Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas.

Título: “Diagnóstico de vulnerabilidades de PC Vía Web”

Autores: Yojandy Batista Chillón

Rogfel Thompson Martínez.

Tutor: Alexander Fernández Castro

Co-tutora: Daimara Martínez Borrell

Julio 2008

“Año del 50 Aniversario del Triunfo de la Revolución”

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Rogfel Thompson Martínez

_____

Firma del Autor

Yojandy Batista Chillón

_____

Firma del Autor

Alexander Fernández Castro

_____

Firma del Tutor

Agradecimientos

Le agradecemos a la Revolución y a Fidel por haber creado esta hermosa universidad,
a nuestros padres por ser hijos de ellos,
a nuestros tutores Daimara y Alexander,
a nuestro clientes Raydel y Pablo,
a nuestros amigos.

Dedicatoria

A mis padres por su compañía y confianza durante estos años.

A mi abuela por su preocupación y apoyo.

A toda mi familia por plasmar tantos momentos buenos en mi vida.

A mis amigos especialmente los que me acompañan desde el IPVCE, pues han jugado un papel familiar cuando lo he necesitado.

A todos ellos dedico este trabajo…….

Yojandy Batista Chillón

A mi tía Francisca, la primera en hablarme de la UCI,
A mi mamá, a mi papá, a mi hermana,
A toda mi familia y a mis amigos,
A todos los que me han enseñado algo importante en la vida …

Rogfel Thompson Martínez

Resumen

Una de las principales amenazas de Internet son las vulnerabilidades de los Sistemas Operativos y la no actualización de los parches concernientes a la Seguridad de los mismos. El usuario común desconoce u obvia que su Computadora Personal (PC, *siglas en inglés*) necesita ser diagnosticada seguidamente para determinar sus vulnerabilidades, que estas representan una ventana abierta a cualquier atacante. Los usuarios conectados a la red tanto en nuestra Universidad como en el país no tienen a su disposición una herramienta capaz de brindarle información acerca de las vulnerabilidades que presentan sus máquina así como los posibles parches que puedan solucionarlas; por lo que es necesario, para mantener una buena seguridad informática en nuestro centro, desarrollar o adquirir una herramienta capaz de informarle al usuario los puertos abiertos, programas que están corriendo sobre dichos puertos, la versión de estos y parches de seguridad que necesita su PC con vista a la mejora de su seguridad informática.

Esta herramienta debe de estar montada en un sitio Web para que varias PC puedan conectarse remotamente y obtener un informe de las vulnerabilidades presente en su computadora. De estas herramientas llamadas Escáneres de Vulnerabilidades la de mayor relevancia es Nessus, la cual es utilizada en este sistema; además se adoptan otras herramientas para desarrollar el sistema como son: Apache, como servidor Web y postgreSQL, como gestor de base de datos; las cuales ayudaron a dar finalidad a la versión 1.0 del sistema.

Índice

INTRODUCCIÓN

Debido al desarrollo vertiginoso que están adquiriendo la informática en nuestro país es necesario desarrollar herramientas capaces de proteger nuestros intereses y soberanía por lo que se considera que el principal eslabón en el desarrollo, puesta en práctica y prueba de estas herramientas es la Universidad de Ciencias Informáticas. Entre las herramientas necesarias para mantener una buena seguridad informática en nuestros ordenadores son los Sistemas de Detección de Vulnerabilidades o Escáneres de Vulnerabilidades, como se conocen popularmente.

Los Escáneres de Vulnerabilidades son componentes vitales en redes de organizaciones medianas y grandes. La red de la Universidad continúa creciendo y se hace entonces cada vez más difícil conocer las vulnerabilidades presentes en nuestra red. Los usuarios de la red de la Universidad de Ciencias Informáticas no saben con certeza las vulnerabilidades que presentan su máquina de escritorio, ni a que tipos de ataques pueden ser sometidos desde la red, que actualizaciones debe colocar para evitar la penetración de programas intrusos que acceden a sus informaciones personales. Nuestros servidores tampoco se escapan de estos riesgos y se encuentran expuestos a algún posible ataque. Con este sistema cada usuario podrá saber que parches o programas necesita instalar en su PC para lograr una mejor seguridad informática. En el mundo existen muchas herramientas capaces de realizar estas tareas lo que ocurre es que la mayoría son propietarias y adquirirlas implicaría un gasto adicional al país en compra y actualizaciones. En el área de software libre existen herramientas de este tipo como Nessus (antes de la versión 3), que es herramienta líder entre los Escáneres de Vulnerabilidad.
Por esta misma situación el **Problema Científico** que se plantea es ¿Cómo diagnosticar las vulnerabilidades que presentan las computadoras personales y servidores de red de la Universidad de Ciencias Informáticas?
Por tanto el **Objeto de estudio** constituye ser La Gestión de Vulnerabilidades y el **Campo de acción** los Escáneres de Vulnerabilidades en la Universidad de Ciencias Informáticas.
Se persigue con esta investigación lograr como **objetivo general**: desarrollar una herramienta capaz de diagnosticar las vulnerabilidades de las computadoras personales o servidores. Como **objetivos específicos:** adoptar una herramienta capaz de diagnosticar vulnerabilidades; integrar todas las herramientas necesarias en un sitio Web.

Para cumplir con el objetivo propuesto se han definido las siguientes tareas:

- Estudiar y analizar:
  - Varios escáneres de puertos para posible adopción.
  - Herramientas capaces de detectar vulnerabilidades
  - Los reportes que se obtienen después de un test de vulnerabilidad.
- Permitir que el sistema muestre el reporte de vulnerabilidades.
- Garantizar que todas las herramientas que se utilicen tengan licencia de Software Libre.

Para una mejor organización y un mejor entendimiento del documento se encuentra estructurado el contenido con una breve explicación de sus partes en resumen, introducción, 5 capítulos de contenido, conclusiones, recomendaciones, bibliografía, anexos y glosario.

Capítulo 1, "*Fundamentación Teórica*", incluye un estado del arte de los Escáneres de vulnerabilidades, de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad.

Capítulo 2, "*Características del Sistema*", se define una propuesta del sistema, especificándose los requisitos de la herramienta, los cuales ayudan a desarrollar los casos de uso basándose en la metodología del RUP.

Capítulo 3, "*Análisis y Diseño del Sistema*", se define como continúa el desarrollo del sistema llevándose acabo el análisis y diseño del mismo.

Capitulo 4, "*Implementación y Prueba*", se llevan acabo el desarrollo del Diagrama de despliegue, y los diagrama de componentes.

Capítulo 5 "Estimación del proyecto", se determina el costo del sistema, así como el tiempo que es necesario invertir para la realización del mismo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Internet además de brindarnos información actualizada y novedosa sobre cualquier tema que deseemos, esta minada de software maligno, a los que son vulnerables todas las redes informáticas. Aparejado a esto se comenzaron a desarrollar herramientas de seguridad capaz de prevenir y evitar los posibles daños, las cuales se han tenido que ir desarrollando marcándole los pasos a los malware que se perfeccionan cada vez más. Las redes están amenazadas por programas capaces de aprovechar los puertos que tiene abierto las PC así como las vulnerabilidades que presentan algunos programas que corren en dichos puertos, estos programas ya dentro de una PC pudiera darle el control total de esta al atacante, los derechos de copiar, crear, modificar y eliminar cualquier información en ella. Los escáneres de vulnerabilidades son herramientas capaces de detectar las vulnerabilidades que presentan las máquinas y orientarles a los usuarios que parches deben usar para eliminarlas. En este capítulo se hará un estudio de cómo funcionan los sistemas de escaneo de vulnerabilidades, que herramientas utilizan, cuales son sus ventajas y desventajas, entre otras cuestiones que permitirán conocer si se puede adoptar alguna de estas herramientas, perfeccionarlas o crear la propia.

1.2 Seguridad Informática

Según la Real Academia Española [1] Seguridad: Calidad de seguro, Seguro: Libre y exento de Peligro, por lo que podemos concluir que Seguridad Informática es "la cualidad de un sistema informático exento de peligro". Sin embargo, esta no es una definición aceptable, para este trabajo. La seguridad informática consiste en asegurar que los recursos del sistema de información (material informático o programas) de una organización sean utilizados de la manera que decidieron y que el acceso a la información ahí contenida, así como su modificación sólo sea permitida a las personas que se encuentren acreditadas y dentro de los límites de su autorización [2]

Para que un sistema se pueda definir como seguro debe tener estas cuatro características:

- Integridad: La información sólo puede ser modificada por quien está autorizado.
- Confidencialidad: La información sólo debe ser legible para los autorizados.
- Disponibilidad: Debe estar disponible cuando se necesita.
- Irrefutabilidad: (No-Rechazo o No Repudio) Que no se pueda negar la autoría.

En estos momentos la seguridad informática es un tema de dominio obligado por cualquier usuario de la Internet, para no permitir que su información sea robada. Dependiendo de las fuentes de amenaza, la seguridad puede dividirse en seguridad lógica y seguridad física.

La **Seguridad Física** consiste en la "*aplicación de barreras físicas y procedimientos de control, como medidas de prevención y contramedidas ante amenazas a los recursos e información confidencial*"[3]. Se refiere a los controles y mecanismos de seguridad dentro y alrededor del Centro de Cómputo así como los medios de acceso remoto al y desde el mismo; implementados para proteger el hardware y medios de almacenamiento de datos [4].

La **Seguridad Lógica** se refiere a la seguridad en el uso de software y los sistemas, la protección de los datos, procesos y programas, así como la del acceso ordenado y autorizado de los usuarios a la información. La "seguridad lógica" involucra todas aquellas medidas establecidas por la administración -usuarios y administradores de recursos de tecnología de información- para minimizar los riesgos de seguridad asociados con sus operaciones cotidianas llevadas a cabo utilizando las tecnologías de información [5].

1.2.1 Términos relacionados con la seguridad informática

Activo: recurso del sistema de información o relacionado con éste, necesario para que la organización funcione correctamente y alcance los objetivos propuestos.

- Amenaza: es un evento que puede desencadenar un incidente en la organización, produciendo daños materiales o pérdidas inmateriales en sus activos.

- Impacto: medir la consecuencia al materializarse una amenaza.
- Riesgo: posibilidad de que se produzca un impacto determinado en un Activo, en un Dominio o en toda la Organización.
- Vulnerabilidad: posibilidad de ocurrencia de la materialización de una amenaza sobre un Activo.
- Ataque: evento, exitoso o no, que atenta sobre el buen funcionamiento del sistema.
- Desastre o Contingencia: interrupción de la capacidad de acceso a información y procesamiento de la misma a través de computadoras necesarias para la operación normal de un negocio.

Aunque a simple vista se puede entender que un Riesgo y una Vulnerabilidad se podrían englobar un mismo concepto, una definición más informal denota la diferencia entre riesgo y vulnerabilidad, de modo que se debe la Vulnerabilidad está ligada a una Amenaza y el Riesgo a un Impacto.

El término vulnerabilidad como posee gran relación con este trabajo de diploma se amplía a continuación.

1.2.2 Vulnerabilidad

El término vulnerabilidad, para la Seguridad Informática refiere a cualquier error de programación o mal configuración que pudiera permitir a un intruso obtener acceso no autorizado. Esto incluye cualquier problema, desde una contraseña débil en un enrutador o un error de programación sin corregir en un servicio de red.

Las vulnerabilidades en un ordenador conectado a una red son clasificadas, frecuentemente en:

- Vulnerabilidades Críticas.
- Fugas de información.
- Denegación de servicios.

Vulnerabilidades Críticas.

- Desbordamientos de buffers.
- Directory Traversal.
- Ataque a cadenas con formato (C, C++).
- Contraseñas por defecto.
- Errores de configuración.
- Puertas traseras conocidas.

Fugas de información.

- Divulgación de memoria.
- Información de red.
- Información de versión.
- Enumeración de usuarios.

Denegación de servicios.
Los ataques de Denegación de Servicios (DoS), logran que un servicio no esté disponible para los usuarios, usualmente consumiendo todos sus recursos o enviando algún tipo de señal que el servicio no maneja de forma correcta.

1.2.3 Herramientas de seguridad Informática

Existen distintas herramientas de seguridad informática. Estas herramientas pueden ser clasificadas de acuerdo a su uso en cuatro clases, aunque muchas de ellas pueden ser incluidas en cualquiera de estas clases.

- **Herramientas de Reconocimientos:** Son aquella herramientas que recolectan información y asisten al usuario en aprender acerca de la red. Estas herramientas pueden ser activas o pasivas, una herramienta de reconocimiento activo obtiene información haciendo algo en una forma detectable, por ejemplo enviando tráfico y esperando por respuesta. Una herramienta de reconocimiento pasivo trabaja de una forma diferente recibiendo tráfico de red no solicitado y analizándolo. Si una herramienta emplea ambos métodos se clasifica en activa, ejemplo de una herramienta de este tipo, podemos mencionar los sniffers.

- **Herramientas de ataque y penetración:** Estas herramientas prueban la fortaleza de una entidad de red (ej. Host, router, firewall, etc.) y ponen al descubierto sus vulnerabilidades. A menudo trabajan para explotar una vulnerabilidad o una clase de vulnerabilidades en un software, o explotando interacciones no intencionadas entre entidades en ambientes heterogéneos. Estas herramientas requieren actualizaciones periódicas para mantenerse al día en cuanto a las nuevas vulnerabilidades que salen a la luz. Estas herramientas son soportadas por las herramientas de reconocimientos. Ejemplo de este tipo de herramienta podemos encontrar los escáneres de vulnerabilidades.

- **Herramientas defensivas:** Estas herramientas ayudan al usuario en mantener una entidad segura. Ellas pueden realizar esta tarea ya sea encriptando el tráfico sensible, observando actividades ilícitas o bloqueando cierto tipo de tráfico de red. Estas herramientas requieren actualización para aprender acerca de las nuevas vulnerabilidades. Estas herramientas son soportadas por las herramientas de reconocimientos. Ejemplo de este tipo de herramientas están los firewall, sistemas detector de intruso, encriptadores.

1.3 Escáneres de vulnerabilidades

Los Escáneres de vulnerabilidad engloban el conjunto de acciones necesarias para identificar las IPs activas, los puertos y servicios ofrecidos, la identificación del sistema operativo, el nivel de actualizaciones de seguridad aplicadas e incluso la detección y explotación de las vulnerabilidades encontradas. En el mundo se desarrollan 2 tipos de escáneres de vulnerabilidad; los orientados a herramienta de administrador y los on-line.

1.3.1 Escáneres de vulnerabilidad orientados al administrador.

Estos escáneres de vulnerabilidad son herramientas usadas por administradores para detectar las vulnerabilidades que presenta su red. Son las más usadas en el mundo, dándoles a los administradores el poder de controlar sus redes. Por la filosofía que usan estos escáneres se pueden convertir en un arma de doble filo, ya que cualquier usuario que no sea administrador puede usarlas para detectarlas vulnerabilidades de una PC que desea atacar y culpar al administrador de esta fechoría.

1.3.1.1 Los 9 mejores Escáneres de Vulnerabilidad orientados al administrador .[6]

Nessus: Es uno de los mejores escaneadores de vulnerabilidades que existe, está disponible para varios sistemas operativos (Linux, FreeBSD, Windows, Solaris, Mac OS X). Está en constante actualización, consta de más de 11 000 plugins. Está compuesto por dos partes: un servidor, y un cliente. El servidor/daemon, "nessusd" se encarga de los ataques, mientras que el cliente, "nessus", se ocupa del usuario por medio de una interfaz gráfica o a través de la consola, está implementado en un lenguaje de Script permitiendo escribir tus propios plugins. Permite hacer escaneos programados. Escanea los puertos con uno de los mejores escaneadores, nmap; también posee técnicas de escaneo que son realiza mediante otros escáneres. Opcionalmente, los resultados del escaneo pueden ser exportados en reportes en varios formatos, como texto plano, XML, HTML, y LaTeX. El resultado del escaneo de puertos puede ser guardado como base de conocimiento para referencia en futuros escaneos de vulnerabilidades. Las pruebas de vulnerabilidad, disponibles en una larga lista de plugins, son escritos en NASL (Nessus Attack Scripting Language, Lenguaje de Scripting de Ataque Nessus por sus siglas en inglés), un lenguaje scripting optimizado para interacciones personalizadas en redes. Actualmente la versión 3 de Nessus no es Open Source pero sus plugins lo siguen siendo. Permite una auditoría del sistema tanto dentro como fuera. Algunas de las pruebas de vulnerabilidades de

Nessus pueden causar que los servicios o sistemas operativos se corrompan y caigan. El usuario puede evitar esto desactivando "unsafe test" (pruebas no seguras) antes de escanear (7).

GFI LANguard: Es un scanner de vulnerabilidades orientado a Windows: Windows 2000 (SP4), XP (SP2), 2003, VISTA versión: 8.0. Con GFI LANguard N.S.S puedes tratar diferentemente problemas relacionados con problemas de seguridad, administración de parches y auditoría de red en una sola aplicación. Posee más de 15 000 evaluaciones de vulnerabilidad. Importa y chequea las redes usando definiciones OVAL, esta es la base de datos estándar para definir las pruebas necesarias para determinar si las debilidades están presentes en un sistema. Permite el escaneo de hasta 10 hilos. GFI LANguard N.S.S. le da la información y las herramientas que necesita para realizar escaneos multiplataforma a través de todos los entornos, para analizar el estado de la seguridad de su red y para instalar y administrar eficazmente los parches de todos los equipos a través de diferentes sistemas operativos y en diferentes idiomas. Permite actualizarse por la red, permitiendo que los arreglos lleguen más rápido que si bajaran una nueva versión. Es código cerrado y no es libre (8).

Retina: Permite descubrir todas las computadoras conectadas, routers, y otros dispositivos de interconexión así como vulnerabilidades en las configuraciones, parches de seguridad. Además esta bajo la licencia GPL aunque no es libre. Su entorno es austero, con un explorador a su izquierda con opciones como “browser”, que permite navegar por un sitio web y ver texto oculto entre las etiquetas, así como los enlaces a otras urls, otra opción es “minner”, permite lanzar ataques por url en forma de CGI o lo que haga falta, *tracert*, como su nombre indica es un trazador de rutas en forma gráfico. Posee un programador de tareas para realizar muchas de las tareas mencionadas a una hora determinada. Retina permite combinarse con otros programas de la compañía, como una consola que controla todo el segmento de red, así como un programa para parchear con hotfixes Linux, Windows, SQL Server, IIS, etc. Retina necesita disponer de todos estos módulos para posicionarse como una solución completa. Y ya de por sí el precio inicial del programa original (sin contar estos otros programas) es bastante caro. Permite escanear varias plataformas, pero está soportado para Windows solamente (9).

Core Impact: Permite a los usuarios o administradores probar y explotar vulnerabilidades de seguridad en una computadora. Sistema operativo requerido Windows. Permite hacer pruebas de vulnerabilidad a varios sistemas operativos como Linux, CENTOS, Windows. No es un producto libre de costo ni está

bajo la licencia GPL. Soporta una gran cantidad de exploits profesionales en una base de datos que se actualiza con regularidad, y puede realizar trucos como realizar ataques redirigidos usando una máquina afectada con un exploit como esclavo. El principal inconveniente de esta herramienta es su precio: Puede llegar a varios miles de dólares. Hace pruebas a aplicaciones web mediante SQL injection y remote file, y demuestra las consecuencias de estos ataques. Permite configurar y testear la infraestructura defensiva de los firewall, sistemas de detección de intrusos. Identifica donde tu organización es un riesgo de la ingeniería social, phishing, spam. Core Security Technologies potencia su solución de tecnología líder con servicios de consultoría en seguridad world-class, incluyendo tests de intrusión, auditoría de seguridad de software y capacitación (10).

ISS Internet Scanner: Es una herramienta de testeo de vulnerabilidades a nivel de aplicación. Partió como un fino detector de Vulnerabilidades Open Source, no es libre de costo, soporta Windows 2000 Professional with SP4, Windows Server 2003 Standard SP1, Windows XP Professional with SP1. Internet Scanner busca los puntos débiles de la red para proteger los activos más importantes y evitar los riesgos que puedan derivar en la pérdida de disponibilidad, integridad o confidencialidad de la información fundamental para la empresa. Internet Scanner evalúa la seguridad de los sistemas de la red y prioriza las tareas de resolución, lo que permite resolver las vulnerabilidades de alto riesgo antes de que sean objeto de un ataque. Internet Scanner permite automatizar los escaneados y priorizar las vulnerabilidades detectadas para ofrece la respuesta más eficaz para la empresa. Internet Puede identificar más de 1300 tipos de dispositivos en red, incluidos sistemas de usuario final, servidores, routers y switches, cortafuegos, dispositivos de seguridad y routers de aplicaciones. Una vez que se han identificado todos los dispositivos en red, Internet Scanner analiza las configuraciones, los niveles de parche, los sistemas operativos y las aplicaciones instaladas para detectar las vulnerabilidades que pueden utilizar los hackers para obtener algún acceso no autorizado (11).

X-scan: El programa ha sido desarrollado por Xfocus, un grupo de hackers chinos dedicados al descubrimiento de vulnerabilidades en sistemas. Es una herramienta de hacking multihilo que analiza ordenadores y redes en busca de vulnerabilidades. Del equipo al que afecta obtiene numerosos datos, como el tipo y la versión del sistema operativo, estado de los puertos estándar, información del Registro de Windows y de los protocolos SNMP/NETBIOS, vulnerabilidades CGI/IIS/RPC, servidores SQL/FTP/SMTP/POP3, etc. Requiere el sistema operativo Windows y es libre de costo. Su última versión es 3.3 actualizada en el 2005. Posee un soporte de plugins e incluye soporte NASL. Lo curioso de la versión 3.0 y posteriores de X-Scan es que es totalmente compatible con otro escáner de

vulnerabilidades: Nessus, el más conocido entra la comunidad de usuarios de Linux. Este escáner open source cuenta con añadidos, lo que se entiende como plugins en el argot informático (12).

Sara: Asistente de Investigación para el Auditor de Seguridad (Security Auditor's Research Assistant). SARA es una herramienta de evaluación de vulnerabilidades derivada del escáner SATAN. Tratan de publicar actualizaciones dos veces al mes y de fomentar cualquier otro software creado por la comunidad de código abierto (como Nmap y Samba). Opera bajo los sistemas operativos Unix, Linux, MAC OS/X o Windows (a través de coLinux). Sara depende de coLinux para proveer el adecuado ambiente operativo para operar como un proceso de Windows. Integra la base de datos nacional de vulnerabilidades(NVD). Puede adaptarse a muchos ambientes protegidos por firewall. Es libre de costo y está bajo la licencia de SATAN. Coperative Linux (colinux) provee un ambiente virtual de Linux en Windows 200* y Windows XP. El ambiente coLinux parece vivir pacíficamente con las otras aplicaciones de Windows (13).

QualysGuard: Es una herramienta que funciona como servicio Web, lo cual evita todo el proceso de instalación, mantención, y compatibilidad con sistemas operativos, no es libre de costo. Después de la suscripción ofrece 14 días para chequear tu red y obtener un acceso directo de todas sus características totalmente gratis. Ofrece más de 5000 chequeos únicos de seguridad, un motor de búsqueda basado en inferencias y una base de conocimientos que se actualiza diariamente, conteniendo más de 1700 formas de posibles vulnerabilidades y cubriendo más de 300 aplicaciones sobre más de 20 plataformas diferentes. Permite determinar todos los activos de una red e identificar los detalles de un host como: sistema operativo, servicios, claves débiles etc. Identifica las vulnerabilidades de seguridad de manera regular mediante un calendario. Los resultados del scan son enviado encriptados (14).

Saint: Acrónimo de Security Administrator’s Integrated Network Tool, es otra herramienta comercial como Nessus o Retina y es derivado del SATAN. Corre en Unix y solía ser gratuita y Open Source, pero ahora es un producto comercial. Detecta y arregla posibles vulnerabilidades de la red, además permite explotar dichas vulnerabilidades. Escanea direcciones de ip pertenecientes a ipv4 y ipv6, permite almacenar los datos de las vulnerabilidades localmente o remotamente, incluye comprobación sobre servidores de www, pop y smb y además es actualizado con regularidad. Expone todos los peligros reales y potenciales, los ordena de mayor a menor severidad (15).

1.3.2 Escáneres on-line de vulnerabilidad.

Los Escáneres on-line de vulnerabilidad son herramientas que ofrecen el servicio de escanear tu propia PC desde la Web. Con este método el escaneo es realizado por un servidor al cliente que lo solicitó; brinda mayor grado de seguridad que el escaneo orientado al administrador ya que se hace imposible escanear otra PC que no sea en la que se encuentra el usuario solicitante (16).

1.4 Nmap.

Nmap ("mapeador de redes") es una herramienta de código abierto para exploración de red y auditoría de seguridad. Se diseñó para analizar rápidamente grandes redes, aunque funciona muy bien contra equipos individuales. Nmap utiliza paquetes IP "crudos" («raw», N. del T.) en formas originales para determinar qué equipos se encuentran disponibles en una red, qué servicios (nombre y versión de la aplicación) ofrecen, qué sistemas operativos (y sus versiones) ejecutan, qué tipo de filtros de paquetes o cortafuegos se están utilizando así como docenas de otras características. Aunque generalmente se utiliza Nmap en auditorías de seguridad, muchos administradores de redes y sistemas lo encuentran útil para realizar tareas rutinarias, como puede ser el inventariado de la red, la planificación de actualización de servicios y la monitorización del tiempo que los equipos o servicios se mantiene activos (17).

La salida de Nmap es un listado de objetivos analizados, con información adicional para cada uno dependiente de las opciones utilizadas. La información primordial es la "tabla de puertos interesantes". Dicha tabla lista el número de puerto y protocolo, el nombre más común del servicio, y su estado. El estado puede ser open (abierto), filtered (filtrado), closed (cerrado), o unfiltered (no filtrado). Abierto significa que la aplicación en la máquina destino se encuentra esperando conexiones o paquetes en ese puerto. Filtrado indica que un cortafuego, filtro, u otro obstáculo en la red está bloqueando el acceso a ese puerto, por lo que Nmap no puede saber si se encuentra abierto o cerrado. Los puertos cerrados no tienen ninguna aplicación escuchando en los mismos, aunque podrían abrirse en cualquier momento. Los clasificados como no filtrados son aquellos que responden a los sondeos de Nmap, pero para los que Nmap no puede determinar si se encuentran abiertos o cerrados. Nmap informa de las combinaciones de estado open|filtered y closed|filtered cuando no puede determinar en cual de los dos estados está un puerto. La tabla de puertos también puede incluir detalles de la versión de la aplicación cuando se ha solicitado detección de versiones. Nmap ofrece información de los protocolos

IP soportados, en vez de puertos abiertos, cuando se solicita un análisis de protocolo IP con la opción (-sO).

