistema y buscan un apoyo de la especificación de requisitos y en el manual del usuario. (38) Al sistema se le realizaron estas pruebas de aceptación, se presentó para ser aceptado por el cliente, en este caso los especialistas de LabSI, se generó de este encuentro algunas recomendaciones relacionadas con:

- Los colores de las interfaces deben ser más apropiados.

- La combinación de colores entre las interfaces y el banner
- Se le agregará al caso de uso control de acceso una funcionalidad para que pudiera cambiar un usuario de rol.

4.4 Conclusiones

En este capítulo se realizó la implementación de la Plataforma de Seguridad Informática, Modulo pruebas de Intrusión, y se validaron las funcionalidades implementadas, a través de las pruebas de caja negra y caja blanca. Además se manifestó de forma exitosa como las funcionalidades respondieron a los requisitos funcionales detallados. La etapa de pruebas se ejecutó iterativamente respondiendo a las necesidades actuales de los usuarios y las peticiones de los clientes.

Recomendaciones

CONCLUSIONES GENERALES

Durante esta investigación se realizo un profundo análisis, investigación y descripción del proceso de pruebas de intrusión y herramientas utilizadas para realizar estas pruebas en LabSI, así como herramientas de gestión de pruebas existentes.

Se hizo un profundo análisis y comprensión del estilo arquitectónico y los patrones de diseño utilizados.

Se realizo el diseño e implementación del sistema logrando así obtener una aplicación funcional que responde a cada uno de los requerimientos y objetivos planteados por el usuario.

Se obtuvo como resultado una aplicación con las siguientes funcionalidades:

- Adicionar proyectos que solicitan el servicio de pruebas de seguridad.
- Buscar Proyecto y obtener las vulnerabilidades encontradas a sus aplicaciones.
- Realizar solicitudes de pruebas a las aplicaciones.
- Diseñar pruebas con las herramientas correspondientes a cada una de ellas.
- Realizar un reporte de cada proyecto a los que se han realizado las pruebas.
- Calcular cantidad de vulnerabilidades en un periodo de tiempo determinado y asociado en específico a un proyecto.

Para obtener estos resultados se utilizaron diferentes herramientas que ayudaron a la implementación de la aplicación así como Rup como metodología de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta de modelado. También PHP como lenguaje de programación y Zen Studio como IDE de desarrollo.

Recomendaciones

RECOMENDACIONES

- Se recomienda terminar la implementación del módulo Ejecución y Control de Herramientas para que se realice la integración de estos a la Plataforma de Seguridad Informática, y así lograr mejor eficiencia en el Laboratorio de Seguridad Informática (LabSI).
- Se recomienda la utilización de esta aplicación en el laboratorio de seguridad para que sea más rápido y eficiente el proceso de gestión de pruebas y obtener mejores resultados en LabSI.
- Se recomienda agregar a la aplicación la funcionalidad de gestionar las pruebas de intrusión.