Además de la tabla de puertos interesantes, Nmap puede dar información adicional sobre los objetivos, incluyendo el nombre de DNS según la resolución inversa de la IP, un listado de sistemas operativos posibles, los tipos de dispositivo, y direcciones MAC.

Nmap es...

- Flexible: Posee docenas de técnicas avanzadas para escanear una red capaces de evadir filtros de IP, firewall, routers y otros obstáculos. Esto incluye varios mecanismos de escaneos de puertos, detección de sistema operativo, detección de versiones, barridos de ping y otros.

- Poderoso: Nmap ha sido usado para escanear grandes redes de centenares de máquinas.

- Portable: Es portable para muchos sistemas operativos incluyendo Linux, Microsoft Windows, FreeBSD, OpenBSD, Solaris, IRIX, Mac OS X, HP-UX, NetBSD, Sun OS, Amiga, y entre otros.

- Simple: Mientras que Nmap te ofrece avanzadas técnicas, su uso es tan simple como la tradicional línea de comandos o una interfaz amigable si es su preferencia.

- Gratis: Nmap se adquiere de una forma gratis y se puede visualizar y modificar su código bajos los términos de su licencia.

- Bien Documentado: en su sitio oficial se encuentra abundante documentación y en múltiples lenguajes.

- Respaldado: Nmap no tiene garantía, pero esta respaldado por una gran comunidad de desarrolladores a los que se les puede reportar errores y los problemas que este presente.

- Vitoreado: Nmap ha sido premiado por numerosas instituciones en las que se incluye "Information Security Product of the Year" por Linux Journal, Info World and Codetalker Digest. También ha sido presentada en numerosos artículos de revistas, varios videos, docenas de libros y en unas series de comic.

- Popular: Ciento de personas descargan Nmap todos los días e incluyendo para varios sistemas operativos (Redhat Linux, Debian Linux, Gentoo, FreeBSD, OpenBSD, etc.).

1.5 ¿Qué se va hacer?

Debido a que la Universidad de Ciencias Informáticas está compuesto en su mayoría por estudiantes y profesores que aunque algunos no sean informáticos estudiaron carreras afines con esta y el alto número de computadoras que posee nos obliga hacer una consideración mejor de cómo vamos a desarrollar nuestro sistema.

La mejor opción para un centro como este es un Escáner de Vulnerabilidades On-line debido a que se logra que los usuarios sean los encargados de la seguridad de su propia PC, se logra un mejor control de la herramienta de escaneo gracias a que se configuraran solo para que se escanee la PC desde la cual se hace la solicitud y se puede obtener también el usuario que lo solicitó.

Según el análisis que realizamos con los escáneres de vulnerabilidad orientados al administrador, obtuvimos como mejor propuesta para realizar la adaptación a un sistema Web, a la herramienta Nessus, debido a que posee licencia de software libre, hasta la versión 2.2.10, lo que posibilita un buen conocimiento de su funcionamiento; posee un modelo de arquitectura cliente servidor que permite la concurrencia de varios usuarios; posee un gran número de plugins para detectar vulnerabilidades y una buena comunidad de desarrollo; también posibilita que en esta universidad se puedan desarrollar los plugins para las nuevas vulnerabilidades que vayan surgiendo.

1.6 Metodología de Desarrollo (RUP).

Para organizar, guiar y controlar el desarrollo del sistema se decidió utilizar como metodología del software el Proceso Unificado de Modelado (RUP). El Proceso Unificado de Rational (RUP, el original inglés Rational Unified Process) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos; puesto que RUP es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante, las cuatro fases que incluye RUP son: inicio, elaboración, construcción y transición.
Se resume en tres características fundamentales:

Dirigido por casos de uso: Un caso de uso representa una serie de acciones que el sistema realiza para aportar un resultado al usuario. Los casos de uso representan los requerimientos funcionales. Guían el proceso de desarrollo de software ya que a partir de los casos de uso se realiza diseño, implementación y prueba del sistema.

Centrado en la arquitectura: Muestra una visión del sistema completo, describe los elementos del modelo que son más importantes para su construcción. Influye en la selección de los casos de uso significativos y estos a su vez guían la arquitectura.

Iterativo e Incremental: Para facilitar el trabajo, el proyecto se realiza mediante iteraciones que terminan con un incremento. Cada iteración comprende diferentes flujos de trabajo y de esta resulta una versión de un producto que irá creciendo en cada iteración.
Entre los beneficios que aporta la utilización de RUP se encuentran los siguientes:

- Reduce el coste de riesgo al coste de un solo incremento.
- Reduce el riesgo de no sacar el producto al mercado en la fecha prevista.
- Permite definir los requisitos del usuario a medida que se va desarrollando el producto.

RUP presenta como particularidad que agrupa las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

1.7 Plataforma de Desarrollo (Linux).

Linux es uno de los ejemplos más prominentes del software libre y del desarrollo del código abierto, cuyo código fuente está disponible públicamente, para que cualquier persona pueda libremente usarlo, estudiarlo, redistribuirlo, comercializarlo y, con los conocimientos informáticos adecuados, modificarlo.

- **Es más seguro**
  - Ya que la gran mayoría de los ataques de hackers son dirigidos a servidores Windows al igual que los virus los cuales se enfocan principalmente a servidores con éste sistema operativo.
  - La plataforma Linux es más robusta lo cual hace más difícil que algún intruso pueda violar el sistema de seguridad de Linux.
- **Es más rápido**
  - Al tener una plataforma más estable, esto favorece el desempeño de aplicaciones de todo tipo tales como: bases de datos, aplicaciones XML, multimedia, etc.
  - La eficiencia de su código fuente hace que la velocidad de las aplicaciones Linux sean superiores a las que corren sobre Windows lo cual se traduce en velocidad de su página.
- **Es más económico**
  - Ya que requieren menor mantenimiento. En servidores Windows es más costoso debido a que es necesaria una frecuente atención y monitoreo contra ataques de virus, hackers y errores de código, instalación y actualización de parches y service packs.
  - El software Linux así como también un sin número de aplicaciones son de código abierto (gratuitos).
  - No requieren supervisión tan estrecha ni pagos de pólizas de mantenimiento necesarias para obtener los Service Packs.

1.7.1 Modelo de Arquitectura (Cliente-Servidor)

Esta arquitectura consiste básicamente en que un programa -el Cliente informático- realiza peticiones a otro programa -el servidor- que le da respuesta.

Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

Ventajas de la arquitectura Cliente-Servidor

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.

1.7.2 Patrón de Arquitectura (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 de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos. Esto es útil ya que los modelos típicamente tienen cierto grado de estabilidad (dependiendo de la estabilidad del dominio del problema que está siendo modelado), donde el código de la interfaz de usuario sea más robusto, debido a que el desarrollador esta menos propenso a "romper" el modelo mientras trabaja de nuevo en la vista.

1.7.3 Entorno de desarrollo (Zend Studio)

Zend Studio es un poderoso IDE creado para aplicaciones php que incluye las siguientes características:

- **Herramienta de Depuración**- Control de los procesos de la aplicación PHP y recibo de informe de errores, llamada a pilas, variables y las de entorno de salida.
- **Gestión de proyecto y archivos**- Gestiona los archivos y carpetas en Zend Studio, incluyendo las funcionalidades de FTP.
- **Inspector de Proyecto y Archivo**- Para funciones de visualizaciones y de funcionalidades, constantes, clientes soap etc.
- **Completamiento de código**- ofrece una amplia selección de características de completamiento, incluyendo PHP, HTML, clases, variables miembros, variables, palabras claves y listado de completamiento de código objeto.
- **Plantillas de Código**- Plantillas que ayudan a escribir código de una forma rápida y precisa.
- **Resaltador de sintaxis**- Automáticamente se aplica un resaltador de texto para elementos de diferentes sintaxis.

1.7.4 Lenguaje de desarrollo

1.7.4.1 PHP

PHP: es un lenguaje de programación interpretado usado normalmente para la creación de páginas web dinámicas. PHP es un acrónimo recursivo que significa "**P**HP **H**ypertext **P**re-processor" (inicialmente PHP Tools, o, *P*ersonal *H*ome *P*age Tools). Actualmente también se puede utilizar para la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

Usos de PHP

Los principales usos del PHP son los siguientes:

- Programación de páginas web dinámicas, habitualmente en combinación con el motor de base datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl o Shell scripting.

- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y Qt/GTK+, lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Ventajas

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.

1.7.4.2 NASL

NASL responden a *Nessus Attack Scripting Language,* es un lenguaje de scripting con una sintaxis basada en C, fue realizado especialmente para Nessus. Este lenguaje contiene grandes facilidades para la manipulación de cadenas de caracteres y también para la manipulación de arreglos. Estos dos tipos de datos son fundamentales en el procesamiento de datos a través de una red.

1.7.4.3 HTML

El HTML, acrónimo inglés de *Hypertext Markup Language* (lenguaje de formato de documentos de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Explorer, Mozilla, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

HTML es hijo de SGML, aunque hay unas versiones de XHTML que son descendientes de XML y exigen que se escriba mucho más para facilitar la vida a los navegadores, que son aquellos programas que muestran información en pantalla.

1.7.4.4 AJAX

AJAX, acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaSript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asincrónica con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilo en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAscript como JavaScript y Jscript , para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML. [*VER ANEXO 1*]

Diferencias con las aplicaciones web tradicionales

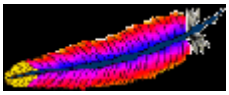
Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.[18] [VER ANEXO 2]

1.8 Herramientas tecnológicas

1.8.1 Servidor Web (Apache)



El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: en 2005, Apache fue el servidor HTTP más usado, siendo el servidor empleado en el 48% de los sitios web en el mundo. Sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft).

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Ventajas de Apache

- Modular
- Open source
- Multi-plataforma
- Extensible

- Popular (fácil conseguir ayuda/soporte)
- Gratuito

1.8.2 Gestor de Base de Datos (PostgreSQL)

PostgreSQL es un servidor de base de datos relacional orientada a objetos de software libre, liberado bajo la licencia BSD.

Como muchos otros proyectos open source, el desarrollo de **PostgreSQL** no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (***PostgreSQL Global Development Group***).

Características

Algunas de sus principales características son:

- **Alta concurrencia**

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- **Amplia variedad de tipos nativos**

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)

- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

- **Otras características**

- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (*foreign keys*).
- Disparadores (*triggers*).

Un *disparador* o *trigger* se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. Ahora todos los disparadores se definen por seis características: -El nombre del trigger o disparador -El momento en que el disparador debe arrancar -El evento del disparador deberá activarse sobre... -La tabla donde el disparador se activara -La frecuencia de la ejecución -La función que podría ser llamada Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (triggers).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

- **Funciones**

Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientación a objetos o la programación funcional.

Los disparadores (triggers en inglés) son funciones enlazadas a operaciones sobre los datos.

Algunos de los lenguajes que se pueden usar son los siguientes:

- Un lenguaje propio llamado PL/PgSQL (similar al PL/SQL de Oracle).
- C.
- C++.
- Gambas
- Java PL/Java web.
- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.
- PL/sh.
- PL/Tcl.
- PL/Scheme.
- Lenguaje para aplicaciones estadísticas R through PL/R.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (query en inglés).

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (stored procedures en inglés).

1.8.3 XML

XML, sigla en inglés de ***Ex****tensible* ***M****arkup* ***L****anguage* («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es

realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Ventajas de XML

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

1.9 Herramienta CASE.

Visual Paradigm

Visual Paradigm ofrece un entorno de creación de diagramas para UML 2.0, un diseño centrado en casos de uso y enfocado al negocio, además brinda el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación, además de ser disponible en múltiples plataformas. Visual Paradigm incluye la representación gráfica de una colección de elementos de modelado a menudo dibujada como un grafo con vértices conectados por arcos. Como características presenta que ofrecen un mecanismo general para la organización de los modelos/subsistemas/capas agrupando

elementos de modelado. Cada paquete se corresponde a un submodelo (subsistema) del modelo (sistema). También de que se pueden anidar paquetes. Una clase de un paquete puede aparecer en otro paquete por importación a través de una relación de dependencia entre paquetes.
Posee entre sus principales características las siguientes:

- Es profesional: brinda la posibilidad de crear un conjunto bastante amplio de artefactos utilizados con mucha frecuencia durante la confección de un Software. Todos estos, cumpliendo con el Standard UML 2.0.
- Es amigable: puede ser utilizado en varios idiomas, sus componentes se encuentran relacionados, por lo que se hace muy fácil la creación de cualquier tipo de diagrama, ya que cada componente utilizado en el diagrama que se esté creando, sugiere nuevos posibles componentes a utilizar, por lo que ya no es necesario localizarlos en la barra donde pueden aparecer un número grande de componentes.
- Brinda un número considerable de estereotipos a utilizar, lo que permite un mayor entendimiento de los diagramas.
- Facilidades para redactar especificaciones de casos de uso: es posible crear plantillas para las especificaciones de casos de uso y describirlos, por lo que no se necesita de una herramienta externa como editor de texto.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así cómo obtener diagramas a partir del código.
- Integración con distintos Ambientes de Desarrollo Integrados (IDE): se integra fácilmente con varios IDEs, entre ellos el de Visual Studio y el Eclipse.
- Interoperabilidad con otras aplicaciones: brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose. Además permite importar y exportar XML y XMI.
- Generación de código ORM: permite generar a partir de un Diagrama de Entidad Relación una Base de Datos Relacional y el código necesario para acceder a esta base de datos utilizando Java, PHP, C# o Enterprise Object Framework.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.

1.10 Conclusiones.

En este capítulo se ha hecho una valoración de las herramientas que se usan actualmente en el mundo para detectar las vulnerabilidades de las redes y cómputos informáticos, se muestran sus ventajas y propiedades, de las cuales valoramos la herramienta más conveniente para nuestro sistema y la forma de realizarlo. También se expusieron las tecnologías y otras herramientas que usaremos en el desarrollo de nuestro sistema.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

En este capítulo se hace una descripción del objeto de estudio del trabajo, se describe el entorno de trabajo sobre el que se desarrolla la aplicación. Además, se hace una propuesta del sistema y se analizan los requerimientos funcionales y no funcionales. Se realiza el modelado de dominio y se definen los casos de usos y los actores que intervienen en cada caso de uso. También se muestran los diagramas realizados para cada caso de uso.

2.2 Objeto de estudio.

En el mundo existen diversos sistemas de detección de vulnerabilidades los cuales te brinda el conocimiento de los problemas que presentan las PC de tu red informática, ellos trabajan con la filosofía de: úsenme antes que un intruso aproveche su vulnerabilidad. La herramienta líder de estos sistemas es Nessus ya que es la más usada a nivel mundial y posee un gran número de pruebas a realizar simulando ataques para descubrir vulnerabilidades; es un escáner orientado al administrador lo que dificulta su uso extendido en una red amplia. Otra herramientas utilizadas son los Escáneres de vulnerabilidades Online los que te permiten desde un sitio Web darte los diagnósticos de vulnerabilidad de una PC solicitante, en este caso cada usuario se encargaría de realizar las pruebas sobre su PC.

2.2.1 Problema y situación problémica.

En la Universidad de Ciencias Informáticas que posee una gran infraestructura de redes, computadoras con informaciones muy importantes para el desarrollo de proyectos productivos, grandes informaciones de estudiantes y profesores, no posee una herramienta que sea capaz de detectar las vulnerabilidades de sus servidores y computadoras personales para una posterior instalación de los parches correspondientes para mantener la seguridades de las computadoras personales del centro. Cualquier intruso con aras de hacer el mal, se puede aprovechar de estos huecos de seguridad no parchados para corromper cualquier información que posee alguna PC determinada.

Toda esta situación de seguridad no queda solamente en manos de los administradores de red y los técnicos de laboratorio, debido a que las PC de los cuartos también pueden ser sometidas a ataques de intrusos, además quien mejor para proteger la seguridad de su máquina que el propio usuario.

2.2.2 Objeto de automatización.

Teniendo en cuenta lo planteado anteriormente se hace necesario desarrollar un sistema capaz de detectar las vulnerabilidades de las PC o servidores de la red informática UCI y les oriente donde encontrar los parches para eliminar esta vulnerabilidad, un sistema que permita además la actualización de las pruebas de vulnerabilidades.

2.2.3 Información que se maneja.

Se manipula la información de los puertos abiertos de las PC que solicitan las pruebas de vulnerabilidad, así como los servicios que corren en estos puertos y el usuario que solicito la prueba, la configuración del tipo de escaneo que se desea realizar y el informe de reportes que se genera. Se maneja la actualización de las pruebas de vulnerabilidad.

2.2.4 Propuesta de sistema.

En este trabajo se propone un sistema que desde un ambiente Web un usuario pueda solicitar un escaneo de vulnerabilidades de su PC, el sistema se encargue de hacerles las pruebas, generarle un reporte que le informe sus vulnerabilidades y los parches que debe usar para eliminarlas. El administrador de la Web se debe encargue de actualizar las pruebas de vulnerabilidad (Plugins) o garantizar que se realice de forma automática, crear los directorios donde se guardaran los reportes, configuraciones de escaneos e IPs a escanear, además este tiene la posibilidad de escanear cualquier PCs de la red.

2.3 Modelo de Dominio.

Debido al bajo nivel de estructuración que presenta el negocio que se esta estudiando, ya que se centra en su mayoría en herramientas y tecnologías informáticas, se propone un modelo de dominio que permita visualizar los principales conceptos que se manejan en el sistema a desarrollar. Trayendo consigo un mejor entendimiento del contexto en que se emplaza el sistema por parte de los usuarios, desarrolladores e interesados.

Conceptos

Usuario: Persona que interactua con el sistema através de su interfaz.

Administrador: Usuario con permiso de actualización de los plugins del sistema.

Sistema: Conjunto de aplicaciones informáticas que interactuan entre si con un propósito específico.

PC: Computadora personal.

Puertos: es una forma genérica de denominar a una interfaz por la cual diferentes tipos de datos pueden ser enviados y recibidos. Dicha interfaz puede ser física, o puede ser a nivel software.

KB: La Base de Conocimiento (Knowledge Base) no es más que una lista de toda la información recopilada sobre un host analizado. Se almacenan todos los resultados del escaneo de puertos, análisis de servicios y los host activos. Permite la reutilización de análisis previos, evitando la redundancia en los análisis.

Reporte: Resultado de las pruebas de vulnerabilidades a las PC solicitantes.

Nessus: Es un escáner de vulnerabilidades libre y actualizado que brinda grandes posibilidades y tiene una arquitectura flexible/extensible.

Escáner: Herramienta para detectar los puertos abiertos de una PC desde la red, incluso algunos detectan los servicios que corren en estos. Ej.: Nmap.

Plugins: Son las pruebas de vulnerabilidad que se realizan sobre los puertos abiertos de la PC que solicitó el escaneo de vulnerabilidades.

El modelo del dominio cuenta con un diagrama de clases UML donde se especifican las principales clases conceptuales que pueden intervenir en el sistema a desarrollar, como muestra la figura 2.1

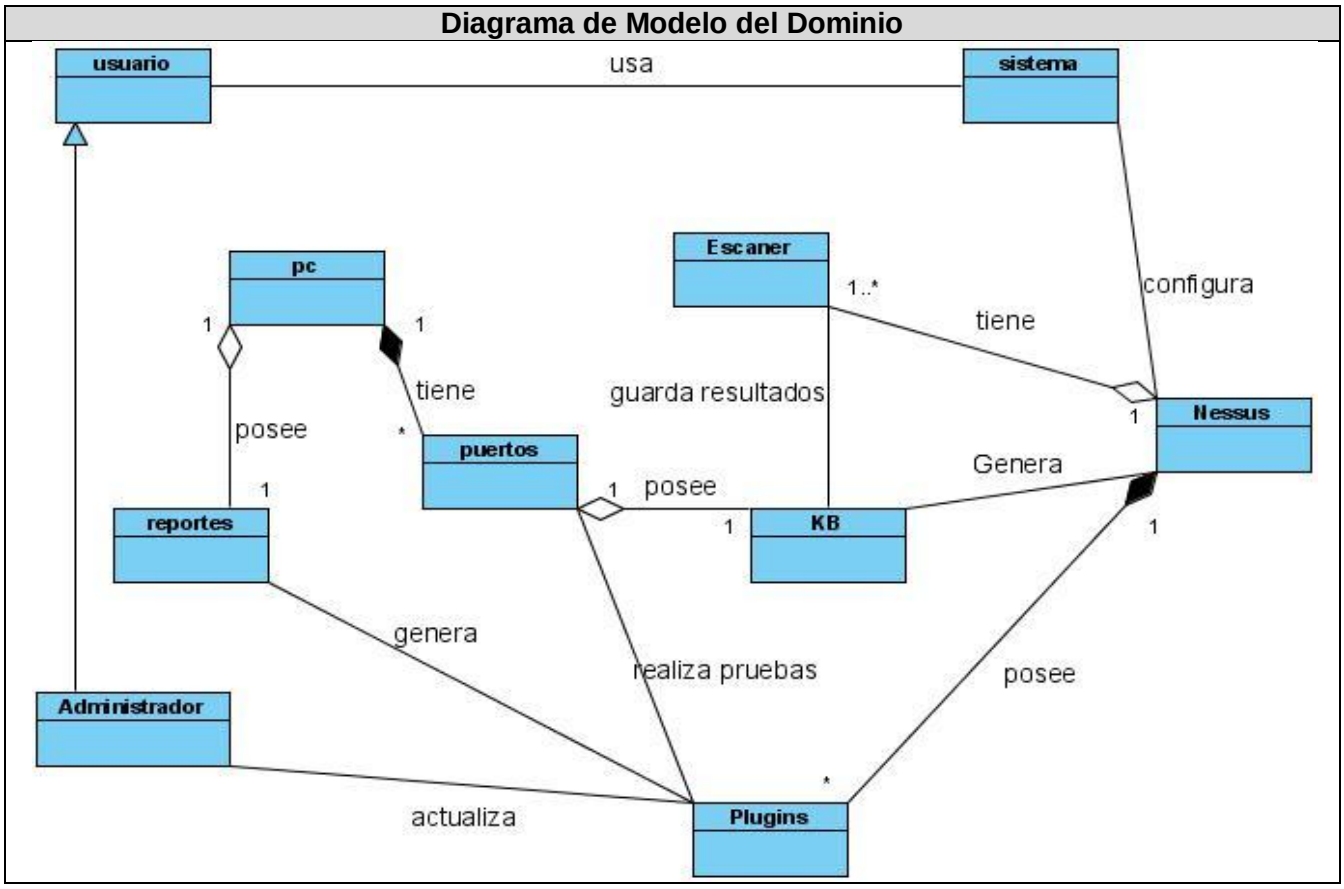


Figura 2.1 Diagrama de Modelo del Dominio

2.3.1 Reglas del Negocio.

Como reglas del negocio se establecen las siguientes:

- El sistema solo permitirá que se detecten las vulnerabilidades únicamente sobre la máquina en la que se encuentra el usuario que solicitó el escaneo y solamente en esta se visualizara el reporte posteriormente.

2.4 Especificación de los requisitos de software.

2.4.1 Dependencias y Relaciones con otros software.

Para lograr un buen funcionamiento de nuestro sistema usaremos Nessus para realizar las pruebas de vulnerabilidades sobre los puertos y generar los reportes. Esta herramienta garantiza que las pruebas sean correctamente hechas y los reportes de vulnerabilidad se realicen con los estándares mundiales y relacionados con la CVE (*Common Vulnerabilities and Exposures*).

2.4.2 Requerimientos Funcionales.

Los Requerimientos Funcionales son un listado de las características básicas del sistema, las que serán enumeradas según la jerarquía que exista entre ellas.

1. Solicitar escaneo.
2. Gestionar IP
   2.1 Obtener IP de la PC solicitante.
   2.2 Registrar IP de la PC solicitante.
3. Registrar usuario
   3.1 Solicitar usuario y contraseña.
   3.2 Verificar usuario y contraseña.
4. Otorgar permisos correspondientes.
5. Mostrar opciones de escaneo.
6. Seleccionar opciones de escaneo.
7. Guardar configuración de opciones de escaneo.
8. Registrar configuración.
9. Escanear puertos de la PC solicitante.
10. Realizar pruebas de vulnerabilidades.
11. Recibir informe de vulnerabilidades.
12. Generar reporte de vulnerabilidades.
13. Mostrar reporte al usuario.

14. Almacenar el reporte.

15.Solicitar actualización.

16.Mostrar posibilidad de actualización.

17.Actualizar pruebas de vulnerabilidades.

17.1 Actualizar plugins.

18 Cambiar directorios de archivos.

18.1 Cambiar el directorio donde se almacenan los ip de las máquinas ha escanear.

18.2 Cambiar el directorio donde se almacenan los informes.

18.3 Cambiar el directorio donde se almacenan las configuraciones hechas por los usuarios.

2.4.3 Requerimientos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen que el producto sea atractivo, usable, rápido o confiable.

- **Apariencia o interfaz externa.**

  La apariencia de nuestro sistema es bastante sencilla al usuario, posee solo 3 interfaces, de las cuales los usuarios aceden a 2, una de ellas tiene 5 checkbox, 2 radiobox y un botón para seleccionar los tipos de configuraciones y los tipos de escaneo, y la otra interfaz posee 2 texbox además del botón para comprobar autenticidad del usuario.

- **Usabilidad.**

  El sistema le permite al usuario realizar las configuraciones del escaneo de una forma rápida, entendible y sin complicaciones, además se le facilita un manual que explica de una forma abreviada y amena en que consiste cada configuración.

- **Rendimiento.**

A pesar del gran procesamiento que se lleva acabo en el sistema el rendimiento puede ser excelente gracias a los bajos requerimientos de hardware que necesita el sistema el cual aumentando sus niveles optimizaría el rendimiento.

- **Soporte.**

Extensibilidad: El sistema permitirá su extensibilidad, permitiendo agregar nuevas funcionalidades en un futuro.

Mantenimiento: El sistema debe ser bien documentado de forma tal que en caso de mantenimiento el tiempo que se requiera sea el mínimo.

- **Seguridad.**

El acceso será controlado con nombres de usuario y contraseñas. Solo los usuarios con derechos de administrador podrán tener acceso a las funciones administrativas. El sistema solo permitirá ver el reporte de Vulnerabilidades de la PC solicitante en el caso del usuario común.

- **Aspectos legales y de licencias.**

El sistema en su conjunto se ha desarrollado sobre herramientas OpenSource, licencias que permiten hacer modificaciones y adaptaciones de las herramientas para el sistema.

- **Hardware.**

Se requiere un servidor de base de datos con 160Gb de disco duro a lo sumo, 2 servidores en la capa de negocio, una para Nessusd con 1Gb o 2Gb de RAM a lo sumo para una red pequeña (300 PC) y si es posible más de un núcleo en el CPU, y el otro para la aplicación Web con las mismas prestaciones.

- **Software.**

El sistema trabajara sobre una distribución Linux, con servidor Web Apache, PostgreSQL como gestor de base de datos y de Escáner de Vulnerabilidad a Nessus.

2.5 Definición de los casos de uso.

Los casos de uso constituyen una técnica narrativa para describir el comportamiento del sistema y sus funcionalidades. Cada caso de uso puede describir una o más funcionalidades requeridas para el sistema por parte del usuario.

2.5.1 Definición de los actores.

En la siguiente tabla se muestra una descripción del rol que desempeña cada uno de los actores del sistema.

Actores	**Justificación**
Usuario	Es la persona que le solicita al sistema el escaneo de su PC y la que recibe el reporte de este.

Administrador	Es la persona encargada de actualizar los plugins del escáner de vulnerabilidades, puede modificar la dirección de los directorios donde se guardan las configuraciones del escaneo, los reportes y los IPs.

Tabla 2.1 Definición y descripción de los actores del sistema.

2.5.2 Listado de los casos de uso.

Como se puede ver, en las siguientes tablas se hace una descripción de los casos de uso del sistema, definiéndose el actor que inicializa cada caso de uso y una breve reseña de las funcionalidades que cumple cada uno.

CU-1	Configurar Escaneo de Vulnerabilidades.
Actor	Usuario
Descripción	El usuario selecciona una de las dos opciones de escaneo, escaneo básico o escaneo avanzado.
Referencia	R-8,9,10,11

Tabla 2.2 CU - Configurar Escaneo de Vulnerabilidades.

CU-2	Realizar Escaneo de Vulnerabilidades.
Actor	Usuario
Descripción	El sistema ejecuta el escáner y obtiene el reporte que este genera y se lo muestra al usuario.
Referencia	R-12,13,14,16,17

Tabla 2.3 CU - Realizar Escaneo de Vulnerabilidades.

CU-3	Gestionar Actualización.
Actor	Administrador
Descripción	El administrador ingresa en el sistema los plugins de actualización para que las pruebas de vulnerabilidades sean lo más reciente posible.
Referencia	R-18,19,20, 20.1

Tabla 2.4 CU – Gestionar Actualización.

CU-4	Autenticar
Actores	Usuario y Administrador
Descripción	El usuario se autentica en el sistema, este comprueba que tiene permiso a usar el sistema.
Referencia	R- 1, 2, 2.1, 2.2, 3, 3.1, 3.2, 4, 5, 6, 7

Tabla 2.5 CU – Autenticar

CU-5	Configurar Directorios
Actor	Administrador
Descripción	El Administrador configura los directorios en los que se van a guardar las configuraciones de escaneo de los usuarios, sus IPs, así como el resultado de sus reportes.
Referencia	R- 18, 18.1, 18.2, 18.3

Tabla 2.6 CU – Configurar Directorios

2.5.3 Diagrama de casos de uso.

La figura 2.2 muestra una representación gráfica de los casos de uso del sistema así como su relación con los actores del sistema, los cuales son los encargados de inicializar en interactuar con cada caso de uso.

Diagrama de Casos de Uso del Sistema

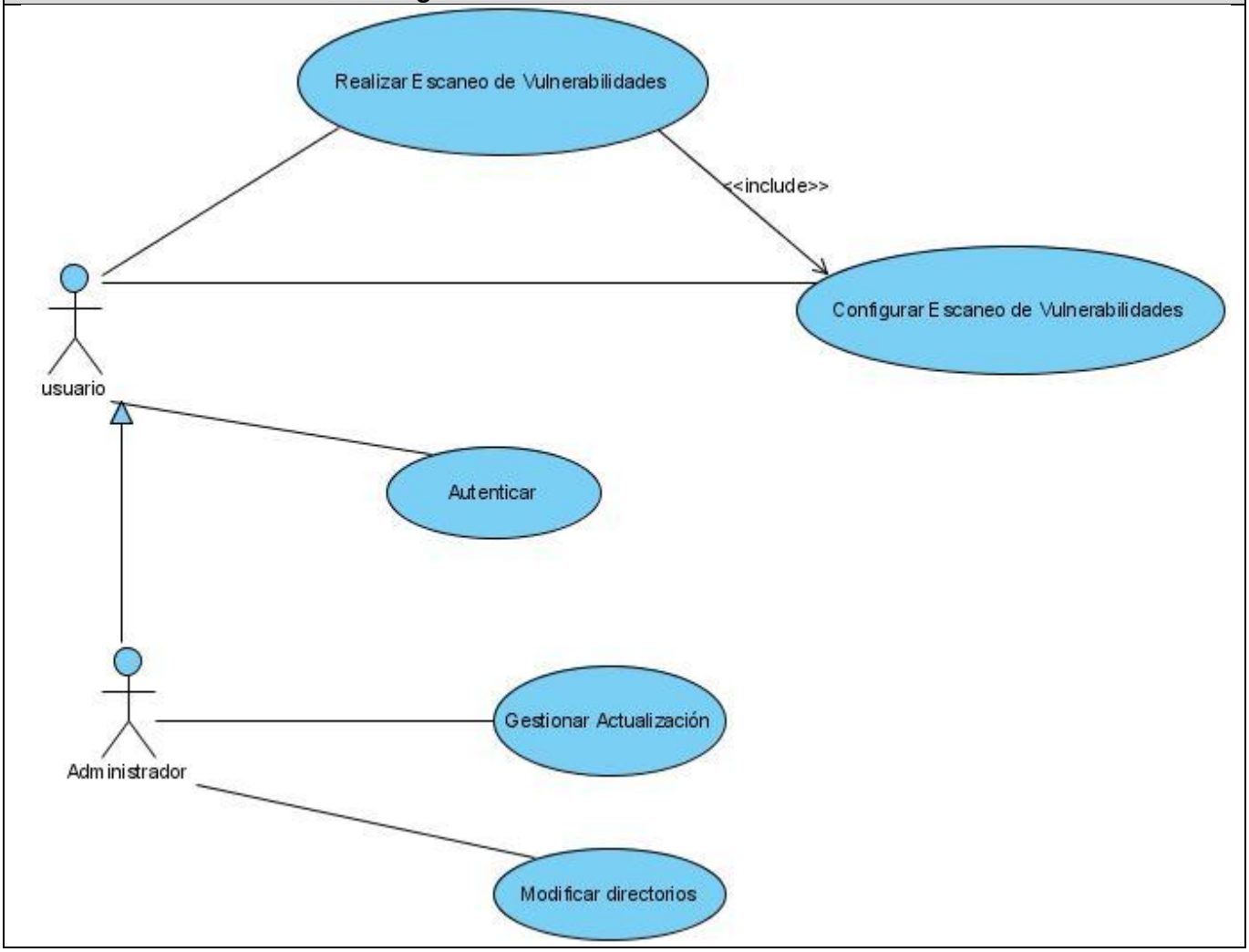


Figura 2.2 Diagrama de Casos de Uso del Sistema

2.5.4 Casos de uso por ciclo.

Ciclo de desarrollo primario			
Código	**Nombre del Caso de Uso**	**Paquete**	**Justificación**
CU-1	Configurar Escaneo de Vulnerabilidades.		Son los casos de uso que representan el proceso primario imprescindible
CU-2	Realizar Escaneo de Vulnerabilidades.		

CU-4	Autenticar		para el funcionamiento básico del sistema e influye en la arquitectura básica.
Ciclo de desarrollo secundario			
CU-3	Gestionar Actualización.		Son casos de uso importantes pero no imprescindibles para el funcionamiento básico del sistema.
CU-5	Configurar Directorios		

Tabla 2.7 Ciclo de Desarrollo.

2.5.5 Casos de uso expandidos.

En las siguientes tablas se hace una descripción más detallada de cada caso de uso del sistema. Cada tabla muestra en secciones una secuencia de las acciones (actores) y respuestas del sistema para cada una de las funcionalidades del caso de uso. Cada tabla puede tener una o más secciones en dependencia de la complejidad de cada caso de uso. [*VER ANEXO 1*]

2.6 Conclusiones.

En este capítulo se ha planteado la necesidad de que exista un Escáner de Vulnerabilidades y los objetivos que cumple este, el por qué es necesario que se desarrolle en un entorno Web. Se hace propuesta de un sistema capaz de resolver estas dificultades de vulnerabilidades de las PC. Se hace además un análisis de los requerimientos funcionales y no funcionales del sistema, facilitando la identificación de los casos de uso del sistema.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se centrará más en los aspectos internos del sistema desde el punto de vista de los desarrolladores, estructurado por las clases y paquetes desde una vista interna. Este capítulo esta centrado principalmente a los desarrolladores, por tanto se introducen mayores especificaciones internas del sistema. Se especificarán detalles arquitectónicos, así como diagramas de clases del análisis, del diseño y los diagramas de interacción.

3.2 Análisis.

A través del análisis ofrecemos una especificación más precisa de los requisitos que la que tenemos de la captura de requisitos. El modelo de análisis estructura los requisitos de un modo que facilita su comprensión, su preparación, su modificación y en general su mantenimiento.

Clases Interfaz: Se utiliza para modelar la interacción de entre los actores y el sistema. permitiendo recibir o presentar información hacia los actores y hacer peticiones hacia los sistemas expertos. Representa abstracciones de ventanas, formularios, paneles etc.

Clases de control: Estas clases representan secuencia, transacciones, se usan con frecuencia para encapsular el control de un caso de uso en concreto.

Clases de entidad: Estas son las clases encargadas de modelar la información que posee larga vida y que es a menudo persistente.

Los diagramas de clases muestran las interacciones de estos tipos de clases para cada caso de uso, permitiendo una mejor comprensión.

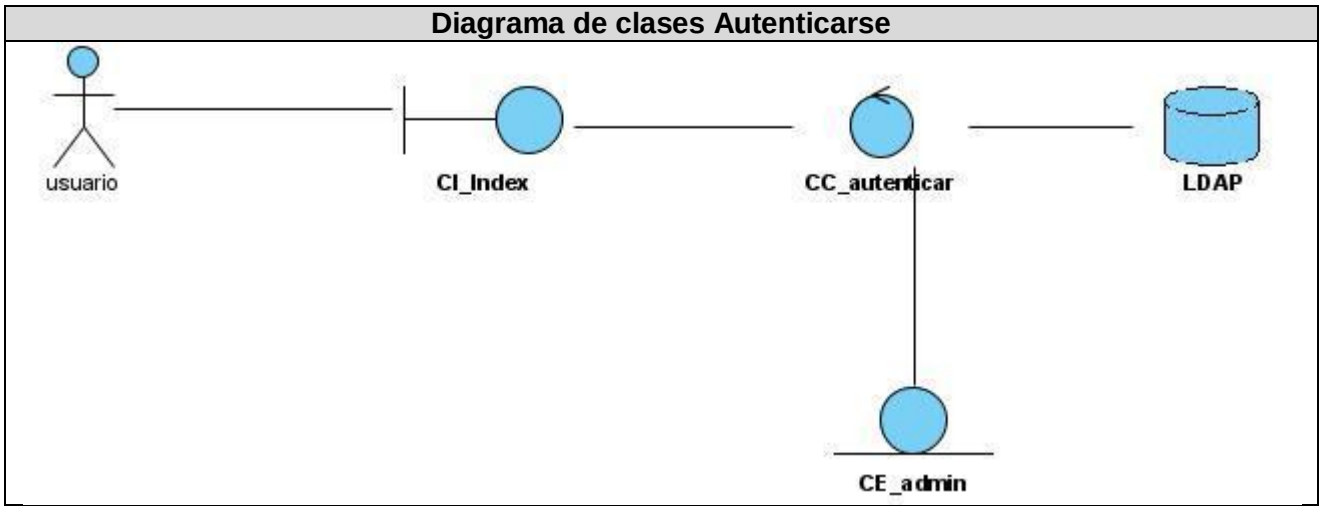


Figura 3.1 Diagrama de clases Autenticarse

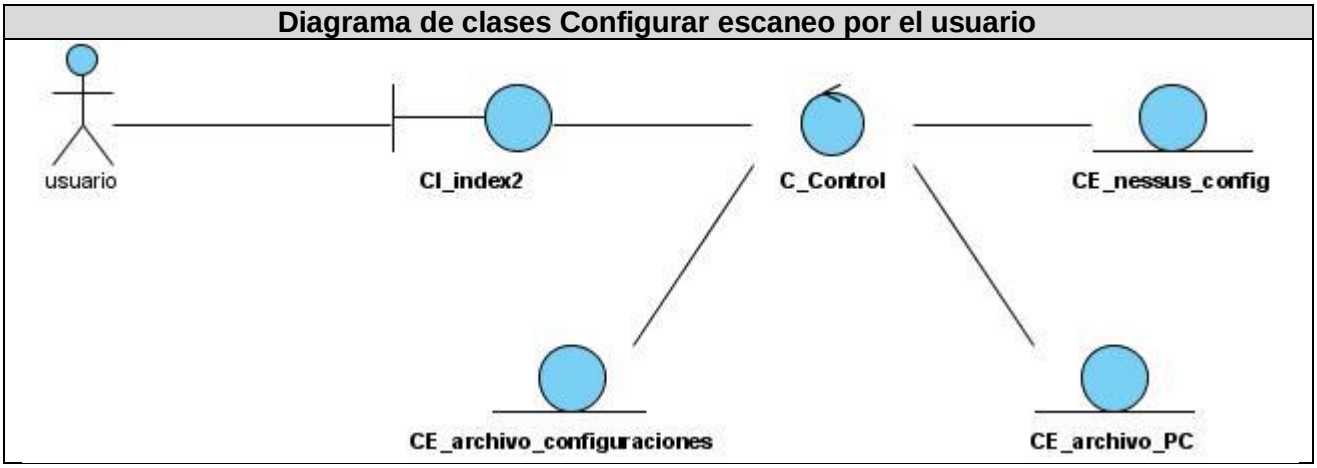


Figura 3. 2 Diagrama de clases Configurar escaneo por el usuario

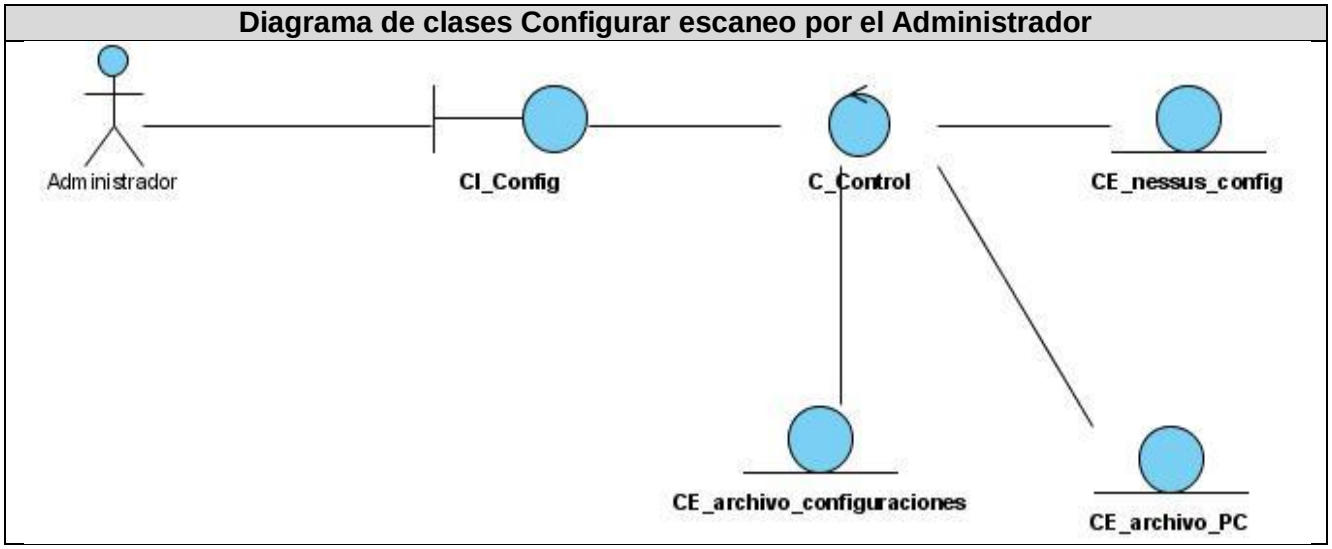


Figura 3.3 Diagrama de clases Configurar escaneo por el Administrador

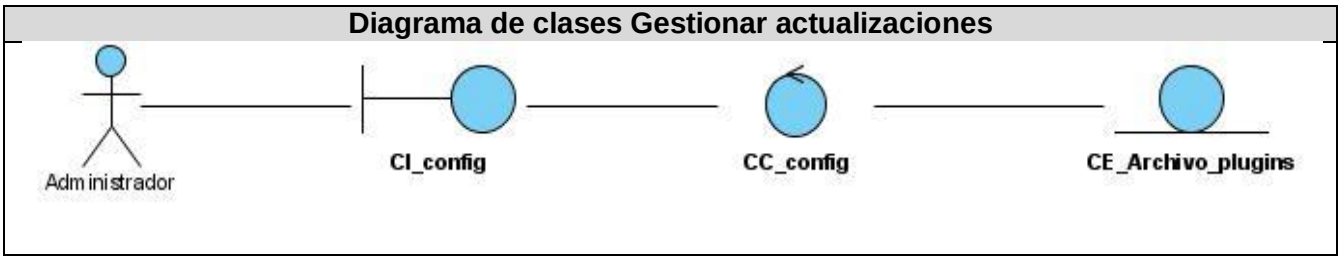


Figura 3.4 Diagrama de clases Gestionar actualizaciones

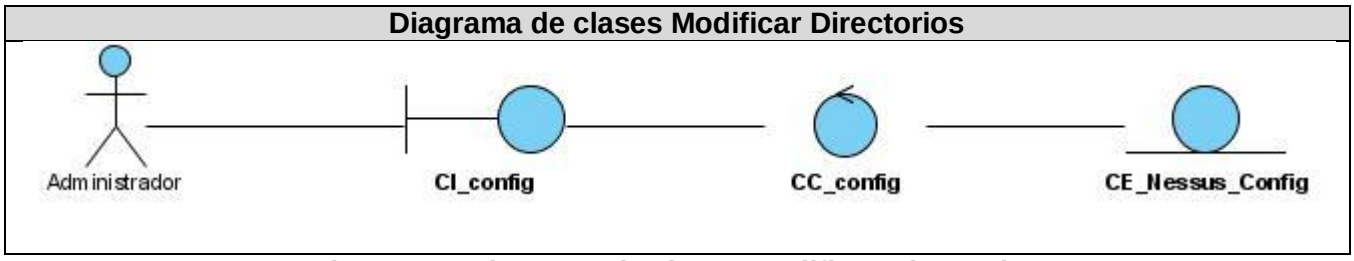


Figura 3.5 Diagrama de clases Modificar Directorios

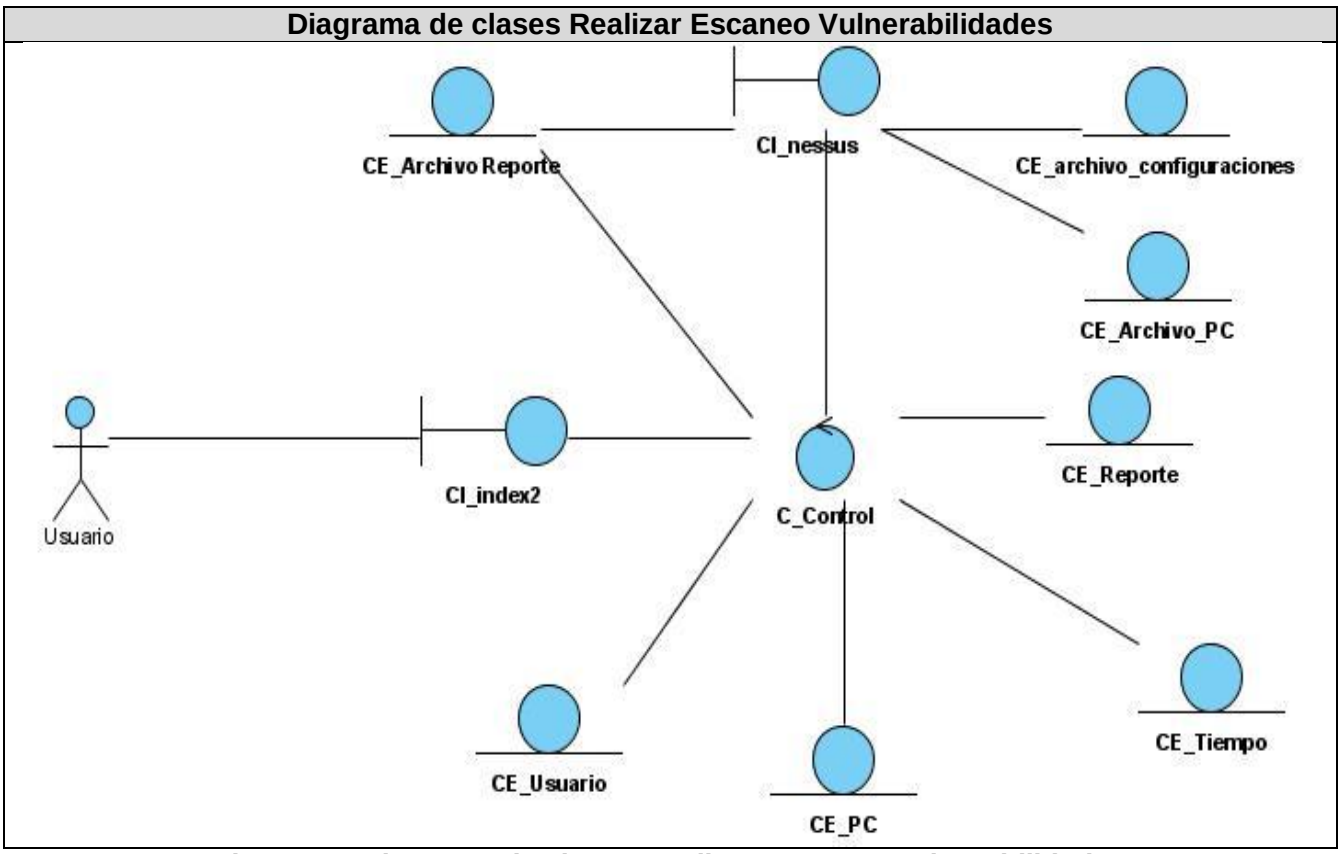


Figura 3.6 Diagrama de clases Realizar Escaneo Vulnerabilidades

3.3 Diseño.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. El diseño impone una estructura del sistema que debemos mantener lo mas fielmente posible.

3.3.1 Diagrama de clases del diseño

En el diagrama de clases podemos ver las diferentes clases que estructuran el sistema así como sus interacciones. En el diagrama se puede ver las colaboraciones que existen entre cada página donde cada una representa a una clase. Los diagramas de clase fueron definidos a partir de cada caso de uso.