Bibliografía

REFERENCIAS BIBLIOGRAFICAS

1. http://definicion.de/seguridad-informatica/. *[Citado: 5 ,11 ,2010.]*
2. *owasp, Sitio Oficial de OWASP. [En línea] 2006. [Citado: 8 de febrero de 2011.]* www.owasp.org.
3. http://www.netzweb.net/html/print/segurid/det_int.pdf. *[Citado: 5 ,11 ,2010.]*
4. http://es.testhouse.net/index.php. *[Citado: 7 ,11 ,2010.]*
5. http://protegete.jccm.es/protegete/opencms/Pymes/Seguridad/Amenazas/herramientas.html. [Citado: 20 ,2 ,2011.]
6. http://mtesauro.com/livecd/index.php?title=Main_Page. *[Citado: 7 ,11 ,2010.]*
7. http://www.backtrack-linux.org/. *[Citado: 10 ,11 ,2010.]*
8. 8.http://www.dragonjar.org/samurai-entorno-de-trabajo-para-el-testing-de-seguridad-a-aplicativos-web.xhtml. *[Citado: 17,2 ,2011.]*
9. *El Análisis de Anomalías detectadas durante las pruebas de Software. Peralta, Ing. Ramón Enrique Gonzáles.*
10. *http://www.cirt.net/code/nikto.shtml. [Citado:25 ,2 ,2011.]*
11. *http://www.ict-romulus.eu/web/wapiti. [Citado: 27 ,2 ,2011.]*
12. *http://www.sensepost.com/research/wikto/. [Citado: 2 ,3 ,2011.]*
13. http://ncsgames.mforos.com/1275323/9060204-escaneado-de-puertos-e-identificacion-de-servicios/.*[Citado: 15,11 ,2010.]*
14. http://unixlandia.com/web_interna/index.php?option=com_content&view=article&id=435:escaneador-de-puertos-nmap&catid=52:generales&Itemid=32. *[Citado: 20 ,11 ,2010.]*
15. http://labitacoradegabriel.wordpress.com/2010/05/28/instalando-openvas/. *[Citado: 20 ,11 ,2010.]*
16. http://www.nessus.org/nessus/perimeter/. *[Citado: 25 ,11 ,2010.]*
17. *GUIA DE PRUEBAS OWASP. [Citado: 27 ,11 ,2010.]*
18. http://www.dragonjar.org/sqlmap-herramienta-automatica-de-inyeccion-sql.xhtml. *[Citado: 5 ,12 ,2010.]*
19. http://translate.google.com/translate?hl=es&langpair=en|es&u=http://www.darknet.org.uk/2007/06/sqlbrute-sql-injection-brute-force-tool/. *[Citado: 5 ,12 ,2010.]*
20. *Herzog, Creado por Pete. Manual de la Metodología Abierta de Testeo de Seguridad. [Citado: 10 ,2 ,2011.]*

Bibliografía

21. http://translate.googleusercontent.com/translate_c?hl=es&langpair=en|es&u=http://alienvault.com/community&rurl=translate.google.com&usg=ALkJrhi2A4icC_1rMMrHLIm3FbY7KseSog. *[Citado: 8 ,12 ,2010.]*
22. *Yanira Pantin Kindelán, Humberto Arencibia Lorenzo. Sistema de Gestión de Reportes de vulnerabilidades para el proyecto LABSI. [Citado: 10 ,12 ,2010.]*
23. *Conferencia 2: Modelos de proceso de desarrollo de software. Metodologías de desarrollo de software y lenguajes de modelado. http://eva.uci.cu/course/view.php?id=102. [Citado: 12 ,12 ,2010.]*
24. *Conferencia 3: Metodología de desarrollo RUP. Lenguaje de modelado UML. http://eva.uci.cu/course/view.php?id=102. [Citado: 12 ,12 ,2010.]*
25. *http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p /. [Citado: 2 ,3 ,2011.]*
26. http://eva.uci.cu/course/view.php?id=452. Introducción al lenguaje PHP.*[Citado: 13 ,12 ,2010.]*
27. http://framework.zend.com. *[Citado: 14 ,12 ,2010.]*
28. PostgreSQL. [En línea] 2011. [Citado el: 15 de 05 de 2011.] *http://www.postgresql.org/*.
29. *http://www.zend.com/en/products/studio/ . [Citado: 2 ,5 ,2011.]*
30. http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/urbina_p_j/capitulo2.pdf *[Citado: 2 ,5 ,2011.]*
31. *http://es.scribd.com/doc/13500172/actividad2-diagrama-de-casos-de-uso-del-negocio-y-del-sistema. [Citado: 1 ,2 ,2011.]*
32. http://zoftwar.blogspot.com/2006/11/requisitos-no-funcionales-parte-1-con.html. *[Citado: 9 ,1 ,2011.]*
33. *Programación Orientada A Objetos Máster De Computación. II.2 UML: Modelado de casos de uso. Dr. Elena Mediavilla. [Citado: 15 ,2 ,2011.]*
34. *Oktaba, Hanna. Introducción a Patrones. Introducción a Patrones. [Online] [Citado: 9 ,3 ,2011.]* http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.
35. *Gracia, Joaquin. Patrones de diseño. 2005.*
36. *Creador y propietario de AdictosAlTrabajo.com, Director General de Autentia S.L. Ingeniero Técnico de Telecomunicaciones y Executive MBA por el Instituto de Empresa 2007. Twitter:*

@rcanalesmora. Autor del Libro: Informática profesional, las reglas no escritas para triunfar en la empresa. [Citado: 11 ,2 ,2011.]

37. http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm [Citado: 24 de 05 de 2011.]