Diagrama de clases Autenticar

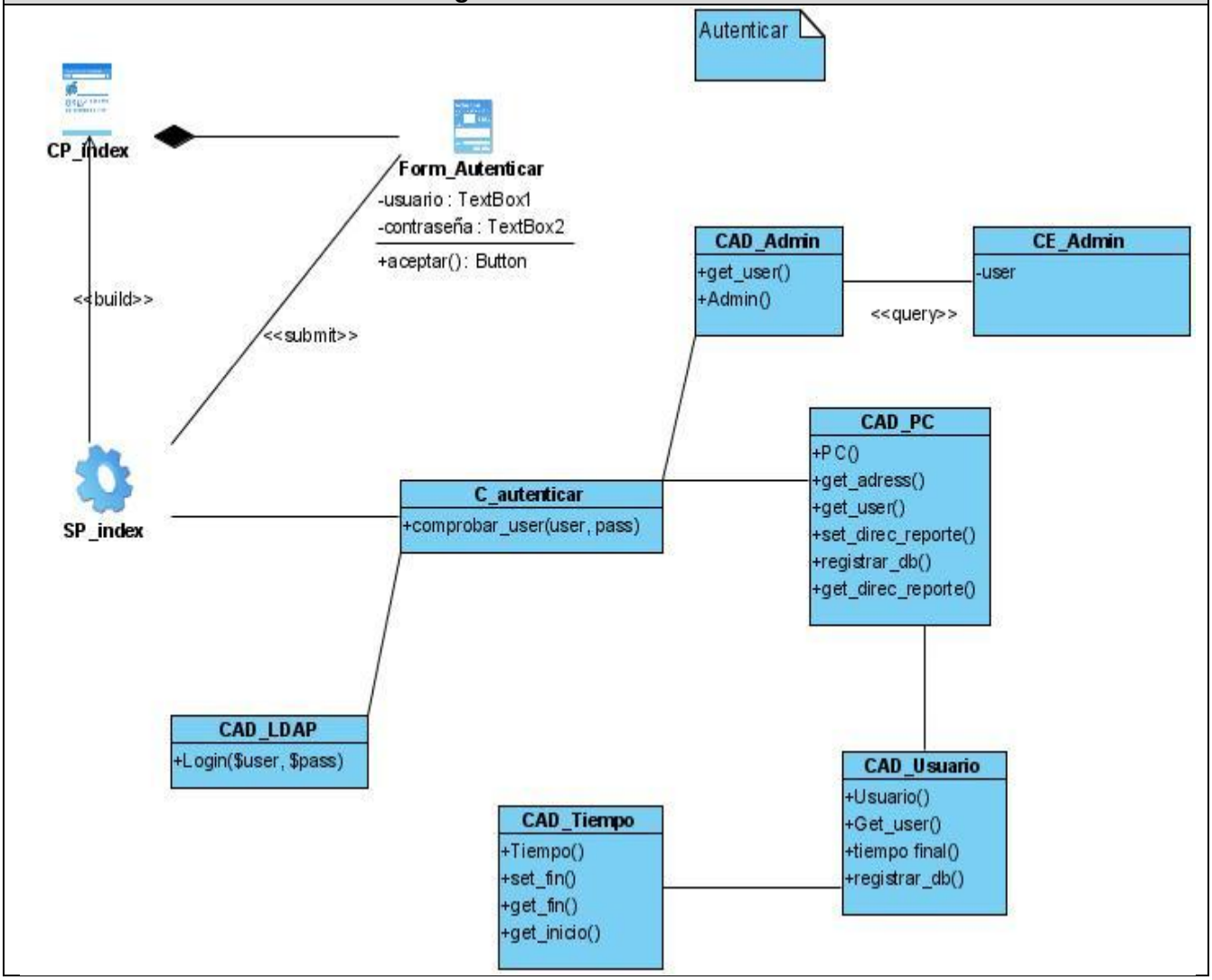


Figura 3.7 Diagrama de clases Autenticar

Diagrama de clases Configurar escaneo por el usuario

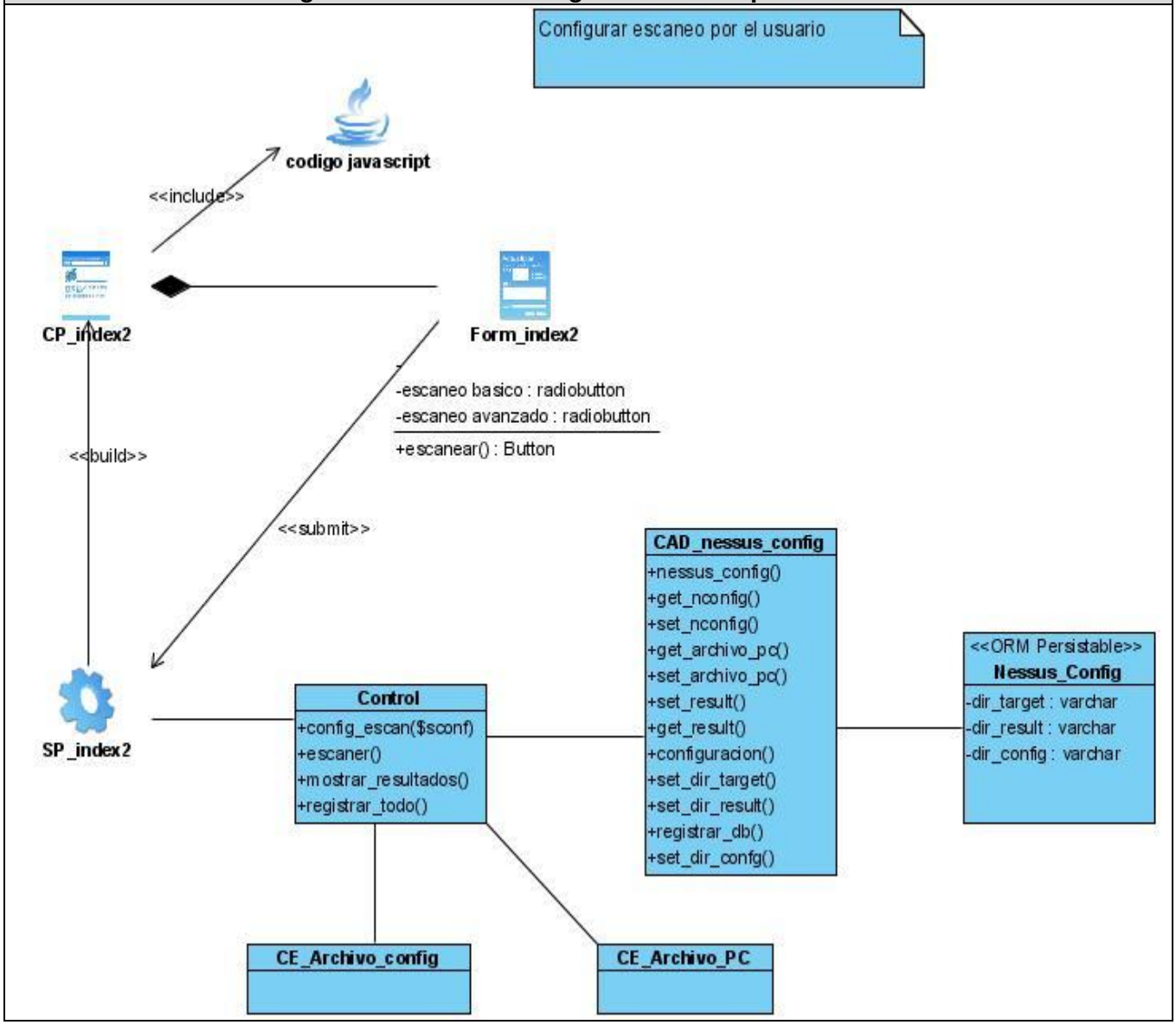


Figura 3.8 Diagrama de clases Configurar escaneo por el usuario

Diagrama de clases Configurar escaneo por el Administrador

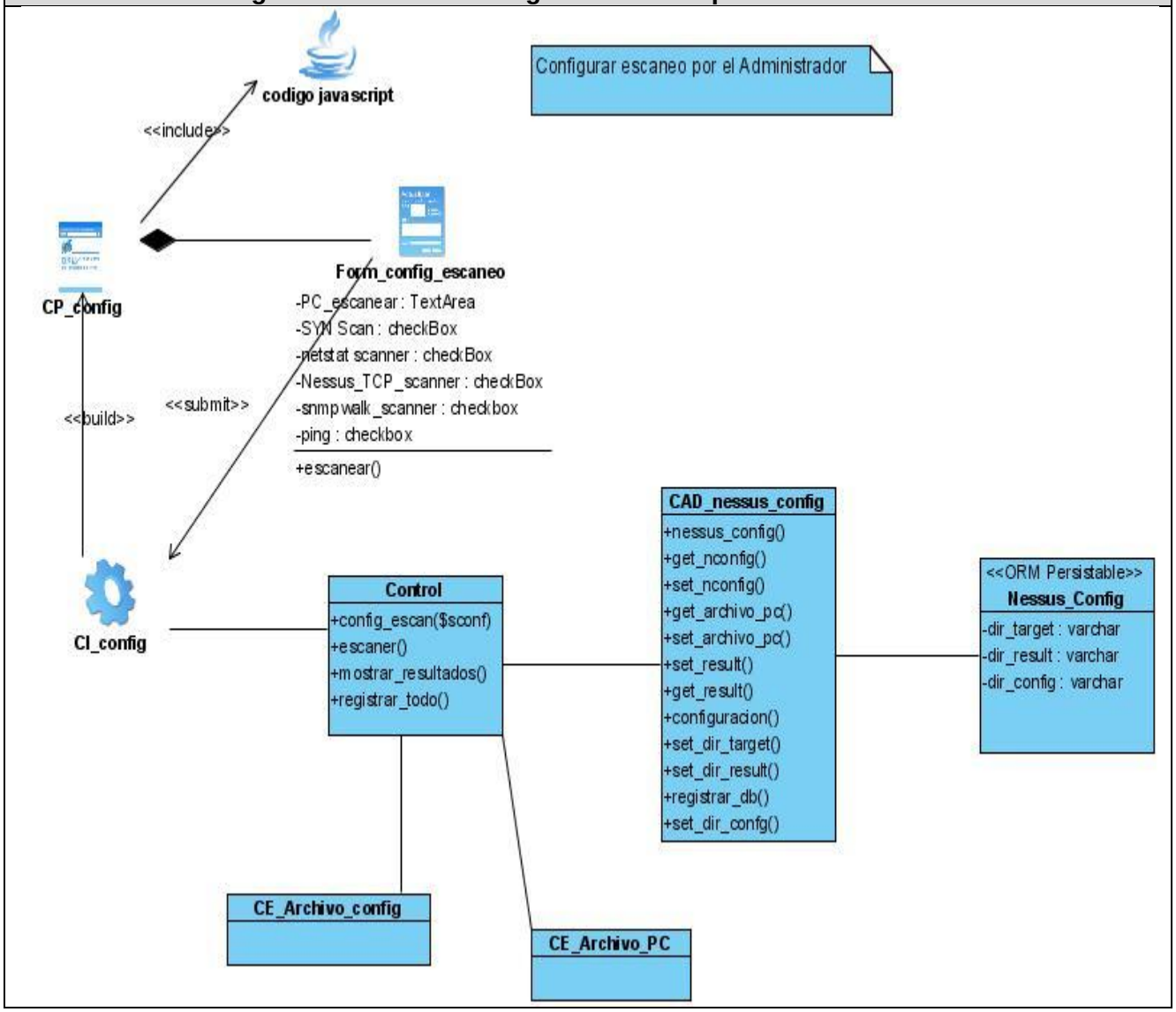


Figura 3.9 Diagrama de clases Configurar escaneo por el Administrador

Diagrama de clases Gestionar Actualizaciones

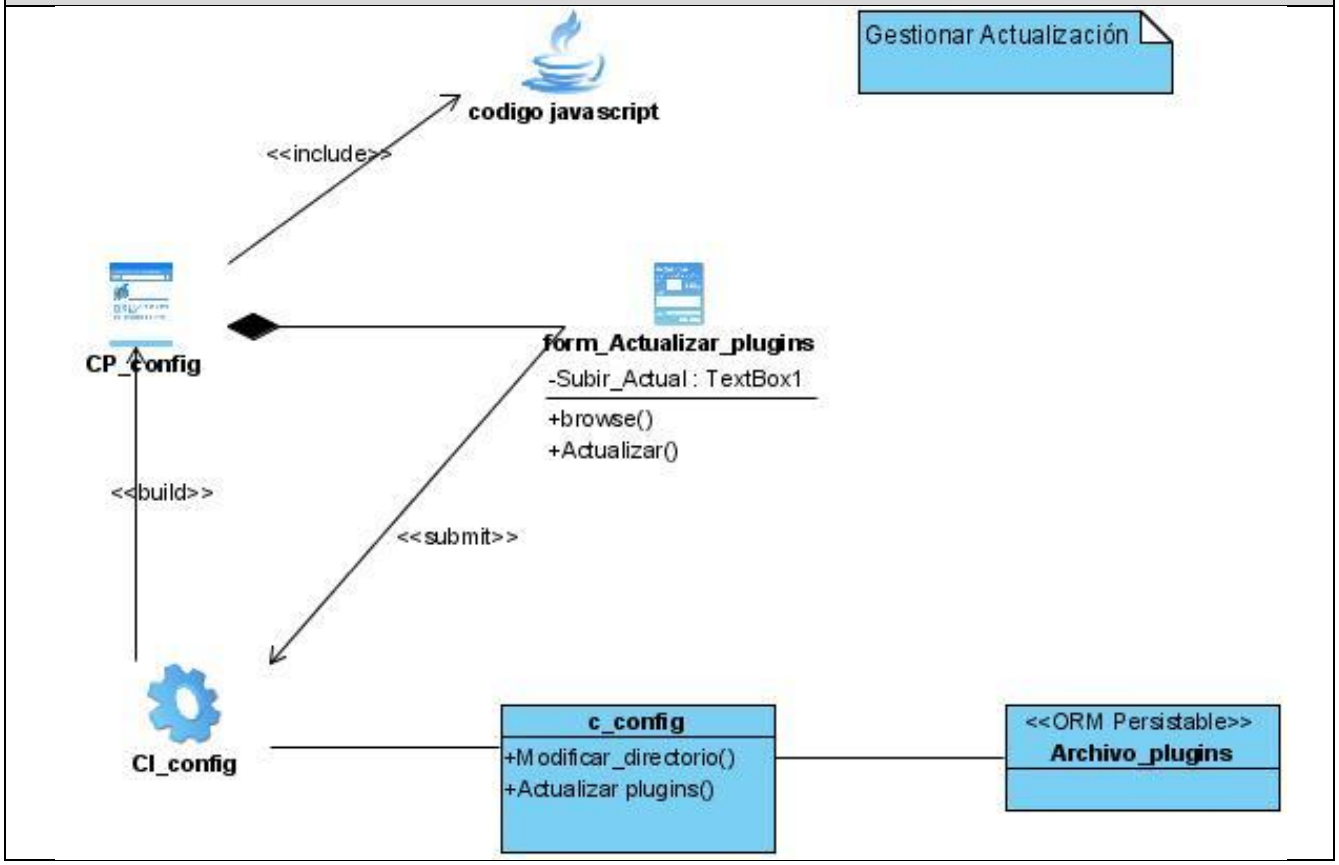


Figura 3.10 Diagrama de clases Gestionar Actualizaciones

Diagrama de clases Modificar directorios

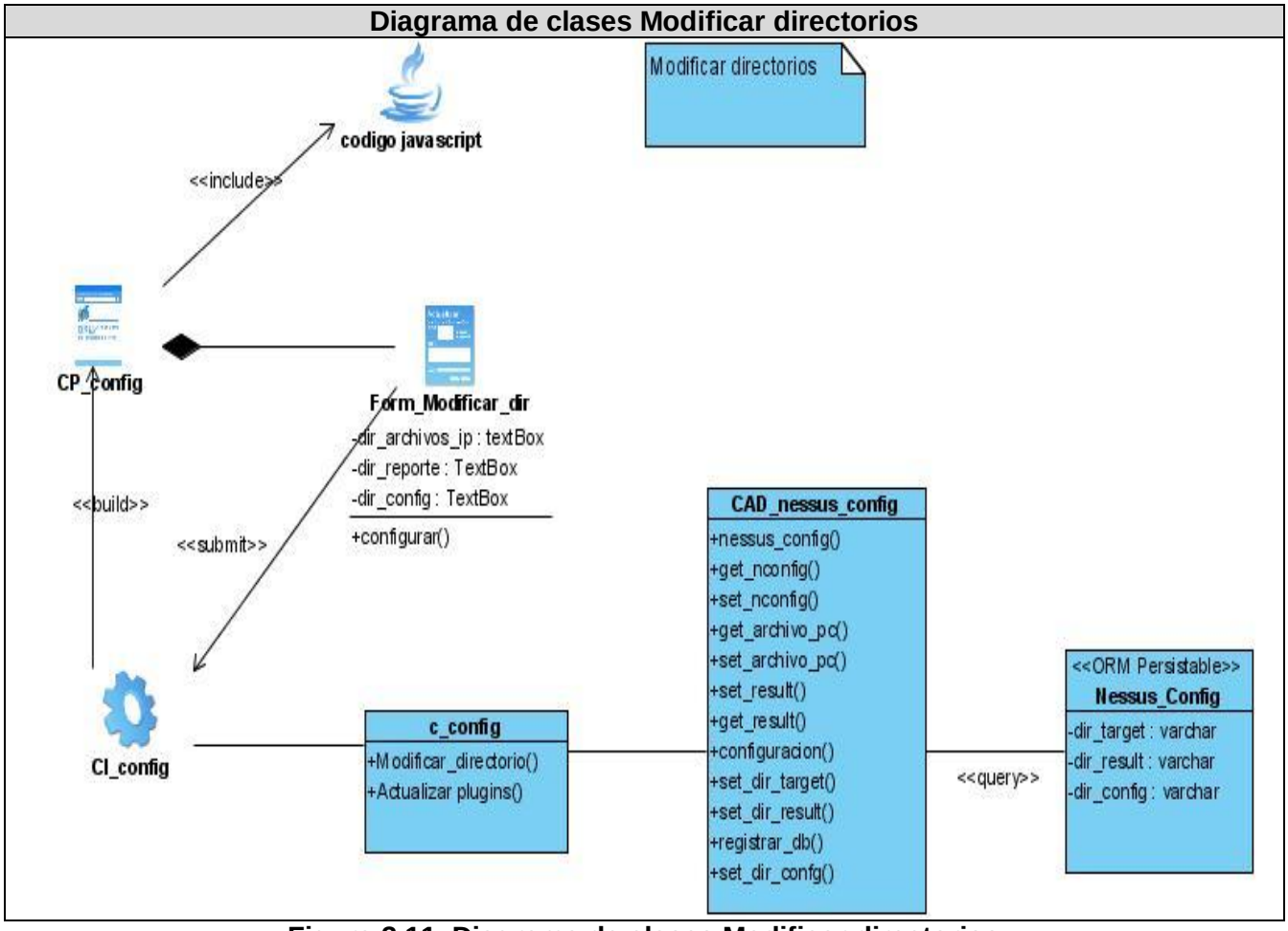


Figura 3.11 Diagrama de clases Modificar directorios

Diagrama de clases Realizar Escaneo Vulnerabilidades

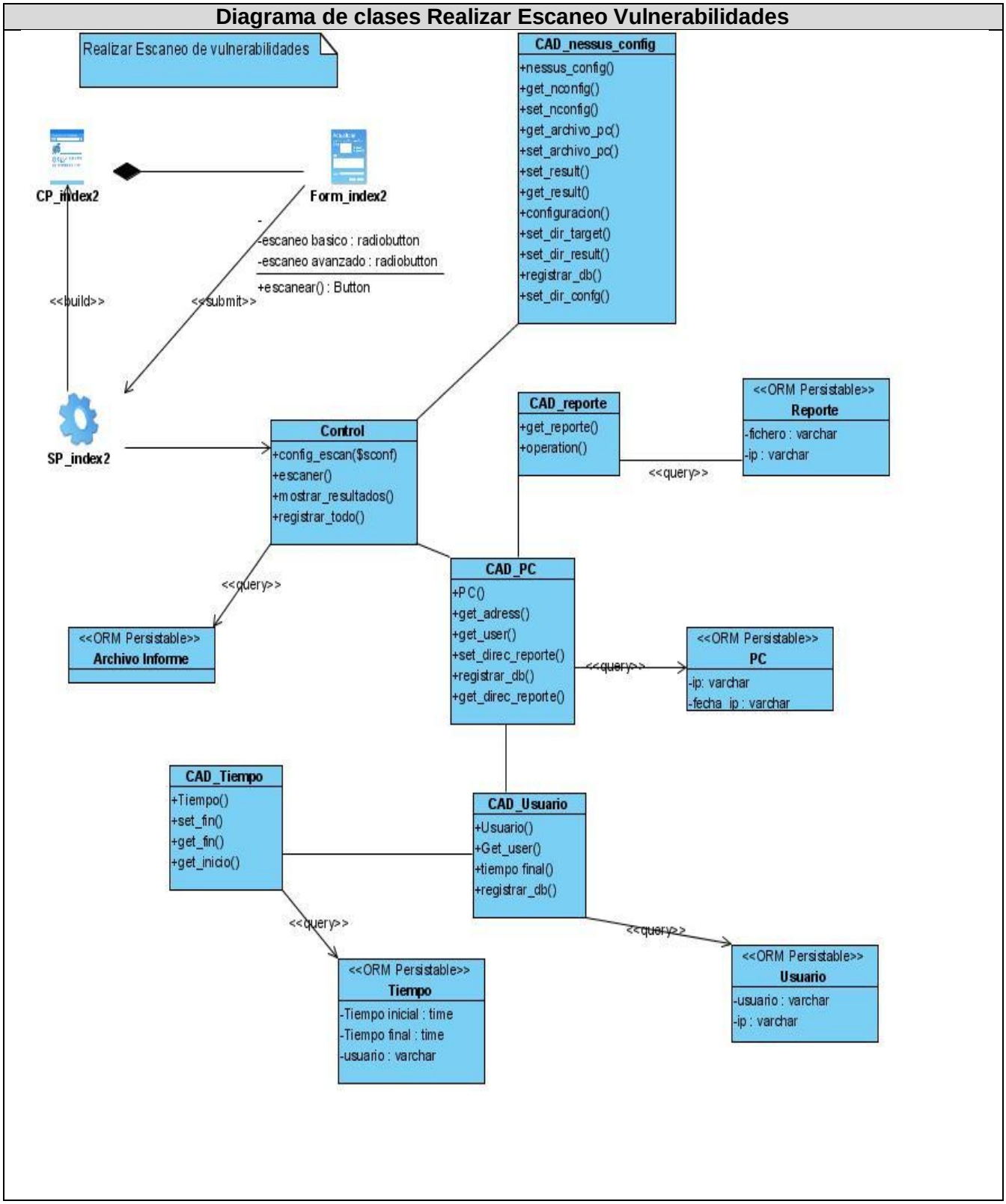


Figura 3.12 Diagrama de clases Realizar Escaneo Vulnerabilidades.

3.3 Diagramas de interacción

Los diagramas de interacción muestran la secuencia de acciones de un caso de uso en específico, permiten modelar los aspectos dinámicos del sistema. Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los tipos de diagramas de interacción son:

- **Diagramas de secuencia: En** estos se destaca el orden temporal de los mensajes como principal diferencia con respecto a los de colaboración.[VER ANEXO 4]

- **Diagramas de colaboración:** En estos se destaca la organización estructural de los objetos que envían y reciben mensajes