BIBLIOGRAFÍA

1. http://www.adictosaltrabajo.com/tutoriales/pdfs/grasp.pdf
2. http://pub.ufasta.edu.ar/fim28/files2005/FIM28%20AyD%20I%20Dominio%20y%20GRASP.pdf
3. *Wilson, Rasmus Lerdorf, Zeev Suraski, Andrei Zmievski, Jouni Ahto.*
4. *Patrones de Diseño. Daniel Mazzini. dmazzini@ubicasolutions.com*
5. *Patrones GRASP. Diana Paola Hurtado Bustamante. Diana Patricia Gutiérrez Valencia. Juan Pablo Suárez Valencia. Gabriel Asakawa. Eudo Quevedo Pantoja. Universidad del Valle.*
6. *Técnicas y Procedimientos para la realización de Test de Intrusión.SG6 –Soluciones Globales en Seguridad de la Información.http://www.sg6.es*
7. *GUIA DE PRUEBAS OWASP.*
8. *OSSTMM 2.1.Manual de la Metodología Abierta de Testeo de Seguridad. Creado por Pete Herzog. INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES.*
9. *Pentest. Alejandro Ramos. www.securitybydefault.com.*
10. http://url?sa=t&source=web&cd=1&ved=0CBsQFjAA&url=http%3A%2F%2Fwww.monografias.com%2Ftrabajos71%2Fethical-hacking-test-intrusion-metodologias%2Fethical-hacking-test-intrusion-metodologias.shtml&ei=38z3TeveJqH00gGUua21Cw&usg=AFQjCNH9CK8NFmxA6ZfSvEpYO6mDAU5gCA
11. *Pruebas del Software: Descubrir Errores y Más...Javier Tuya. Universidad de Oviedo, Dpto. de Informática. Grupo de Investigación en Ingeniería del Software.*
12. http://www.di.uniovi.es/~tuya/
13. https://seguinfo.wordpress.com/category/manual/
14. http://www-01.ibm.com/software/es/rational/quality/
15. http://www.es.sogeti.com/Que-hacemos/Testing/Testing/
16. https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf
17. http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf
18. http://www.nessus.org
19. RFC 2616: “Hypertext Transfer Protocol -- HTTP/1.1”
20. RFC 2975: “HTTP State Management Mechanism”

Bibliografía

21. Jeremiah Grossman: "Cross Site Tracing (XST)" - http://www.cgisecurity.com/whitehat-mirror/WH-Documento_XST_ebook.pdf
22. Amit Klein: "XS (T) attack variants which can, in some cases, eliminate the need for TRACE" -
23. http://www.securityfocus.com/archive/107/308433
24. *Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado del Software. [En línea]* http://bibliodoc.uci.cu/pdf/8478290362.pdf.
25. http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/urbina_p_j/capitulo2.pdf.
26. *El Análisis de Anomalías detectadas durante las pruebas de Software. Peralta, Ing. Ramón Enrique Gonzáles.*
27. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/capitulo5.pdf
28. **Valdés, Damián Pérez.** Maestros del Web. [En línea]. *http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.*

Glosario de términos

GLOSARIO DE TÉRMINOS

A

- **AppSecTestingChecklist**: Lista de chequeo de pruebas de seguridad.

C

- **Combobox**: Se utiliza para mostrar datos en un cuadro combinado desplegable de forma predeterminada.

E

- **Escáner:** analizador de red.

F

- **Framework**: estructura conceptual y tecnológica de soporte definida, normalmente con artefactos de software concretos, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

H

- **Herramientas CASE**: herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.
- **Hosts**: Es una computadora conectada a una red, que proveen y utilizan servicios de ella.

I

- **Intrusión**: Como intrusión se entiende la realización de un acto no autorizado.

L

- **Livecd de OWASP**: Ofrecer los recursos del OWASP desde uno CD auto arrancable.
- **Livecd Backtrack**: Backtrack le ofrece al usuario una extensa colección de herramientas completamente usables desde un CD.
- **LabSI**: Laboratorio de Seguridad Informática. Se realizan pruebas de seguridad a los sistemas antes de ser entregados al cliente.