Diagrama de colaboración Autenticar

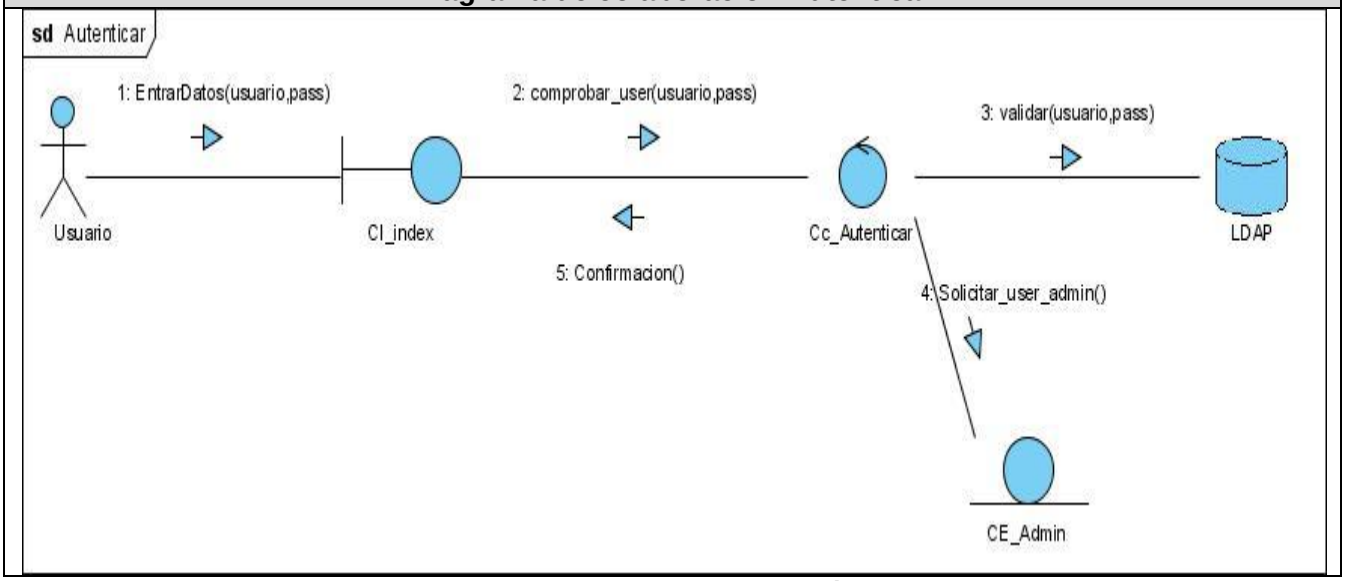


Figura 3.13 Diagrama de colaboración Autenticar

Diagrama de colaboración Configurar escaneo por el usuario

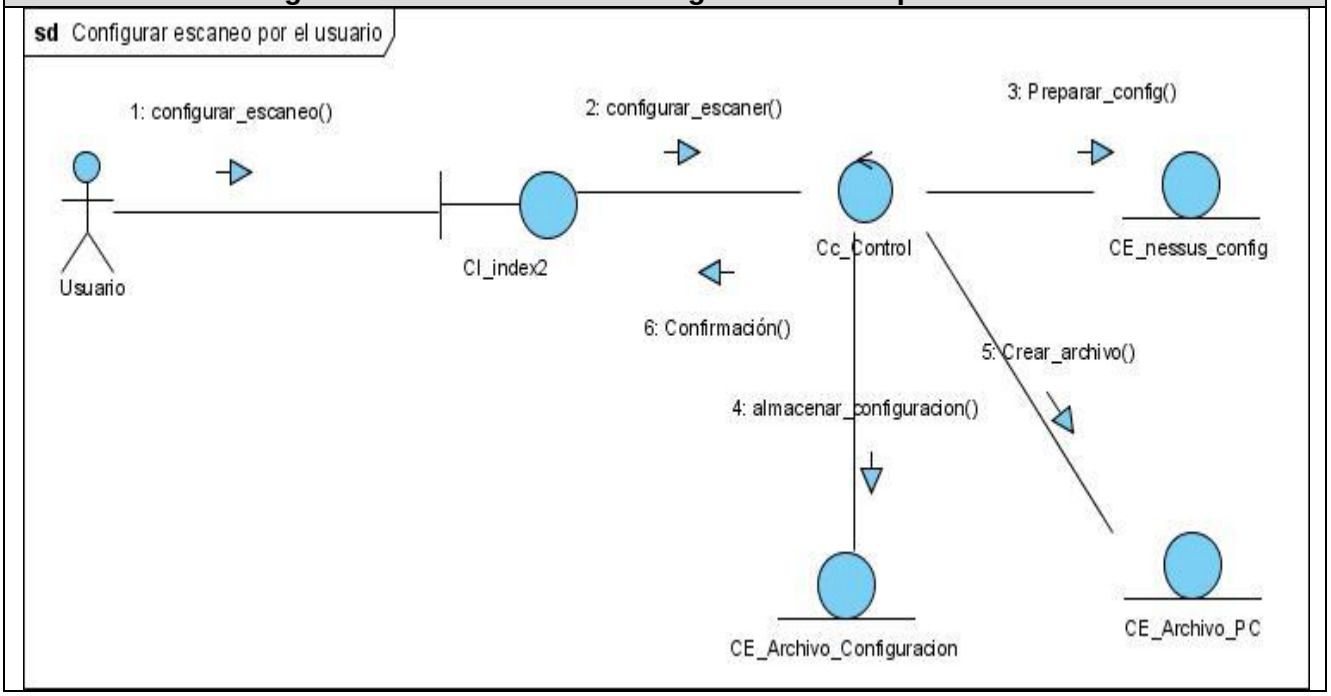


Figura 3.14 Diagrama de colaboración Configurar escaneo por el usuario

Diagrama de colaboración Configurar Escaneo por el Administrador

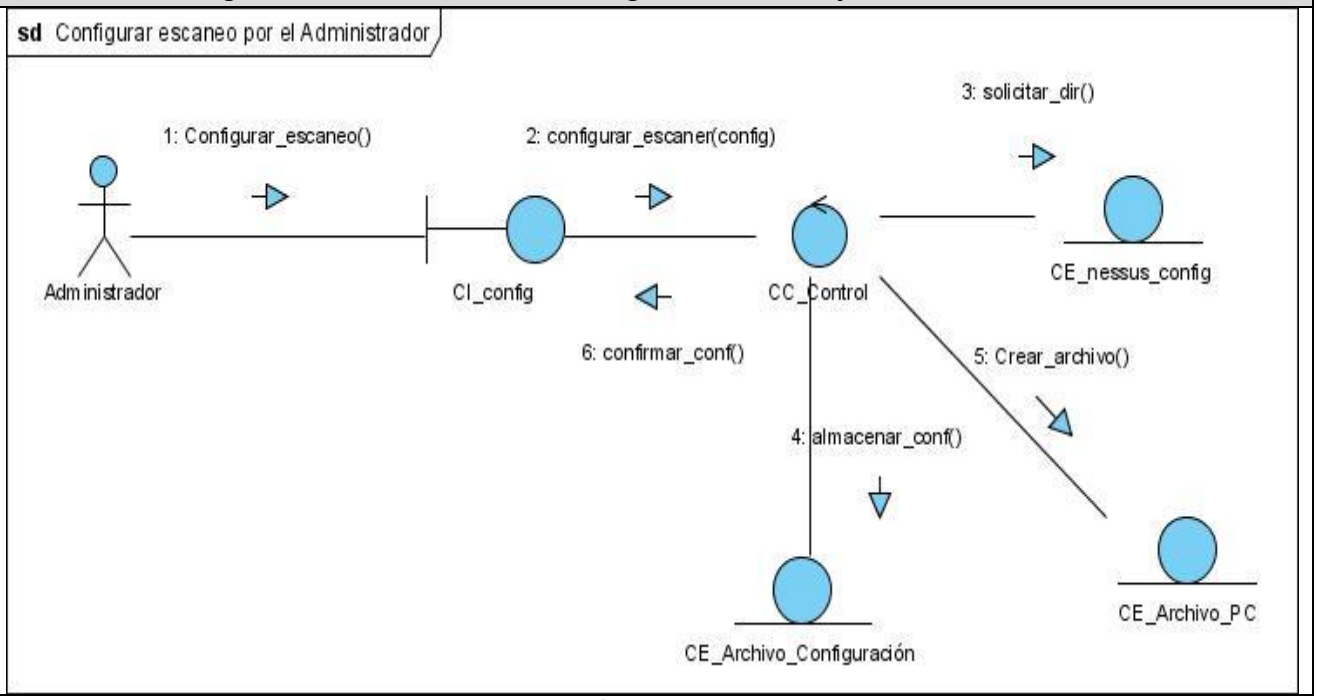


Figura 3.15 Diagrama de colaboración Configurar Escaneo por el Administrador

Diagrama de colaboración Gestionar Actualizaciones

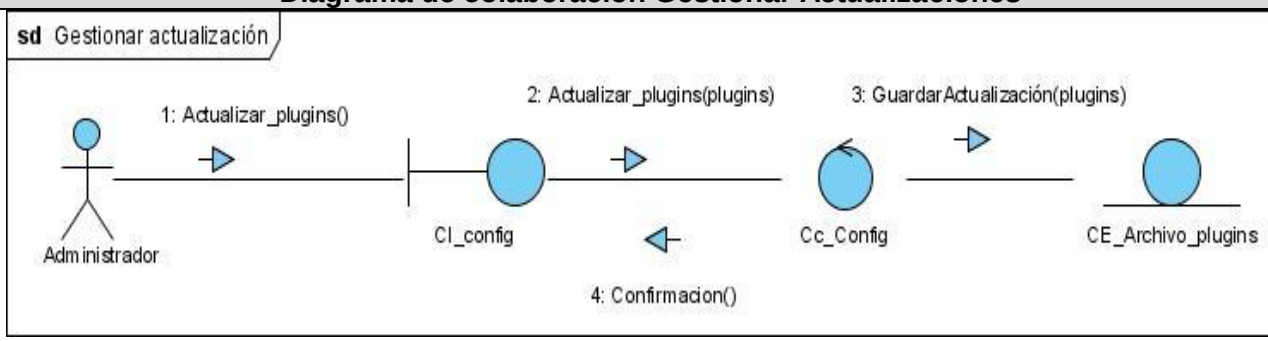


Figura 3.16 Diagrama de colaboración Gestionar Actualizaciones

Diagrama de colaboración Modificar directorios

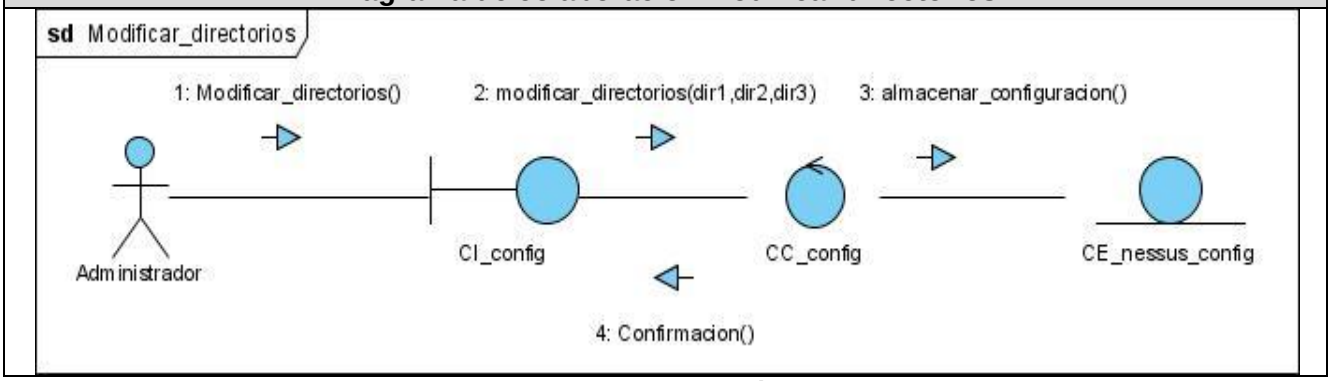


Figura 3.17 Diagrama de colaboración Modificar directorios

Diagrama de colaboración Realizar Escaneo de Vulnerabilidades

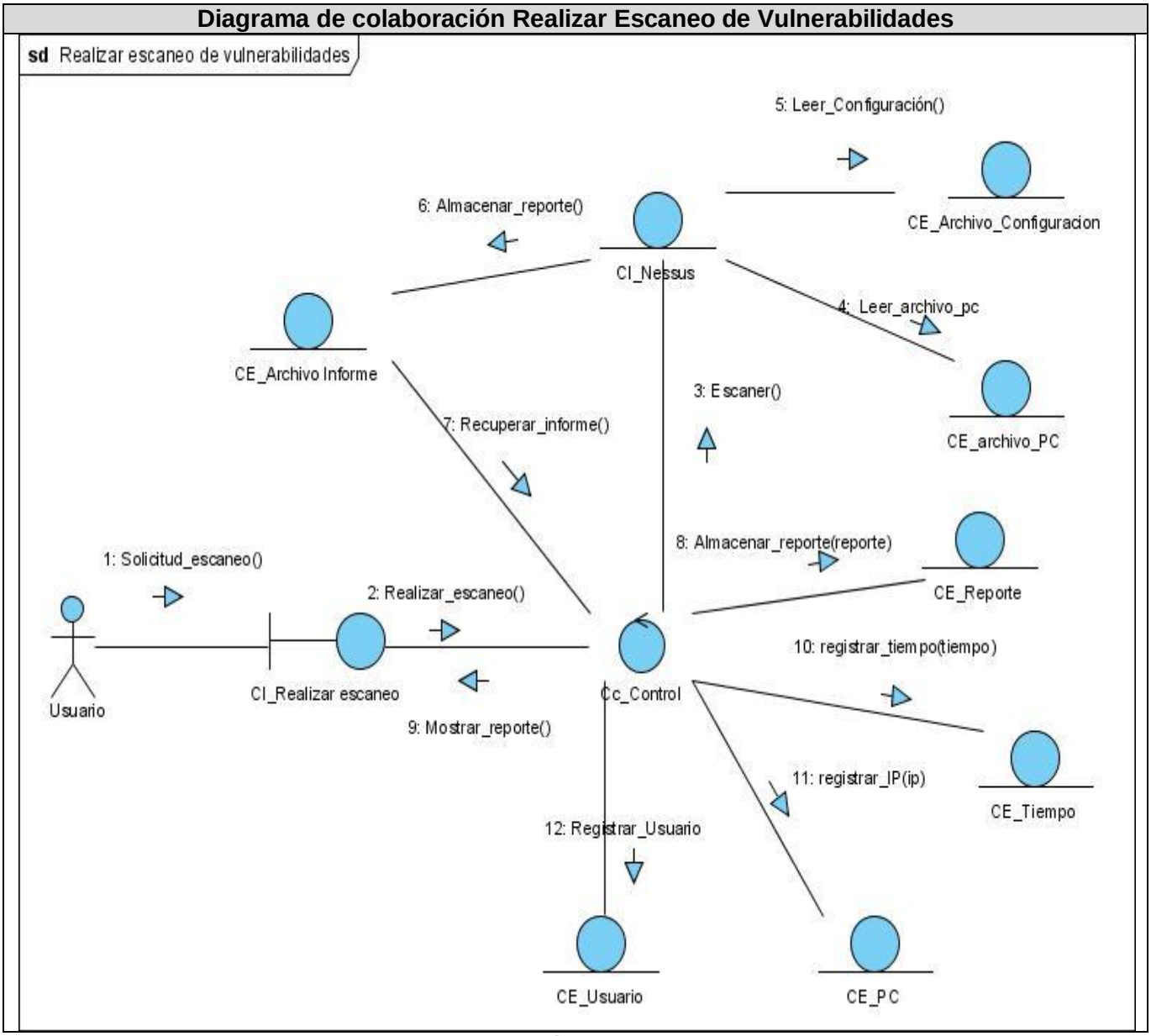


Figura 3.17 Diagrama de colaboración Realizar Escaneo de Vulnerabilidades

3.4 Diagrama de clases persistentes

Las clases persistentes son aquellas cuyos objetos deben ser almacenados en algún repositorio como una base de datos relacional. A continuación podemos ver el diagrama de clases persistente.

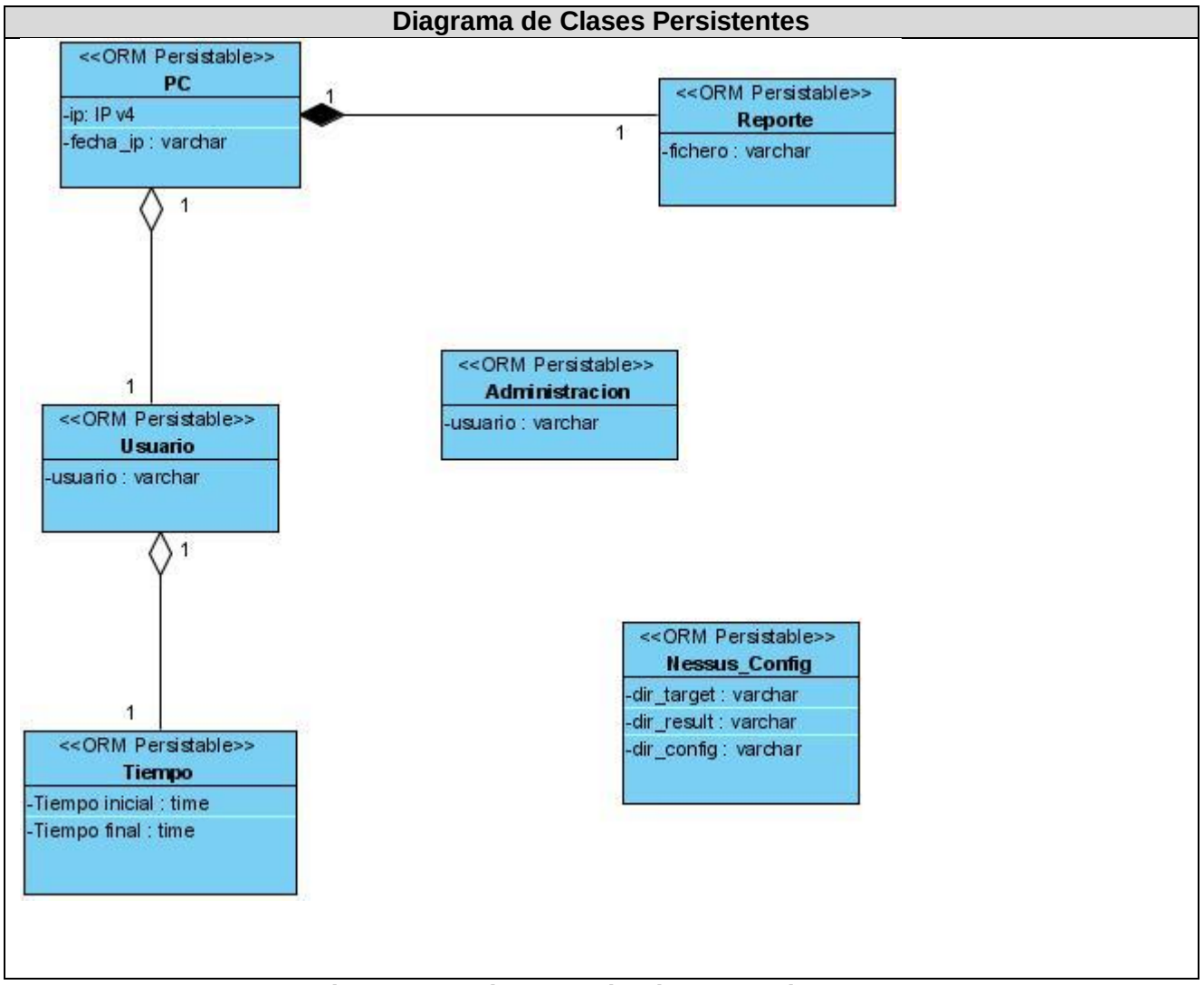


Figura 3.18 Diagrama de Clases Persistentes

3.5 Modelo de datos

Un modelo de datos describe como se representan estos de una forma abstracta en un sistema gestor de base de datos. Estos contienen objetos, atributos y relaciones. A continuación el modelo de datos de la base de datos.

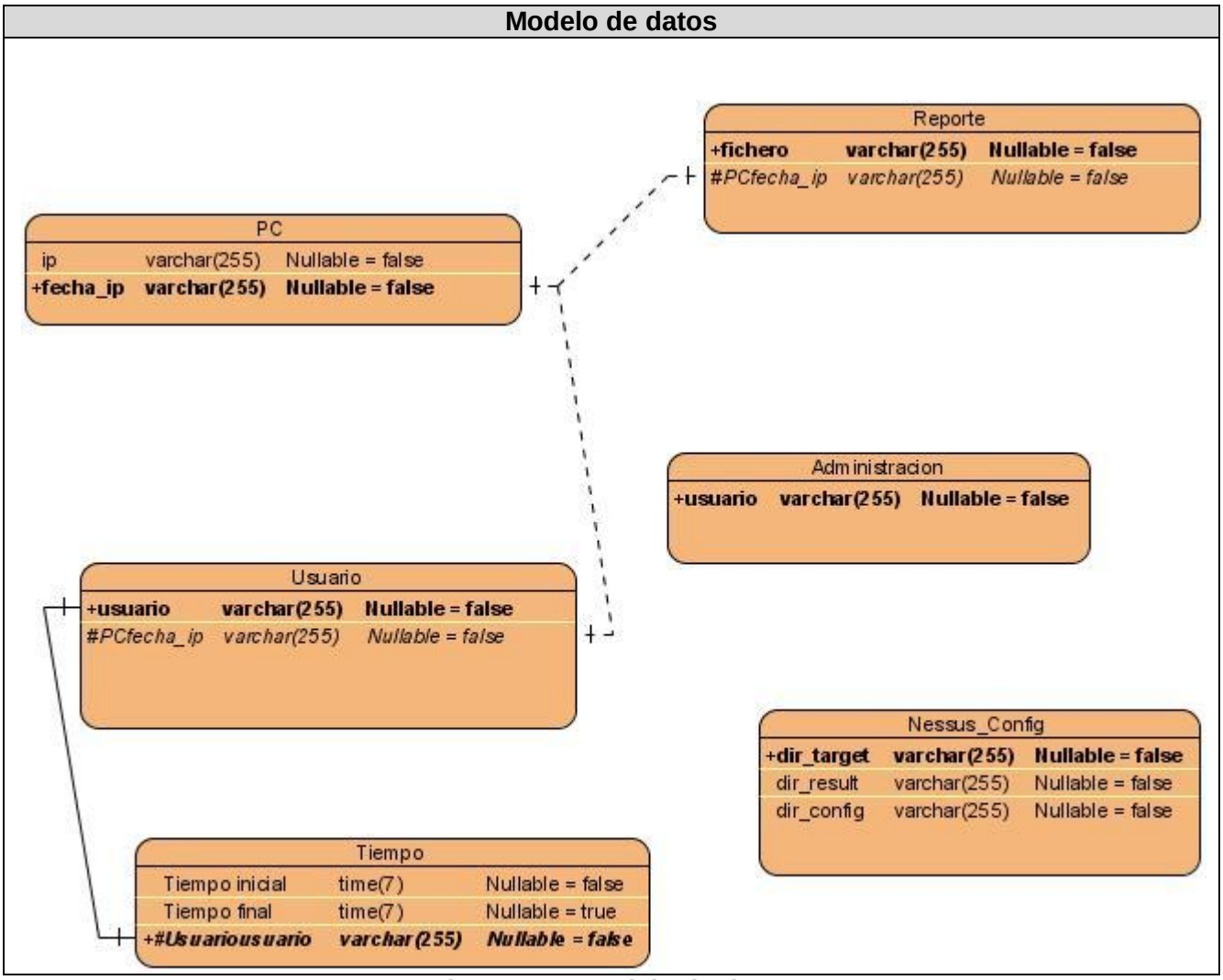


Figura 3.19 Modelo de datos

3.6 Descripción de las tablas.

Nombre: PC		
Descripción: Tabla que se encarga de almacenar el ip del usuario que desea que se le escanea su PC.		
Atributo	**Tipo**	**Descripción**
IP	Ipv4	En este atributo se almacena el ip de la

		maquina del usuario
Fecha_ip	varchar	Almacena la fecha del escaneo.

Tabla 3.1 Descripción de la tabla PC

Nombre: Reporte		
Descripción: Tabla que se encarga de almacenar el reporte por cada PC que se haya escaneado así como su dirección IP.		
Atributo	**Tipo**	**Descripción**
fichero	varchar	Aquí se almacena el reporte del escaneo de vulnerabilidad
Fecha_ip	varchar	Almacena la fecha del escaneo.

Tabla 3.2 Descripción de la tabla Reporte

Nombre: Usuario		
Descripción: Tabla que se encarga de almacenar el usuario que solicitó el escaneo		
Atributo	**Tipo**	**Descripción**
usuario	varchar	Almacena el usuario que solicitó el escaneo
Fecha_ip	varchar	Almacena la fecha del escaneo.

Tabla 3.3 Descripción de la tabla Usuario

Nombre: Tiempo		
Descripción: Tabla que se encarga de almacenar el tiempo inicial del escaneo y el tiempo final		
Atributo	**Tipo**	**Descripción**
Tiempo inicial	varchar	Almacena el tiempo inicial en que se produjo el escaneo

Tiempo final	varchar	Almacena el tiempo final del escaneo
usuario	varchar	Almacena el usuario que solicitó el escaneo

Tabla 3.4 Descripción de la tabla Tiempo

Nombre: Administración		
Descripción: Tabla que se encarga de almacenar el usuario del administrador		
Atributo	**Tipo**	**Descripción**
usuario	varchar	Almacena el usuario del administrador

Tabla 3.5 Descripción de la tabla Administración

Nombre: Nessus_Config		
Descripción: Tabla que almacena la configuración del cliente de nessus		
Atributo	**Tipo**	**Descripción**
dir_config	varchar	Almacena el directorio donde se guarda las configuraciones
dir_target	varchar	Almacena el directorio donde se guardan los ip de escaneo.
dir_result	varchar	Almacena el directorio donde se guardan los informes (reportes) emitidos.

Tabla 3.6 Descripción de la tabla Nessus_Config

3.7 Conclusiones

En este capitulo hemos abordados los diagramas del análisis y del diseño así como el modelo de datos y diagrama de clases persistentes. Todo esto permite una mejor comprensión del sistema y es un punto de partida a los desarrolladores. Además se describió la arquitectura adoptada.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En este capítulo se define la organización del código en términos de subsistemas de implementación organizados en capas. Implementaremos los elementos de diseño en términos de elementos de implementación (ficheros de código fuente, binarios, ejecutables etc.), además se prueban los componentes desarrollados y se integra los resultados producidos en un sistema ejecutable.

4.2 Modelo de despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Consiste en:

- **Nodos:** Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos
- **Dispositivos**: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela
- **Conectores**: Expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

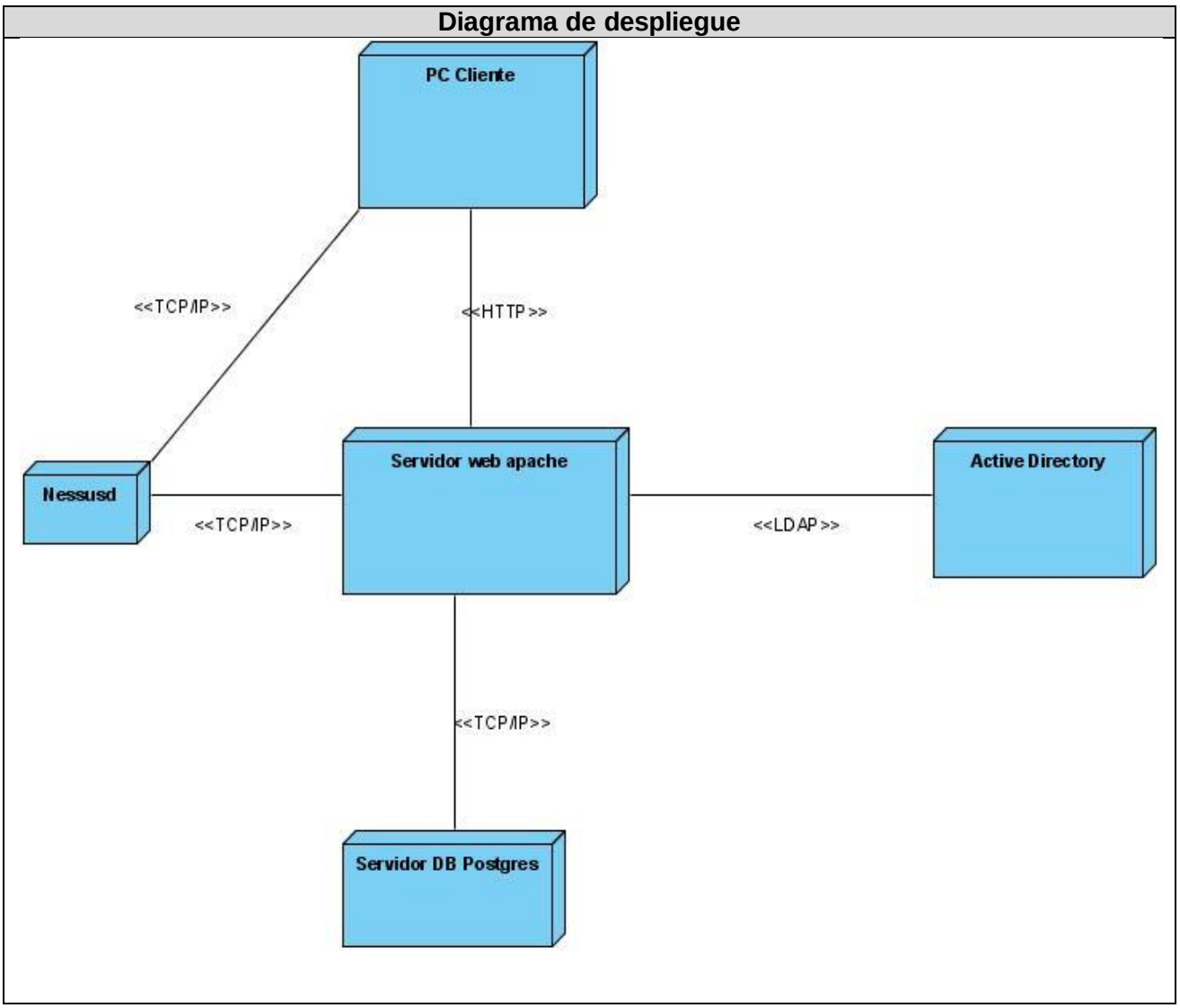


Figura 4.1: Diagrama de despliegue

4.3 Diagrama de componentes

Un diagrama de componente muestra las organizaciones y relaciones lógicas entre componentes de software. Estos componentes pueden ser código fuente, binarios, ejecutables etc.

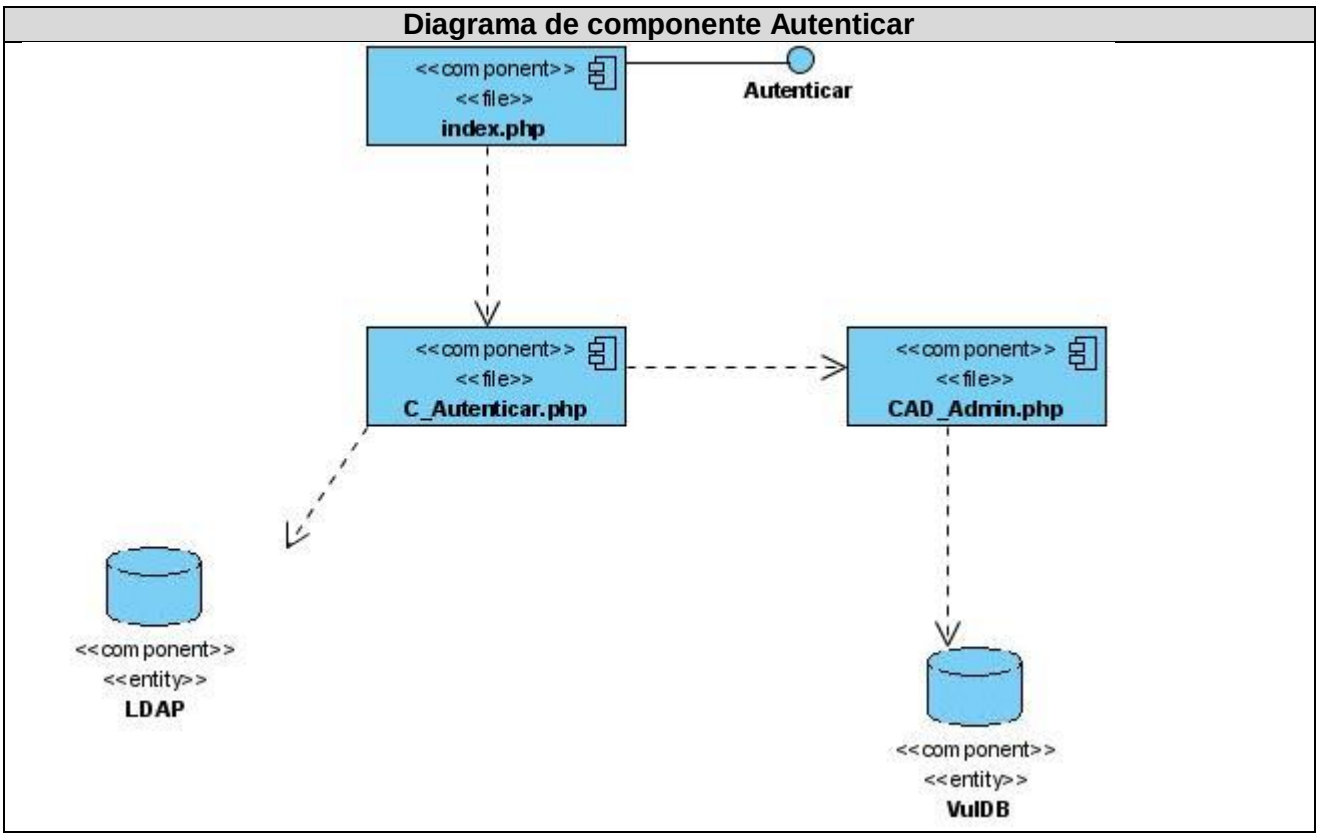


Figura 4.2 Diagrama de componente Autenticar

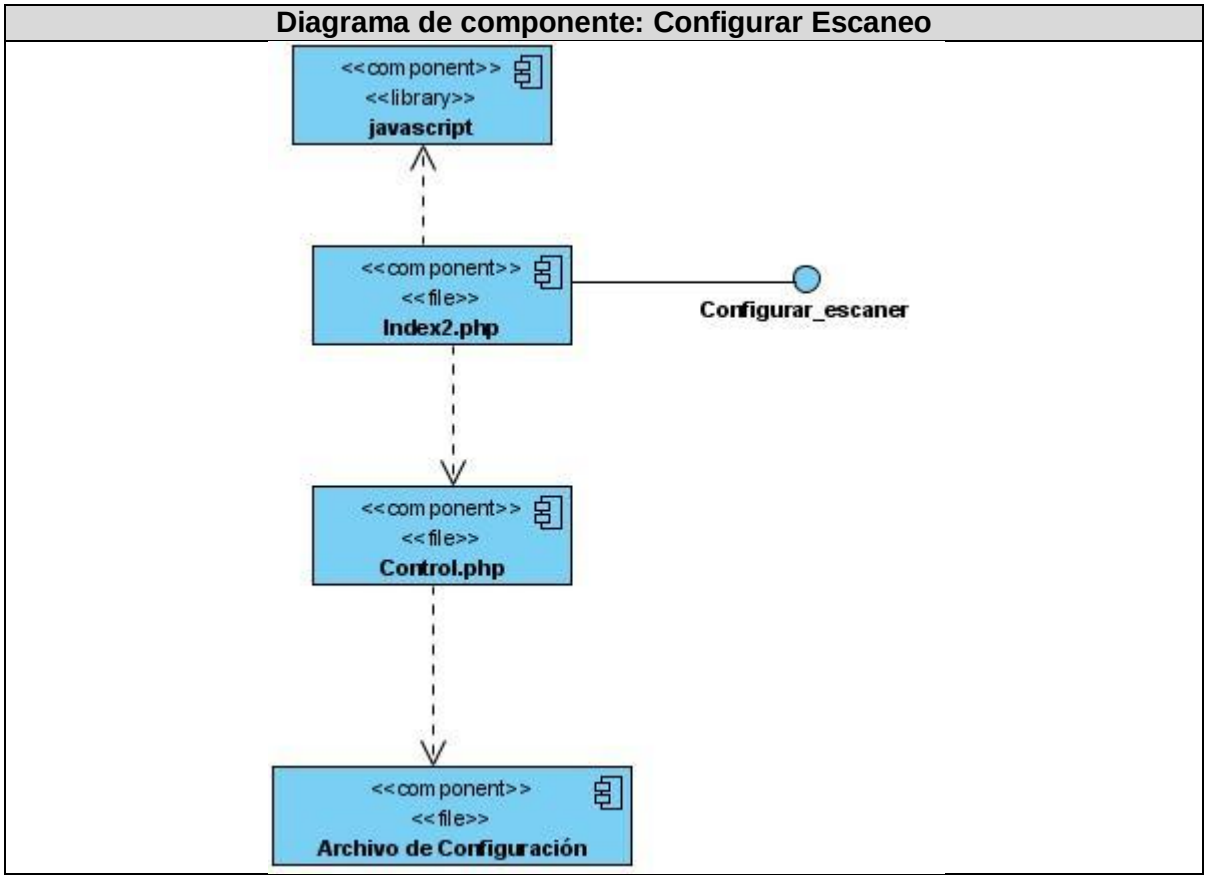


Figura 4.3 Diagrama de componente: Configurar Escaneo

Diagrama de componente: Gestionar Actualización

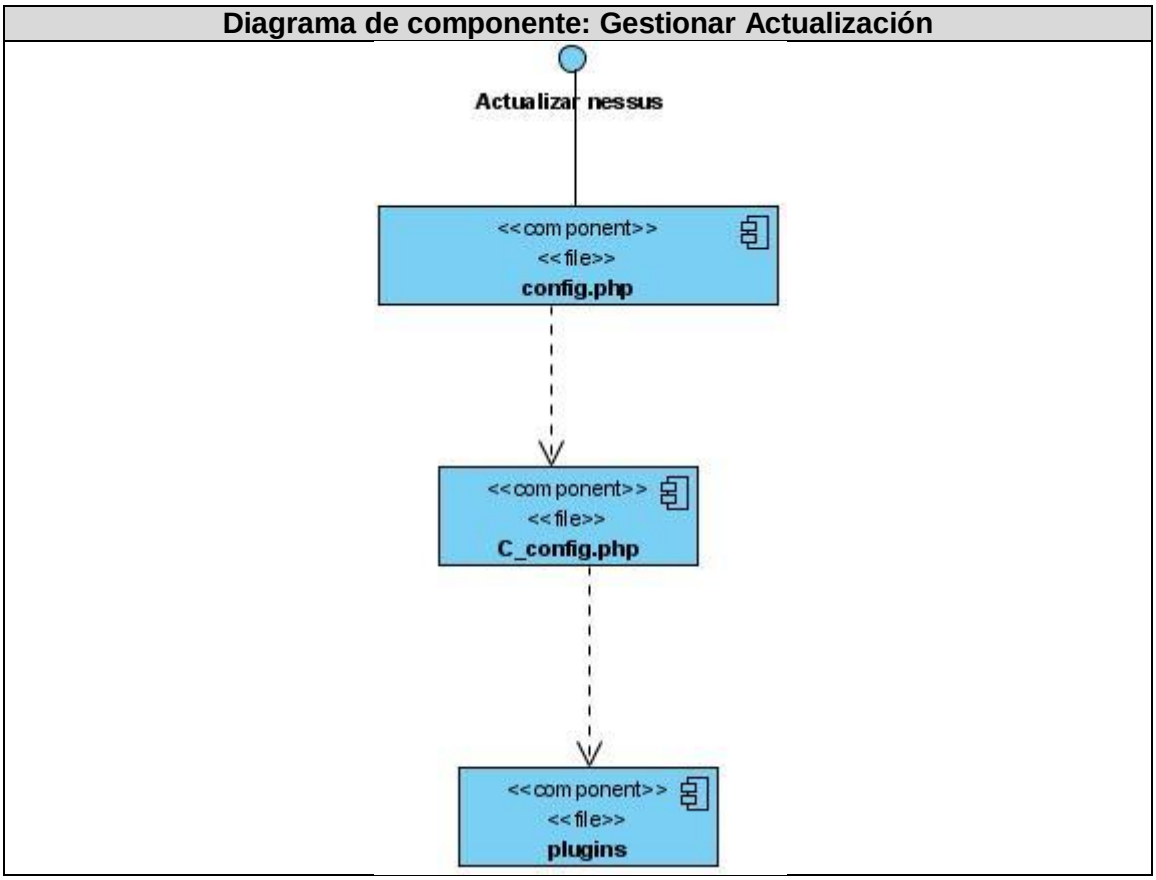


Figura 4.4 Diagrama de componente: Gestionar Actualización

Diagrama de componente: Realizar Escaneo de Vulnerabilidad

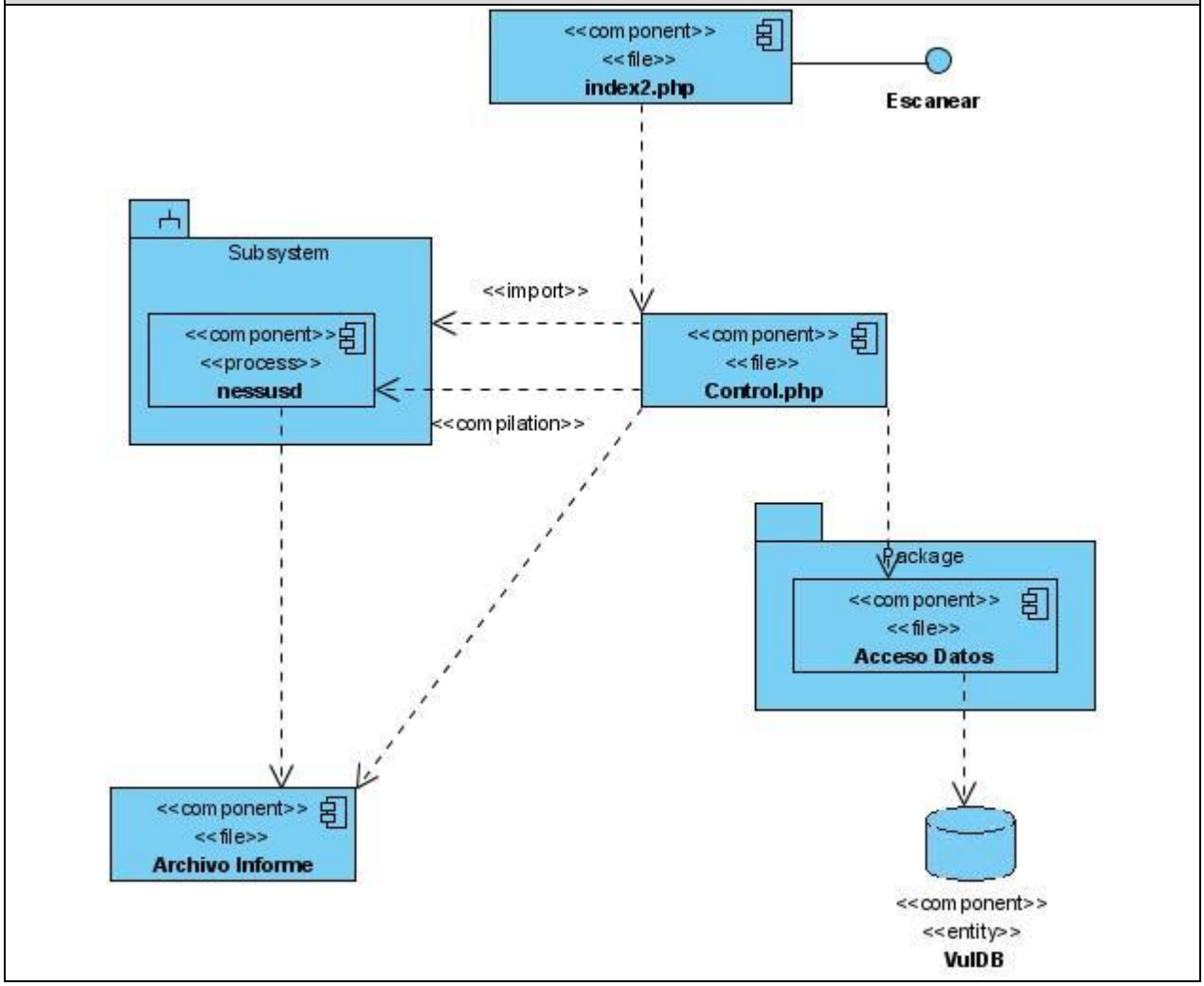


Figura 4.5 Diagrama de componente: Realizar Escaneo de Vulnerabilidad

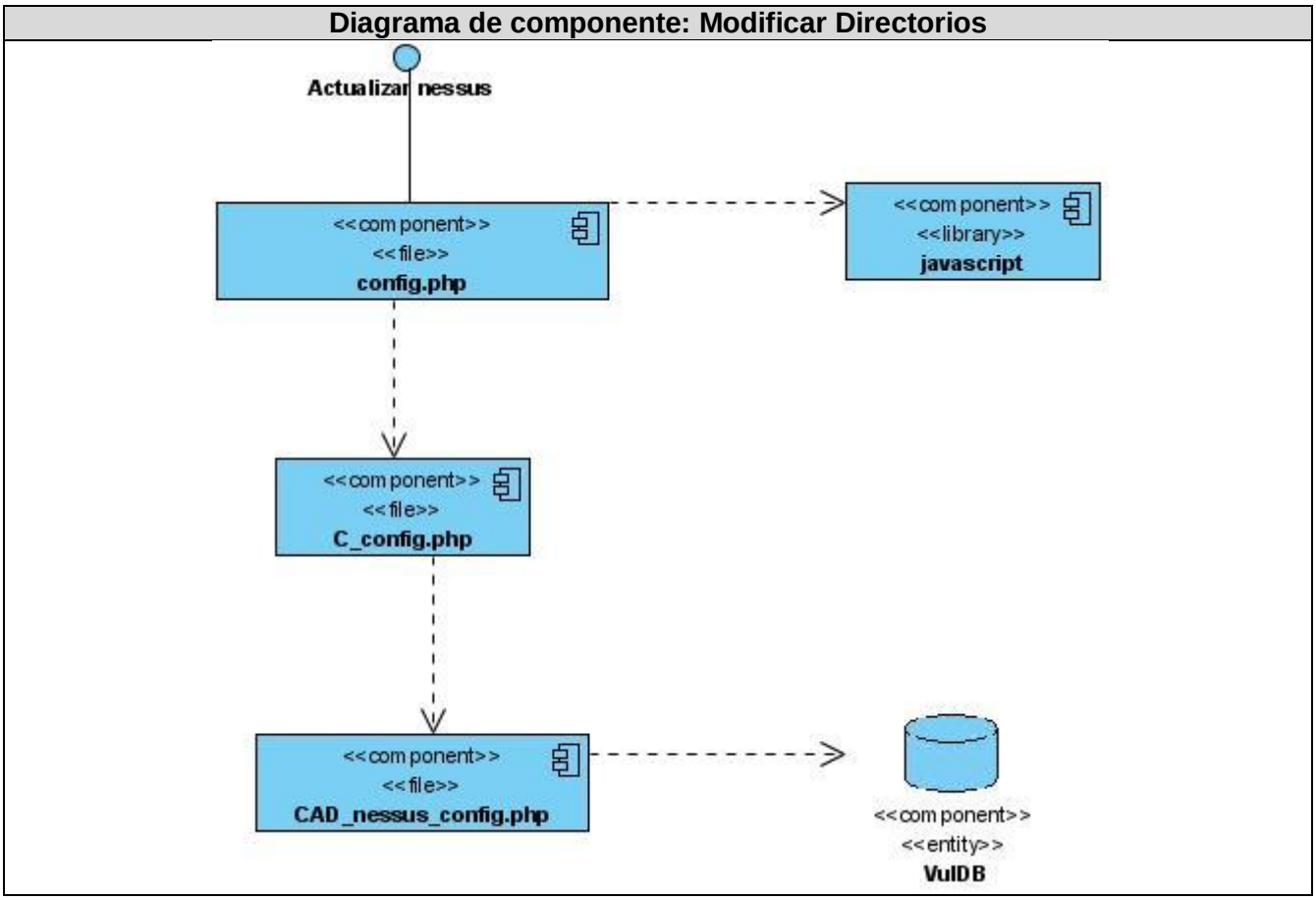


Figura 4.5 Diagrama de componente: Modificar Directorios

4.4 Conclusiones

En el presente capítulo se desarrollaron los diagramas de componentes del sistema los cuales como bien se dijo representan la separación de un sistema en componentes físicos. Además se desarrolló el diagrama de despliegue que muestra la configuración de los nodos de procesamiento.

CAPÍTULO 5: ESTIMACIÓN DEL PROYECTO

5.1 Introducción

En este capítulo se describe la estimación de costo del sistema así como el tiempo que se invierte en la realización del mismo y cuantas personas son requeridas para desarrollar el sistema. Además se determina gracias a estos resultados si es factible o no llevar a cabo la implementación del sistema.

1-Cálculo de puntos de los casos de uso sin ajustar

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar.
UAW: Factor de Peso de los Actores sin ajustar.
UUCW: Factor de Peso de los Casos de Uso sin ajustar.

Tipo de Actor	Descripción	Factor de peso	Actores	Total
Simple	Sistema con sistema a través de interfaz de programación.	1	0	0
Medio	Sistema con sistema mediante protocolo de interfaz basada en texto.	2	0	0
Complejo	Persona que interactúa con el sistema mediante una interfaz gráfica.	3	2	6

Tabla 5.1: Factor de Peso de los Actores sin ajustar.

UAW = $\sum$ (cant actores * peso)
UAW =6

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	4	20
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	0	0
Complejo	El caso de uso tiene más de 8 transacciones.	15	0	0

Tabla 5.2: Factor de Peso de los Casos de Uso sin ajustar.

UUCW = $\sum$ (cant CU * peso)

UUCW =20

Los puntos de los casos de uso sin ajustar
UUCP = UAW + UUCW= 6 + 20 = 26
2- Cálculo de puntos de los casos de uso ajustados
UCP: Puntos de Casos de Uso ajustados
UUCP: Puntos de Casos de Uso sin ajustar
TCF: Factor de complejidad técnica
EF: Factor de ambiente

Ajustando el valor de los caso de usos, mediante esta fórmula
UCP = UUCP x TCF x EF

Factor	Descripción	Peso	Valor asignado	Total
T1	Sistema distribuido.	2	4	8
T2	Objetivos de performance o tiempo de respuesta	1	5	5
T3	Eficiencia del usuario final.	1	1	1
T4	Procesamiento interno complejo.	1	1	1
T5	El código debe ser reutilizable.	1	5	5
T6	Facilidad de instalación.	0.5	3	1.5
T7	Facilidad de uso.	0.5	3	1.5
T8	Portabilidad	2	2	4
T9	Facilidad de cambio.	1	4	4
T10	Concurrencia.	1	2	2
T11	Incluye objetivos especiales de seguridad.	1	1	1
T12	Provee acceso directo a terceras partes	1	5	5
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	3	3

Tabla 5.3: Factor de Complejidad Técnica.

TCF = 0.6 + 0.01 x Σ (Pesoi x **Valor asignadoi) =** 0.6 + 0.01 * 42 = 1.02

Factor	Descripción	Peso	Valor asignado	Total
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	1	1.5
E2	Experiencia en la aplicación.	0.5	0	0
E3	Experiencia en la orientación a objetivos.	1	3	3
E4	Capacidad del analista líder.	0.5	1	0.5
E5	Motivación.	1	5	5

E6	Estabilidad de requerimientos.	2	3	6
E7	Personal Part-Time.	-1	0	0
E8	Dificultad del lenguaje de programación.	-1	3	-3

Tabla 5.4: Factor de Ambiente.

EF =1.4 - 0.03 x Σ (Pesoi x **Valor asignadoi)** = 1.4 – 0.03 x 13 = 0.61

Finalmente, los Puntos de Casos de Uso ajustados resultan:

UCP = 26 * 1.02 * 0.61= 16.17

3-Estimación de esfuerzo a través de los puntos de casos de uso.

Para obtener el factor de conversión (CF) se cuentan cuantos valores de los que afectan el factor ambiente (E1...E6) están por debajo de la media (3), y los que están por arriba de la media para los restantes (E7, E8). Si el total es 2 o menos se utiliza el factor de conversión 20 Horas-Hombre / Punto de Casos de uso. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de Casos de uso. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

E = UCP x CF donde :
UCP: Puntos de Casos de Uso ajustados.
CF: Factor de conversión.
E = 16.17 * 28 = 452.76 Horas-Hombre

4-Calcular esfuerzo de todo el proyecto.

Actividad	Porcentaje %	Horas-Hombre
Análisis	10	113.19
Diseño	20	226.38
Implementación	40	452.76
Pruebas	15	169.79
Sobrecarga(otras actividades)	15	169.79
Total	100	1131.9

Tabla 5.5: Esfuerzo del proyecto

Si $\mathbf{E_T}$ **= 1131.9horas-hombre** y se estima que cada mes tiene como promedio 192 horas laborables, eso daría un $\mathbf{E_T}$ **= 5.89 mes-hombre.**
Esto quiere decir que 1 persona puede realizar el problema analizado en 6 meses aproximadamente.

5-Costo del Proyecto.

Se asume como salario promedio mensual $100.00

CH: Cantidad de hombres.

Tiempo: Tiempo total del proyecto.

CH: = 2 hombres

CHM = 2 * Salario Promedio = $ 200.00/mes
Costo = CHM * E_T / CH = 200.00 * 5.89 / 2 = $ 589
Tiempo = E_T / CH = 5.89 / 2 = 2.95 meses-hombre

De los resultados obtenidos se interpreta que con 2 hombres trabajando en el proyecto el mismo se desarrolla en 3 meses aproximadamente y su costo total se estima que sea $ 589

5.2 Beneficios tangibles e intangibles.

El sistema "Diagnóstico de vulnerabilidades vía web" no es un producto con fines comerciales su principal objetivo es que sea capaz de diagnosticar las vulnerabilidades de las computadoras personales o servidores.

El beneficio de esta aplicación web es que permite a los usuarios conectarse a ella y obtener un reporte (informe) de las vulnerabilidades encontradas en su PC, permitiendo encontrar en ese informe puertos abiertos, aplicación que esté usando dicho puerto, versión de la aplicación, vulnerabilidades encontradas para dicha aplicación, posibles soluciones a dicha vulnerabilidad.

5.3 Análisis de costos y beneficios.

Desarrollar un sistema cuesta. De acuerdo a los beneficios antes expuestos se llevaría acabo o no la implementación e implantación del sistema. El sistema permitirá a los usuarios que se conecten a él emitir un reporte de las vulnerabilidades encontradas en su PC, permitiendo al mismo mantenerse informado y estar al tanto de las brechas que están alojadas en su sistema, mucho de estos sistemas pertenecen a proyectos productivos por lo tanto mantener la seguridad de estos sistemas es primordial.

La tecnología empleada en este sistema es totalmente libre, no se incurre en el pago de licencia, por lo tanto el costo del mismo es bajo.

Analizando lo antes mencionado se puede concluir que la implantación del sistema es totalmente factible.

5.4 Conclusiones.

En este capítulo se hizo una estimación del costo del proyecto así como un análisis del costo y beneficio del mismo. Se concluyó que el proyecto traería un costo de $ 589 y que se desarrollaría en 3 meses aproximadamente trabajando dos hombres. Además se pudo concluir que es factible la implantación del sistema ya que los beneficios son favorables y los costos son muy bajos, ya que la tecnología empleada por el sistema es totalmente libre.

CONCLUSIONES

En este trabajo de diploma se realizó un estudio sobre los Escáneres de Vulnerabilidad más utilizados a nivel mundial, se profundizó en sus características principales y sus funcionamientos, obteniéndose una buena documentación sobre ellos. Se efectuó una selección del mejor Escáner de Vulnerabilidad a utilizar para el desarrollo de nuestro sistema y la forma en que se iba a realizar el sistema para cumplir con los requerimientos propuestos. Se realizó un proceso de Ingeniería de Software usando la metodología de desarrollo del Proceso Unificado de Software (RUP). Se logró un acoplamiento satisfactorio entre las herramientas lográndose la culminación del sistema.

RECOMENDACIONES

A partir de las conclusiones del trabajo, se proponen algunas recomendaciones que permitan dar un seguimiento y mejora al trabajo. Entre las que se encuentran las siguientes:

- Estudiar y analizar la configuración de Nessus para agregar nuevas funcionalidades al sistema de configuración de escaneo y configuración de pruebas de vulnerabilidad.
- Realizar una gestión de los reportes de vulnerabilidad que le permitan al administrador realizar determinados informes.
- Crear un módulo que permita usar la herramienta Nmap para crear informe sobre los puertos de las PCs.