N

Glosario de términos

- **Nikto**: Herramienta de seguridad para la detención de vulnerabilidades. **E**s un scanner de código abierto.
- **Nmap**: Es una herramienta para el escaneo de grandes o pequeñas redes.
- **Nessus**: Es un programa de escaneo de vulnerabilidades en diversos sistemas operativos.

O

- **OSSIM:** Open Source Security Information Management. Ofrece todas las funciones necesarias para la gestión de la seguridad en entornos profesionales.
- **OpenVas:** Es una herramienta que permite identificar las vulnerabilidades de un equipo o servidor desde otra PC remota.
- **OWASP:** Proyecto Abierto de Seguridad en Aplicaciones Web. Es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.

P

- **PHP:** Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.
- **Pruebas de Intrusión**: Una prueba de intrusión es el proceso para evaluar la seguridad en sistemas.

R

- **RUP**: Rational Unified Process traducido al español, Proceso Unificado de Desarrollo.
- **Reportes**: Es un documento, generado por un sistema o herramienta, que presenta de manera estructurada y/o resumida datos relevantes.

S

- **Software**: se refiere al equipamiento lógico o soporte lógico de una computadora digital
- **Sqlmap**: es una herramienta realizar inyección de código SQL automáticamente.

Glosario de términos

- **SQL**: Structured Query Language o Lenguaje de consulta estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas.
- **Sqlbrute**: es una herramienta para forzar la salida bruta de datos de bases de datos.

T

- **TCP-IP**: Protocolo de control de transmisión/Protocolo de Internet es un protocolo que sirve para enlazar computadoras que utilizan diferentes sistemas operativos.

U

- **UCI**: Universidad de las Ciencias Informáticas.
- **URL**: Localizador Uniforme de Recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, para su localización.

V

- Vulnerabilidades: hace referencia a una debilidad en un sistema permitiendo a un intruso a acceder a su información.

W

- **Web**: World Wide Web o Red Global Mundial. Es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de Internet.
- **Wapití**: Herramienta de seguridad para la detención de vulnerabilidades. Es un escáner de vulnerabilidades para aplicaciones web.
- **Wikto**: Herramienta de seguridad para la detención de vulnerabilidades. Realiza un escáner en el entorno de un servidor Web para encontrar vulnerabilidades.

ANEXOS

Anexo 1: Descripción de Casos de Uso del Sistema

Caso de Uso:	**Buscar Proyecto**
Actores:	Cliente, Especialista
Resumen:	El CU inicia cuando el usuario selecciona la opción de buscar proyecto y especifica el proyecto que desea buscar. El caso de uso termina cuando se muestra la información del proyecto.
Precondiciones:	El proyecto debe haber sido registrado.
Referencias	RF5.
Prioridad	Media.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción Buscar Proyecto.	2. Se muestra un campo de selección con todos los proyectos registrados en el sistema en caso de ser especialista y con el proyecto que es líder en caso del líder y otro campo de selección que cargara las aplicaciones del proyecto seleccionado.
4. Selecciona el botón buscar.	3. Se muestra el listado de vulnerabilidades encontrados a ese proyecto antes y después de las pruebas.

Tabla 10. Buscar Proyecto

Listado de casos de uso.

CU_1	Gestionar Permiso Usuario
Actor	Administrador
Descripción	En este caso de uso se gestionan los permisos a los usuarios que podrán

	acceder al sistema, se adicionan nuevos permisos, se eliminan y se modifican.
Referencia	RF2.1, RF2.2, RF2.3
CU_2	Solicitar Auditoría
Actor	Cliente
Descripción	El cliente registra todos los datos de su proyecto en una planilla.
Referencia	RF3
CU_3	Modificar Proyecto
Actor	Cliente
Descripción	El cliente puede modificar los datos del proyecto en caso de que exista algún cambio o error.
Referencia	RF_4
CU_6	Mostrar Vulnerabilidades
Actor	Cliente
Descripción	Con este caso de uso el cliente podrá ver las vulnerabilidades que se le han encontrado a su sistema durante la ejecución de las pruebas
Referencia	RF10

Tabla 29. Descripción de casos de uso.

1 Anexo 2: Gestionar Permisos a Usuarios.

2

1 Anexo 3: Adicionar Proyecto.

2

Anexo 3: Modificar Proyecto.

Anexo 4: Privilegios a usuarios por rol.

1 Anexo 5: Realizar Solicitud de Pruebas.

2

3