BIBLIOGRAFÍA

[1]—Sitio de La Real Academia Española [Consultado: Enero 2008] Disponible en: http://www.rae.es

[2]-- WikiProd, Wikipedia de Producción [Consultado: Enero 2008] Disponible en: http://Wiki.prod.uci.cu

[3]-- HUERTA, Antonio Villalón. "Seguridad en Unix y Redes". Versión 1.2 Digital - Open Publication License v.10 o Later. 2 de Octubre de 2000. http://www.kriptopolis.org

[4]-- Segu-info.com.ar "Seguridad de la Información" [Consultado: Enero 2008] Disponible en: http://www.segu-info.com.ar/fisica/seguridadfisica.htm

[5]—Wikipedia [Consultado: Enero 2008] Disponible en: http://es.wikipedia.org

[6]-- Insecure.org, Top 10 Vulnerability Scanners. [Consultado: Noviembre 2007] Disponible en: http://sectools.org/vuln-scanners.html

[7]— Sitio de Nessus [Consultado: Noviembre 2007] Disponible en: http://www.nessus.org

[8]— Sitio de GFI LANguard [Consultado: Noviembre 2007] Disponible en: http://www.gfi.com

[9]— Sitio de Retina [Consultado: Noviembre 2007] Disponible en: http://www.eeye.com

[10]-- Sitio de Core Impact [Consultado: Noviembre 2007] Disponible en: http://www.coresecurity.com

[11]-- Sitio de ISS Internet Scanner [Consultado: Noviembre 2007] Disponible en: http://www.iss.net

[12]-- Sitio de X-scan [Consultado: Noviembre 2007] Disponible en: http://www.xfocus.net

[13]-- Sitio de Sara [Consultado: Noviembre 2007] Disponible en: http://www-arc.com

[14]-- Sitio de QualysGuard [Consultado: Noviembre 2007] Disponible en: http://www.qualys.com/

[15]-- Sitio de Saint [Consultado: Noviembre 2007] Disponible en: http://www.saintcorporation.com/

[16]-- Sitio de Escáneres On-line, "Online Security Check" [Consultado: Diciembre 2007] Disponible en: http://www.alken.nl/online-security-check.htm

[17]-- Sitio de Nmap [Consultado: Noviembre 2007] Disponible en: http://insecure.org/nmap

[18]--Deraison, Renaud. “Nessus Network Auditing” Versión Digital. Editorial: Syngress Publishing, Inc. 2004. Disponible en: http://www.flazx.com/ebook4188.ph

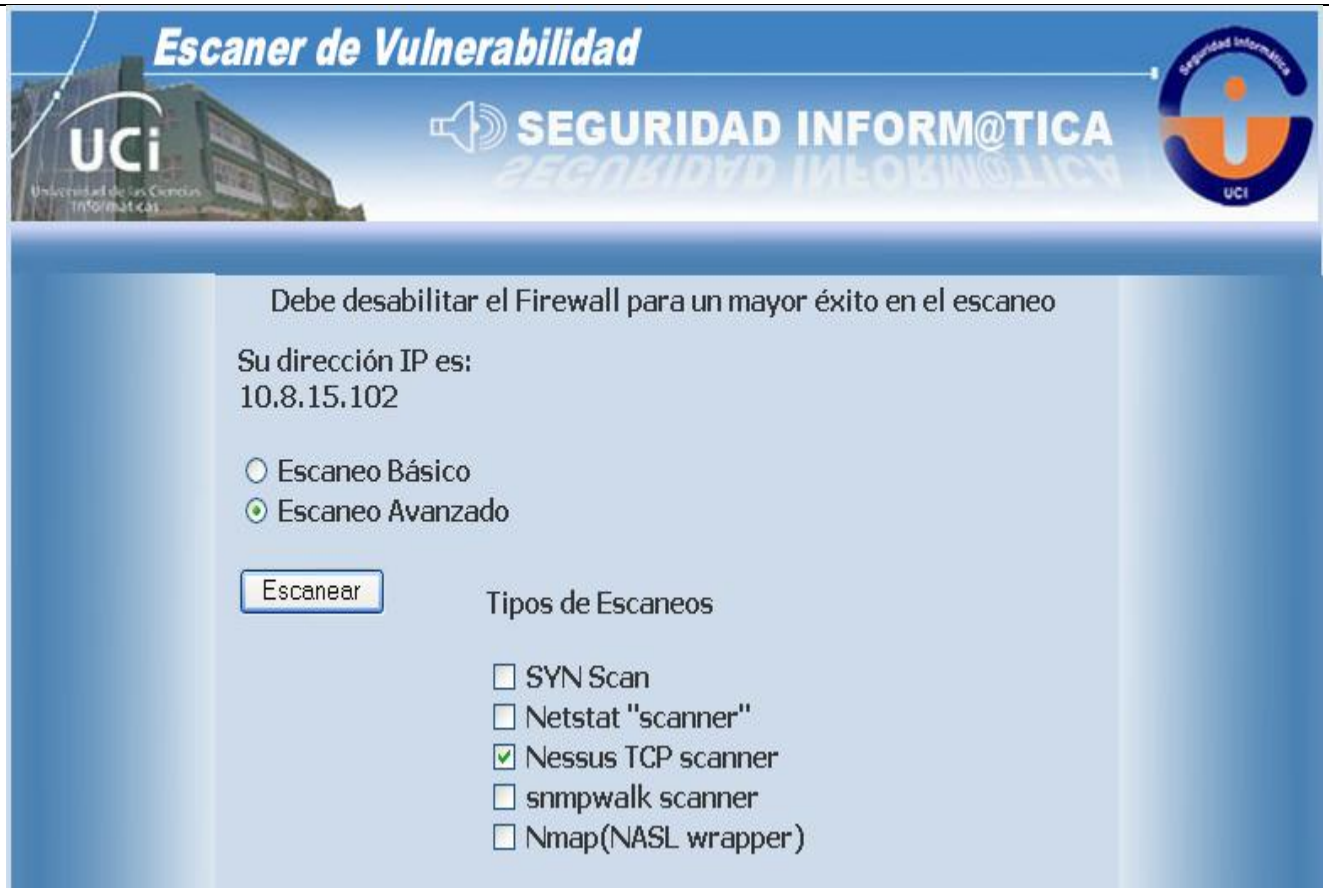
[19]-- Introducción a AJAX, autor: Javier Eguíluz Pérez [consultado: 2 de junio de 2008]

ANEXOS

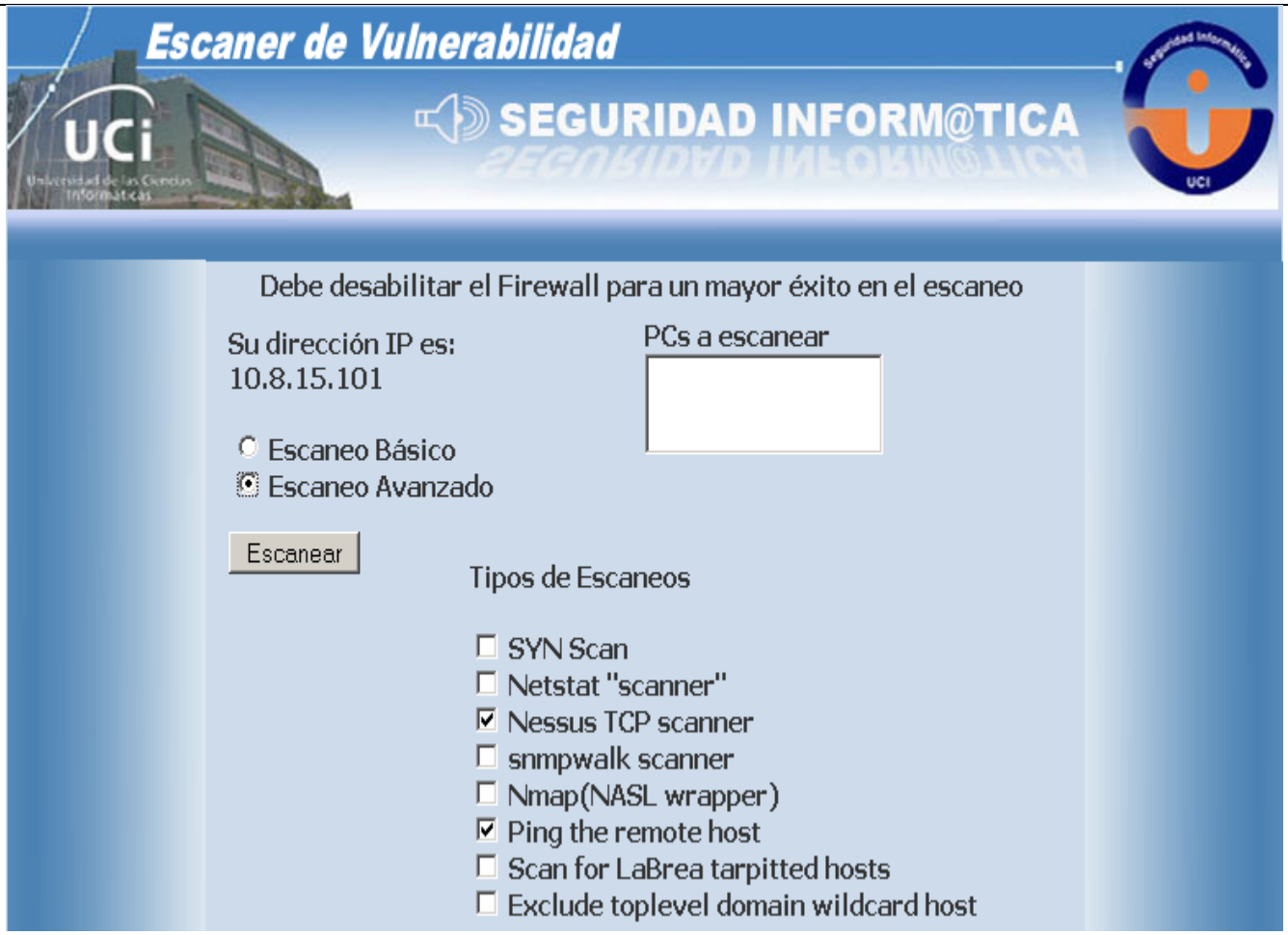
Anexo 1 Definición de los casos de uso.

Caso de uso	
CU-1	Configurar Escaneo de Vulnerabilidades
Propósito	Que el usuario o el administrador solicite y escoja las pruebas de vulnerabilidades que desea hacer sobre su PC.
Actores	Usuario
Resumen: El usuario o el administrador solicita al sistema que realice sobre su PC las pruebas de vulnerabilidad que posee, el sistema obtuvo el IP de la PC solicitante, le muestra los tipos de escaneo existentes, el usuario selecciona los deseados y el sistema guarda la configuración realizada.	
Referencias	R- 8,9,10,11

Sección: "Configurar Escaneo por el Usuario"	
Acción del Actor	**Acción del Sistema**
1-Solicita el escaneo de vulnerabilidades.	
2-Selecciona la opción de escaneo básico.	
	3-El sistema adquiere la configuración seleccionada.
	.
Flujo alternativo	
Acción del Actor	**Acción del Sistema**
2-Selecciona la opción de escaneo avanzado.	
	3-El sistema ofrece las opciones de configuración avanzada.
4-El usuario configura el escáner de acuerdo a las opciones emitidas por el sistema.	
	5-Se almacena la configuración seleccionada por el usuario en el archivo de configuración .
Prototipo de interfaz	

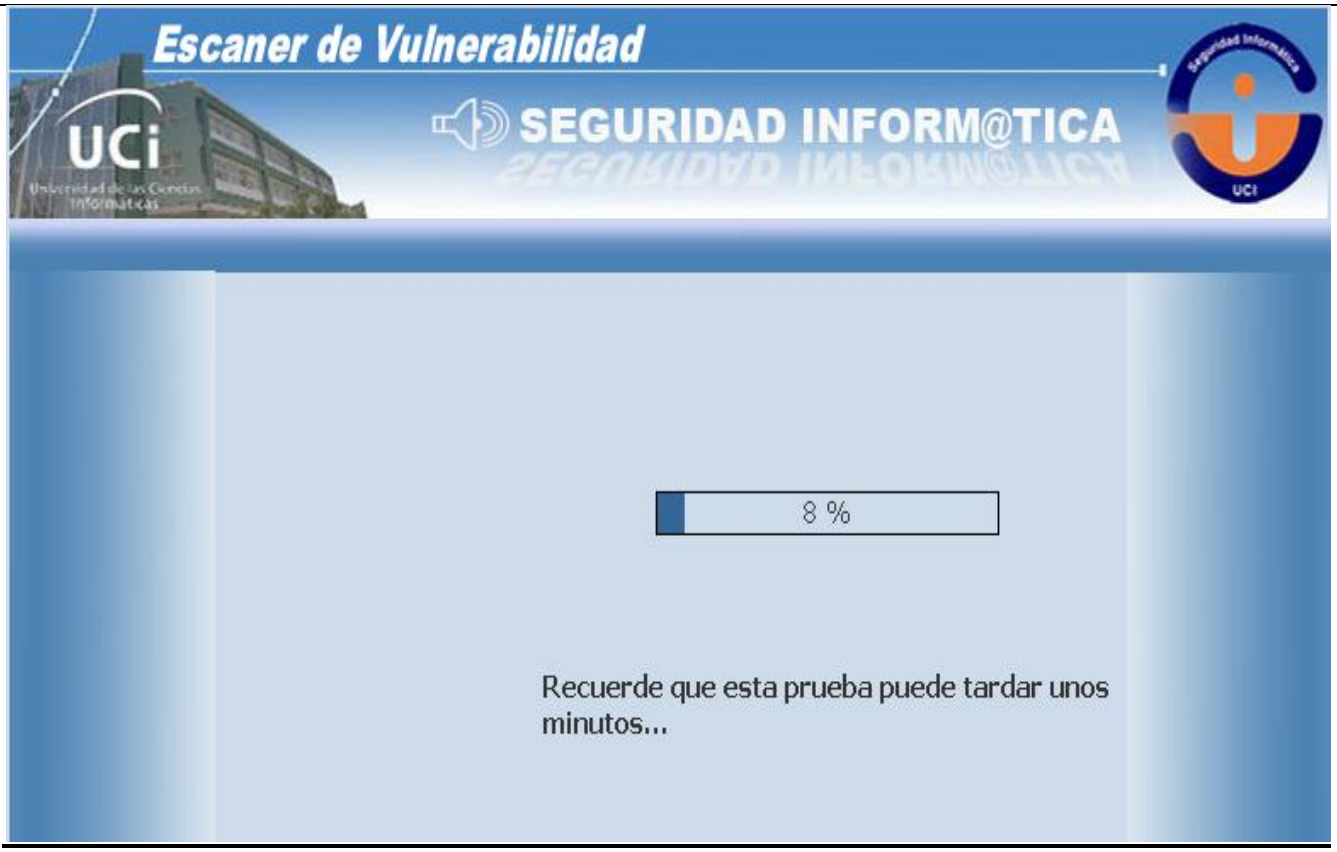


Sección: "Configurar Escaneo por el Administrador"	
Acción del Actor	**Acción del Sistema**
1-El administrador solicita la opción de configuración de escaneo.	
	2-El sistema ofrece las opciones de configuración
3-Selecciona las opciones para el escáner	
	4- Almacena la configuración.
Prototipo de interfaz	



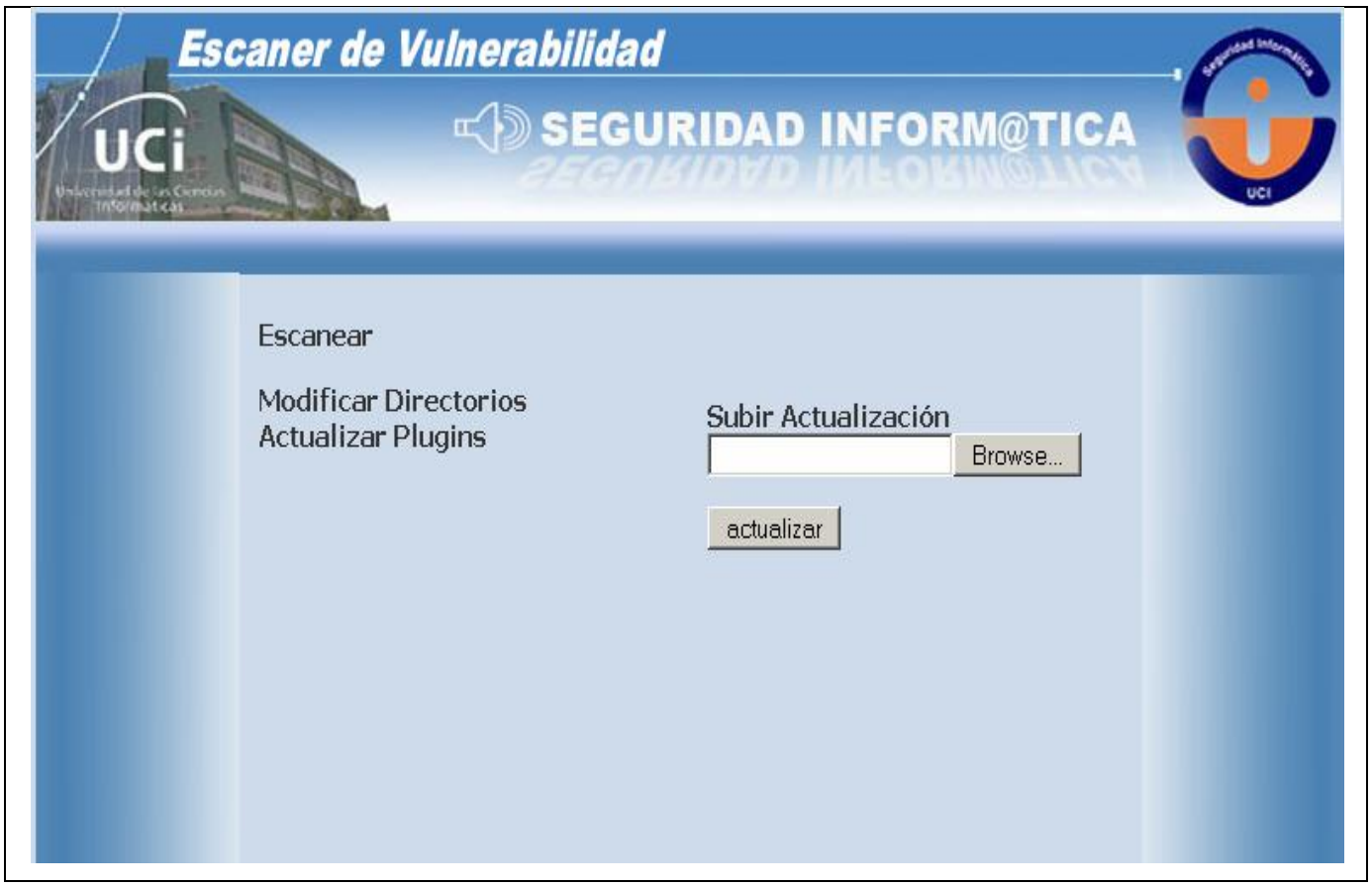
Caso de uso		
CU-2	Realizar Escaneo de Vulnerabilidades	
Propósito	Realizar pruebas de vulnerabilidades y obtener reporte.	
Actores	Usuario	
Resumen: El sistema escanea los puertos a la PC solicitante, crea la KB, con ayuda de esta y la configuración de las opciones que hizo el usuario, selecciona los plugins a utilizar y los ejecuta sobre la PC solicitante obteniendo información de sus vulnerabilidades.		
Referencias	R-12,13,14,16,17	
Acción del Actor		**Acción del Sistema**
1-El usuario o el administrador oprime el botón		2-El sistema ejecuta el escáner.

escanear.	
	3-El sistema le muestra el reporte al usuario emitido por el escáner.
	4- El reporte es almacenado en la base de datos.
Prototipo de interfaz	



Caso de uso	
CU-3	Gestionar Actualización.
Propósito	Actualizar las pruebas de vulnerabilidades y escáner.
Actores	Administrador
Resumen: El administrador solicita al sistema actualizar los plugins, el sistema comprueba que posee permisos de administración; luego el administrador realiza la acción de actualizar sobre estos elementos.	
Referencias	R- 18,19,20,21

Acción del Actor	**Acción del Sistema**
1- Solicita actualización de los plugins	
	2- Le pide el fichero con las actualizaciones.
3- El administrador introduce la dirección del fichero.	
	4- Comprueba el fichero
	5- Adquiere las actualizaciones.
Flujo alternativo	
Acción del Actor	**Acción del Sistema**
3- El administrador introduce la dirección del fichero.	
	4-Comprueba que el fichero no cumple con los requisitos.
	5- El sistema emite una alerta de error y vuelve a solicitar el fichero.
Prototipo de interfaz	



Caso de uso	
CU-4	Autenticar
Propósito	Comprobar que el usuario tenga los permisos necesarios para la solicitud seleccionada del sistema.
Actores	usuario
Resumen: El usuario para realizar la solicitud de escaneo debe antes autenticarse para que el sistema compruebe que este posee los permisos correspondientes, el sistema le pide autenticación y el usuario entra su usuario y contraseña pertenecientes al dominio UCI, el sistema lo comprueba y le	

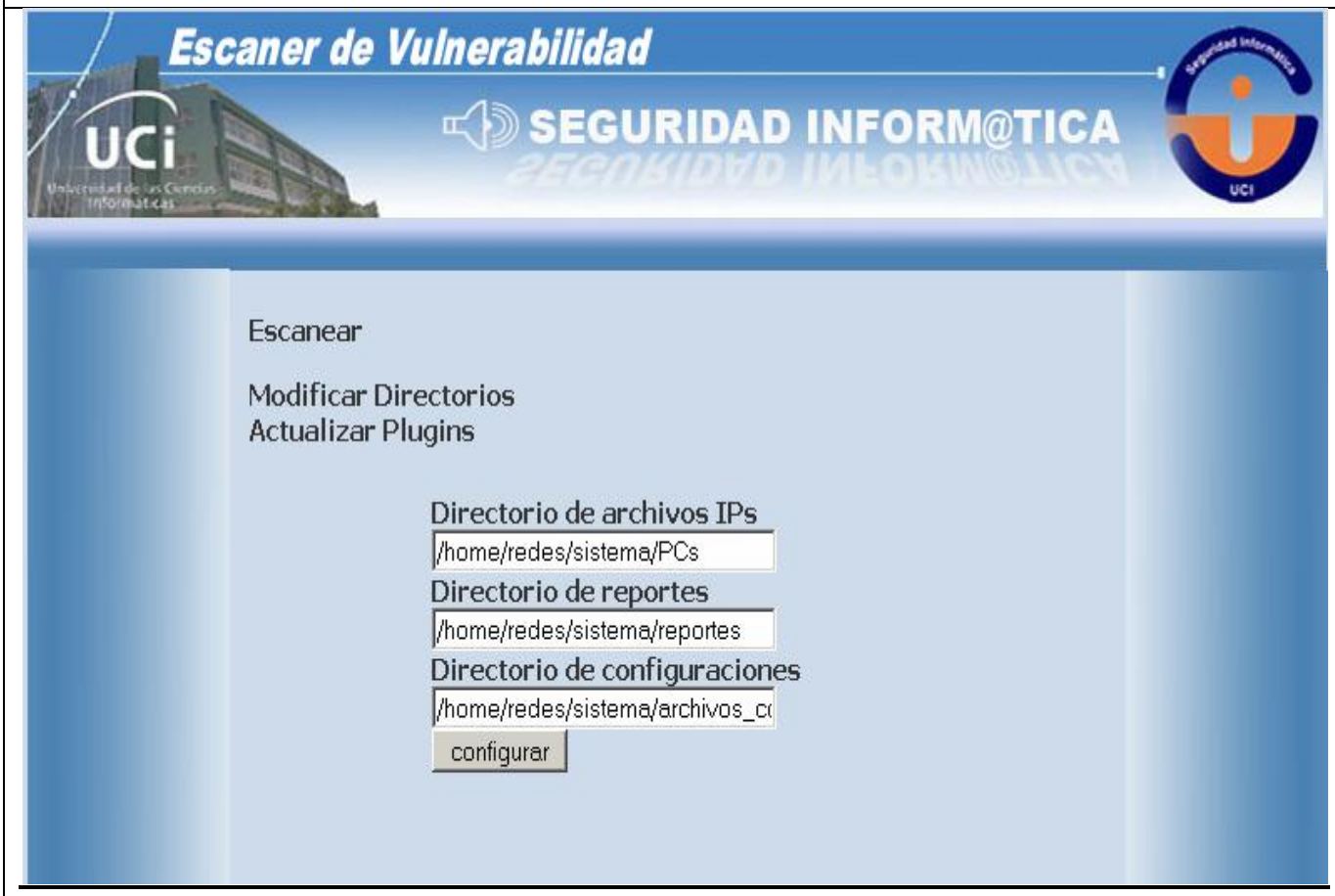
proporciona los servicios correspondientes.	
Referencias	R- 1,2,2.1,2.2, 3,3.1,3.2,4,5,6,7
Acción del Actor	**Acción del Sistema**
1- Este caso de uso se inicia cuando el usuario accede al sistema e introduce usuario y contraseña.	
	2.-Comprueba que en realidad es su usuario y contraseña
	3-Obtiene y almacena el ip y el usuario en la base de datos, además almacena el ip en el archivo de las PCs.
	4- Le da acceso a las opciones de escaneo.
	5-Se almacena el tiempo (hora) en la base de datos.
Flujo alternativo	
Acción del Actor	**Acción del Sistema**
1- El usuario introduce el usuario y contraseña en el sistema.	2- El sistema comprueba que el usuario, la contraseña o ambos son incorrectos .
	3- El sistema emite un mensaje de error y vuelve a solicitarle los datos al usuario.
Prototipo de interfaz	

Caso de uso	
CU-4	Modificar Directorios
Propósito	Modificar los directorios de configuración, de reporte, el de las ip
Actores	Administrador
Resumen: Permitir que el administrador cambie la localización de los directorios donde se almacenan tanto los ip a escanear, como los reportes, como las configuraciones que lleva a cabo el usuario.	
Referencias	R- 21, 21.1 , 21.2 , 21.3 ,21.4 ,21.5 , 21.6, 21.7

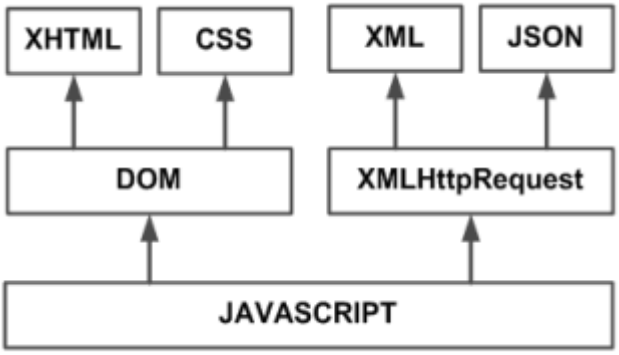
Acción del Actor	**Acción del Sistema**
1- El Administrador después de autenticarse accede a la página donde se le brinda las opciones de modificar directorios, actualizar plugins y escanear.	
2-El administrador selecciona las opciones que	

desea modificar directorios.	
	3-El sistema le muestra los directorios que tiene configurado permitiendo que el administrador cambie los directorios.
4-El administrador cambia el directorio o varios directorios .	
	5-El sistema almacena la configuración hecha por el Administrador para posteriores escaneos.

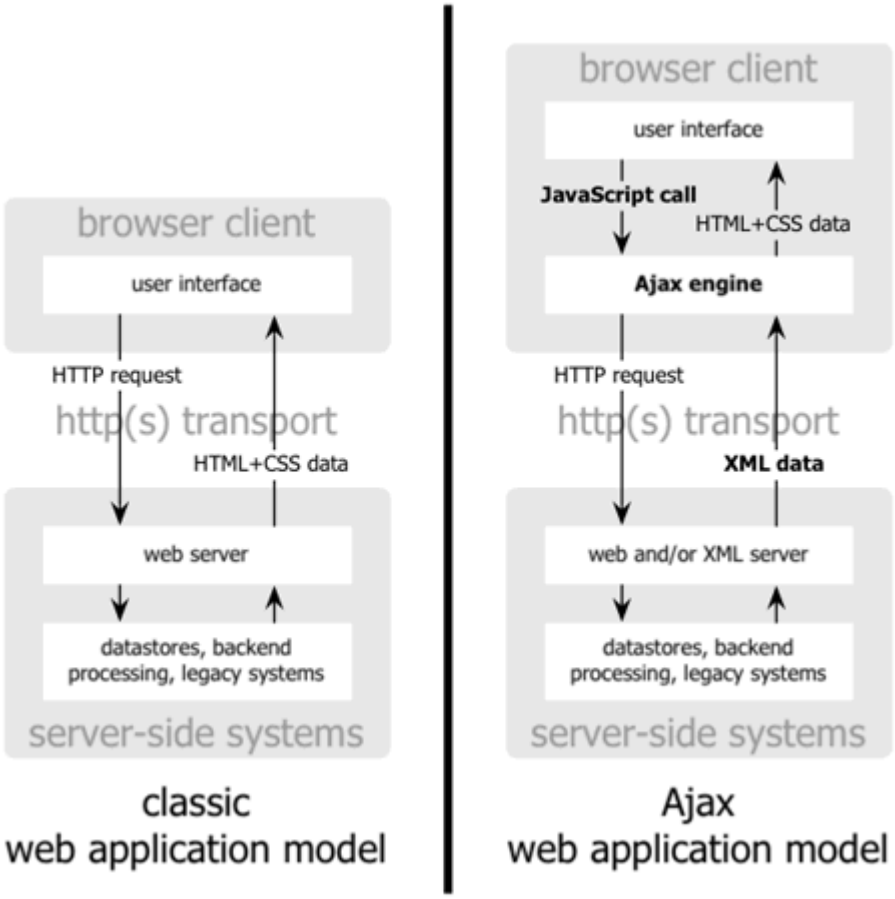
Prototipo de interfaz



ANEXO 2 Tecnologías agrupadas bajo el concepto de AJAX



ANEXO 3 Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX.



Anexo 4: Diagramas de secuencia. Clases del Diseño.

Diagrama de secuencia Autenticar

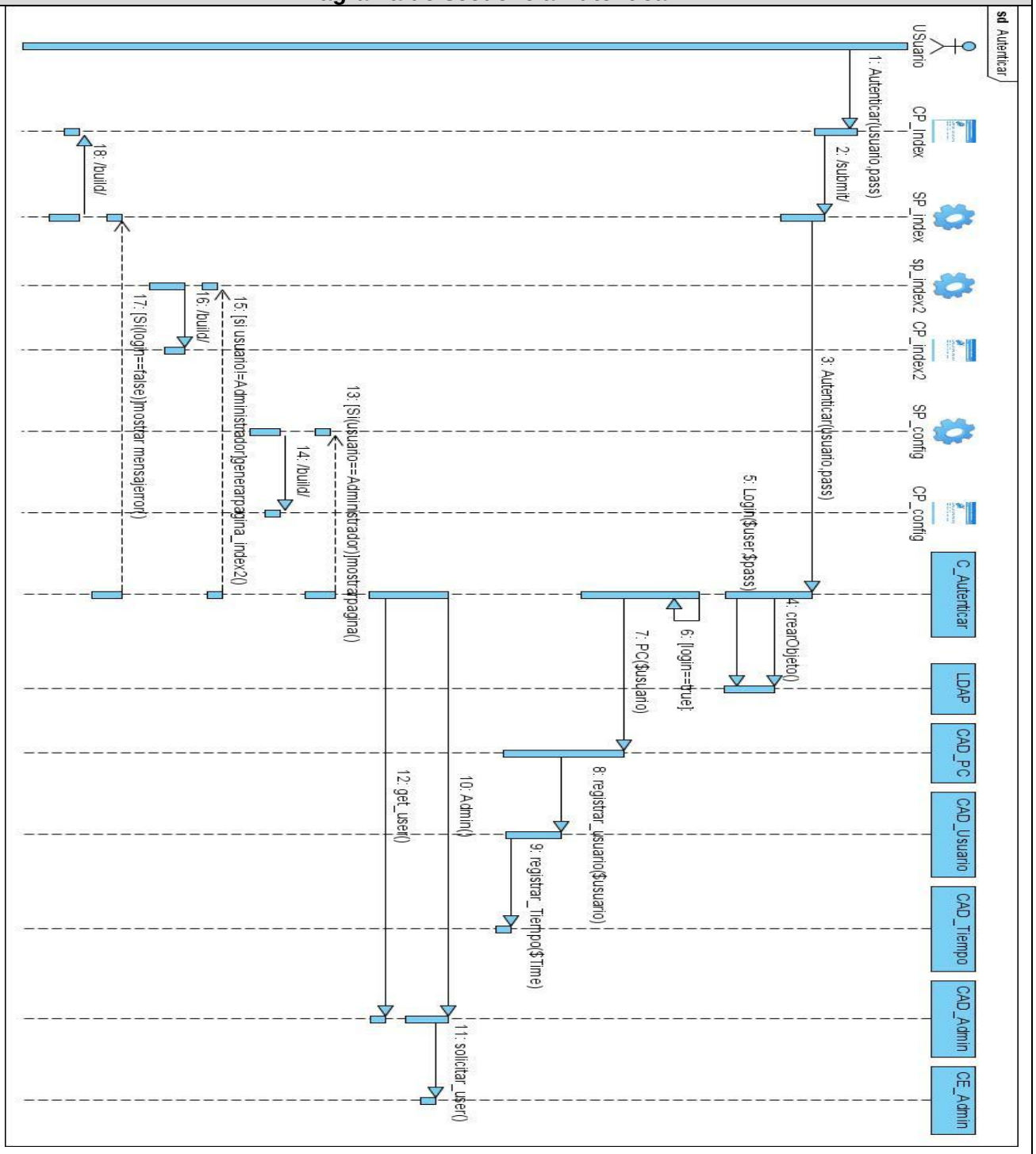


Diagrama de secuencia Configurar Escaneo por el usuario

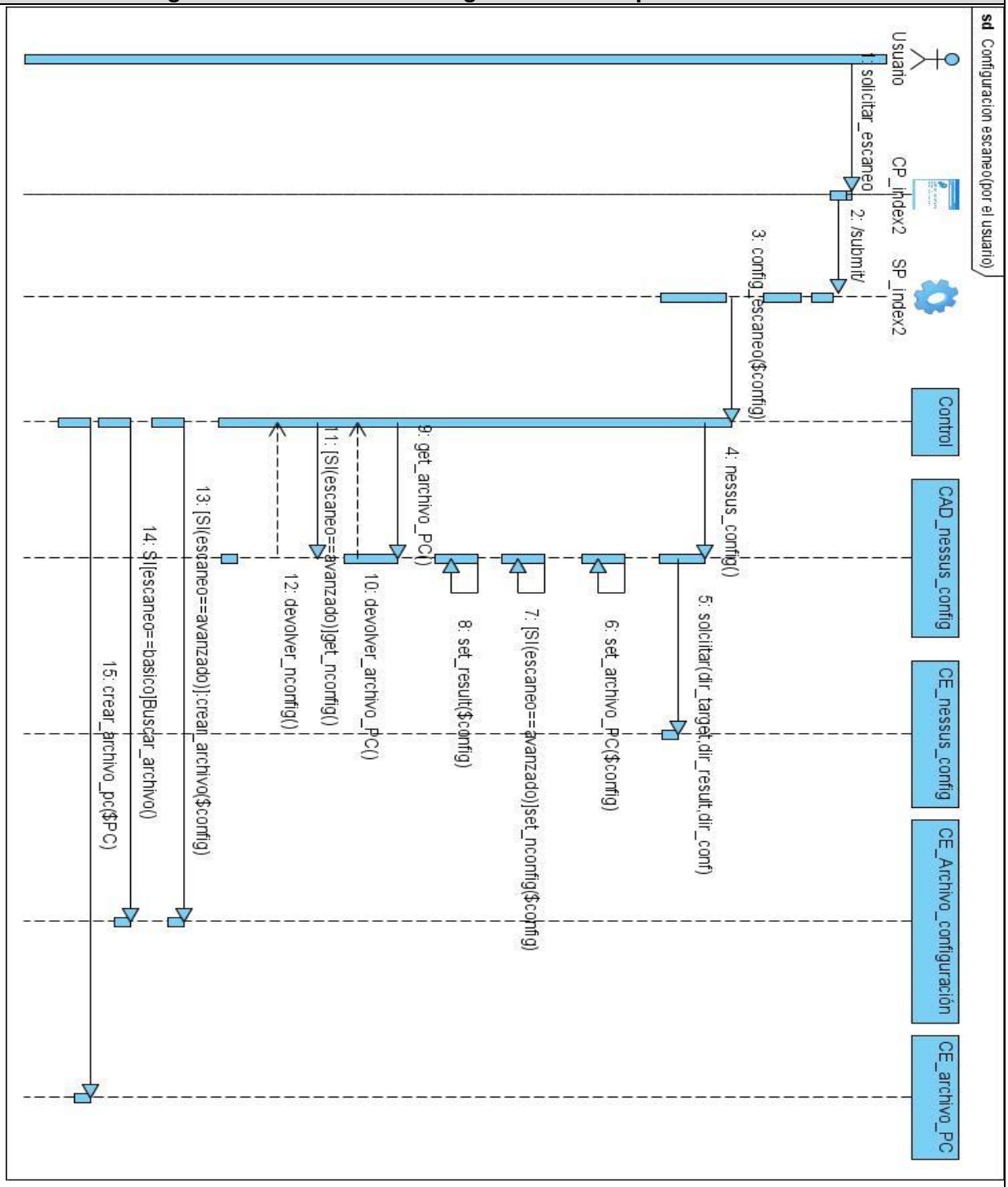


Diagrama de secuencia Configurar Escaneo por el Administrador

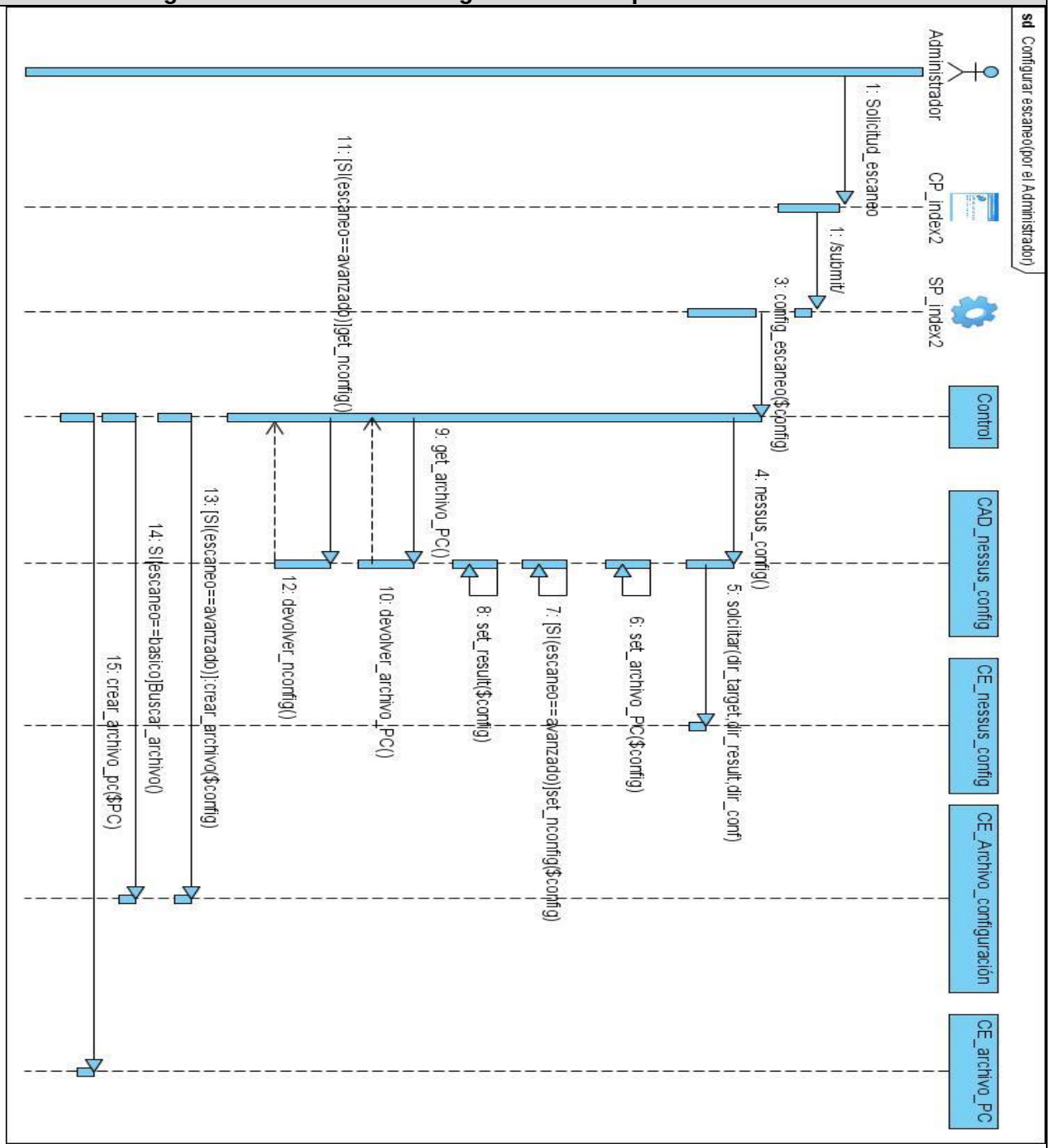


Diagrama de secuencia Gestionar Actualizaciones

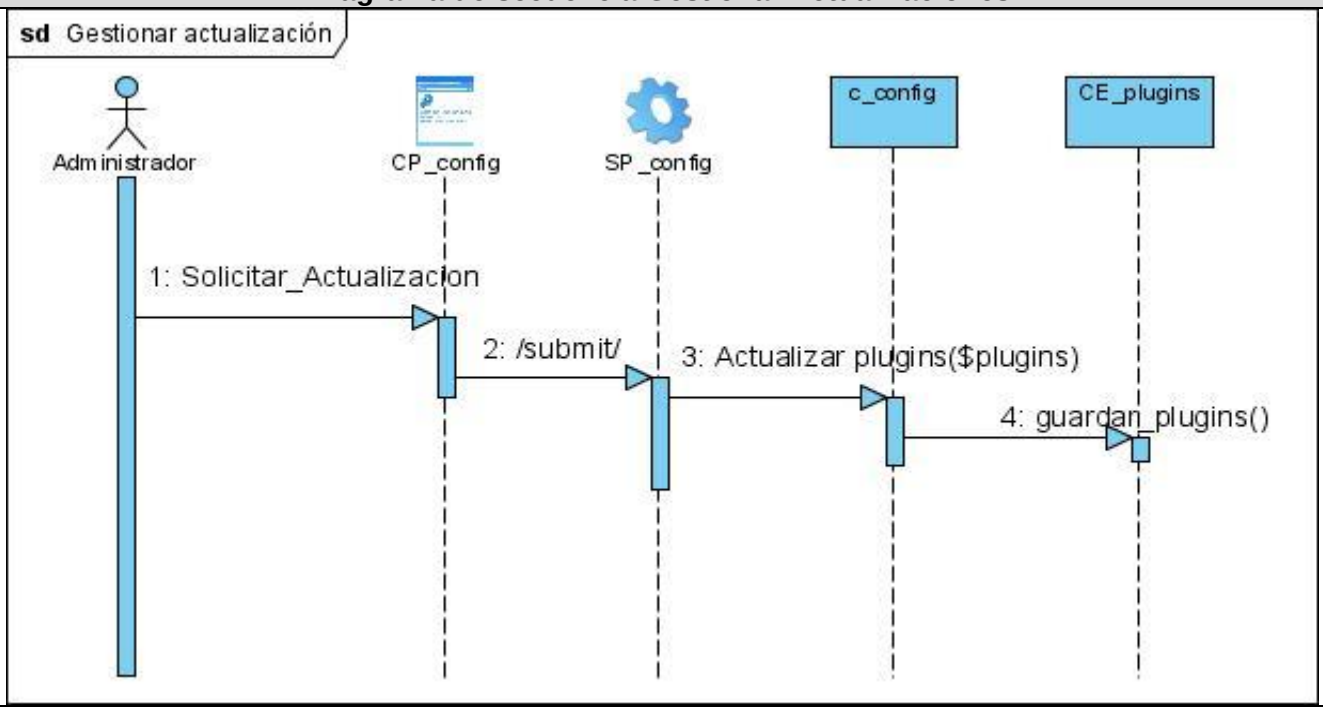


Diagrama de secuencia Modificar Directorios

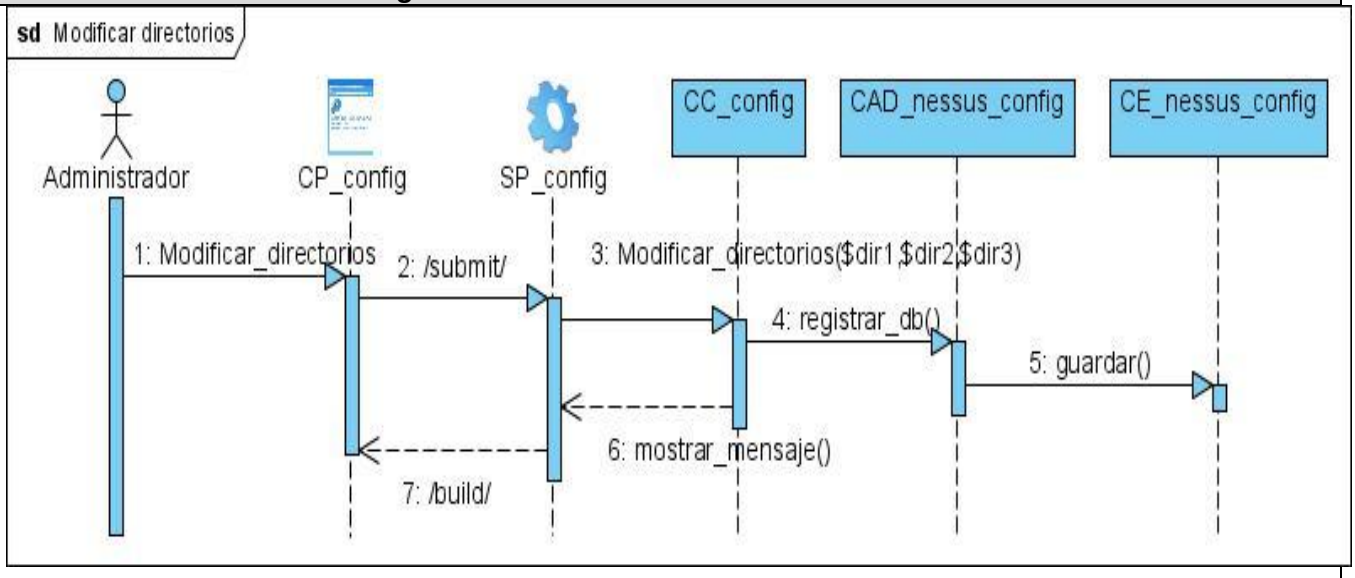
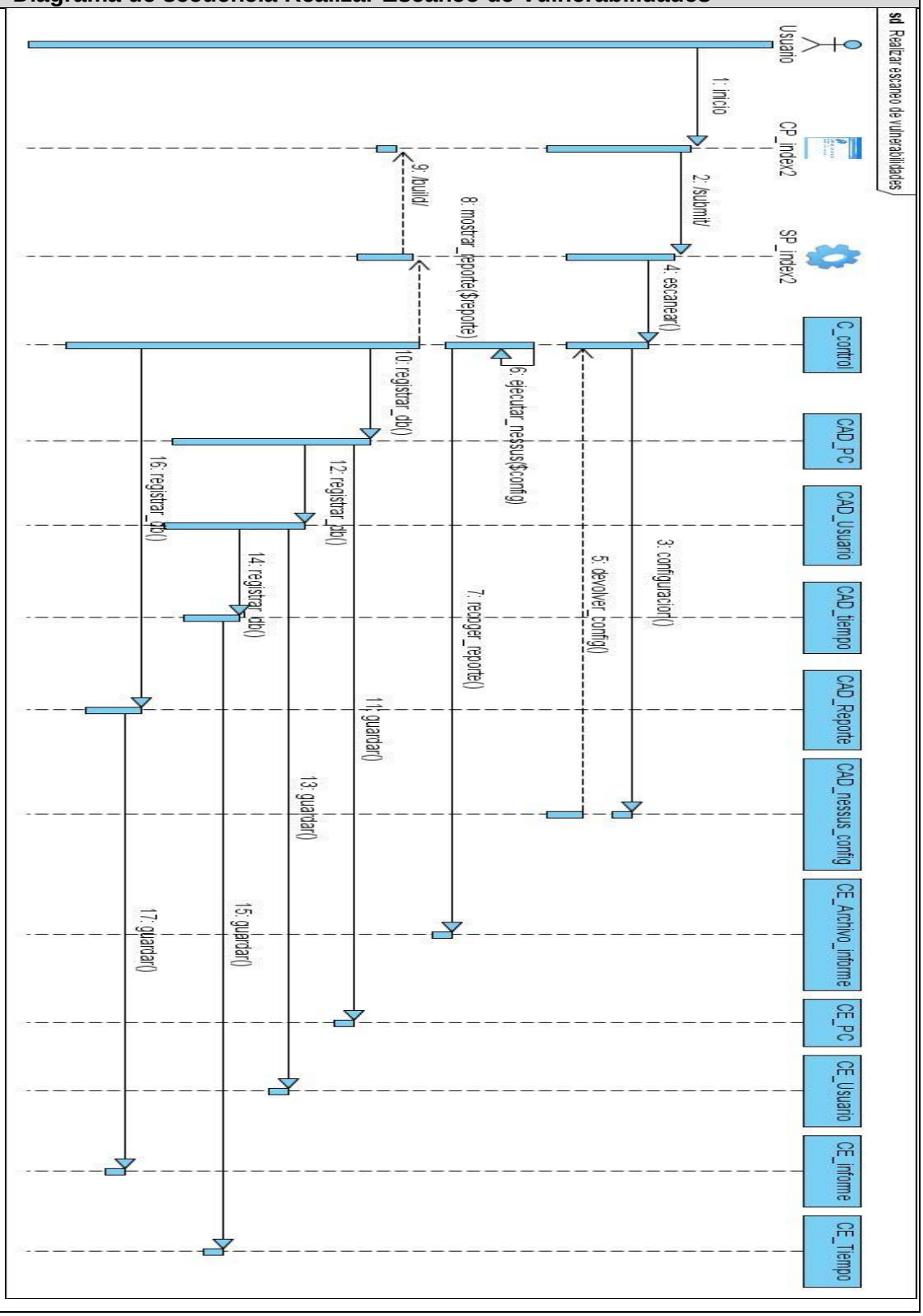


Diagrama de secuencia Realizar Escaneo de Vulnerabilidades



GLOSARIO DE TÉRMINOS Y SIGLAS

1. Administrador: es la persona que tiene privilegios para determinadas funcionalidades del sistema.
2. APACHE: es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1.
3. AJAX: Asynchronous JavaScript And XML.
4. Arquitectura Cliente/Servidor: es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos.
5. CASE: *Computer Aided Software Engineering.*
6. Daemon: Es un tipo especial de proceso informático que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario (es un proceso no interactivo)
7. HTML: *HyperText Markup Language.* Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986. Es un lenguaje de marcas. Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que se pueden controlar desde cualquier editor ASCII.
8. HTTP: *HyperText Transfer Protocol.* Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas Web.
9. Herramientas CASE: Herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.
10. Hardware: Componentes electrónicos, tarjetas, periféricos y equipo que conforman un sistema de computación; se distinguen de los programas (software) porque son tangibles.
11. Internet: Sistema de redes de computación ligadas entre sí, con alcance mundial, que facilita servicios de comunicación de datos como registro remoto, transferencia de archivos, correo electrónico y grupos de noticias. Internet es una forma de conectar las redes de computación existentes que amplía en gran medida el alcance de cada sistema participante.
12. Linux: Es el nombre de un núcleo, pero se suele denominar con este nombre a un sistema operativo de libre distribución software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona, con los conocimientos informáticos adecuados, puede libremente estudiarlo, usarlo, modificarlo y redistribuirlo.

13. LaTeX: Es un lenguaje de marcado para documentos, y un sistema de preparación de documentos, formado por un gran conjunto de macros de tex, con la intención de facilitar el uso del lenguaje de composición tipográfica. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados con LaTeX es comparable a la de una editorial científica de primera línea.
14. MVC: *Modelo Vista Controlador.*
15. NASL (Nessus Attack Scripting Language), Lenguaje de Scripting de Ataque Nessus. Lenguaje para crear los plugins de nessus.
16. Nessusd: El daemon de Nessus, que realiza el escaneo en el sistema objetivo.
17. PC : *Personal Computer.*
18. PHP: *PHP: Hypertext Preprocessor*. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas. Con PHP se pueden combinar páginas HTML y script*s*. Con el objetivo de crear aplicaciones potentes.
19. PostgreSQL: es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre.
20. RUP: *Rational Unified Process* (Proceso Unificado de desarrollo). Metodología para el desarrollo de Software.
21. Software: Programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.
22. Sitio Web: Es un conjunto de páginas web, típicamente comunes a un dominio de Internet o subdominio en la World Wide Web en Internet.
23. SGBD: *Sistema de Gestión de Bases de Datos*. Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.
24. Tex: Es un sistema de composición de textos de alta calidad creado por Donald E. Knuth, dirigido en particular a aquéllos textos que contienen una gran cantidad de expresiones matemáticas.
25. UCI: *Universidad de las Ciencias Informáticas*.
26. UML: *Unified Modeling Language*. Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.
27. XML: *Extensible Markup Language*. Es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium. Orientado principalmente al almacenamiento, procesamiento y transmisión de mensajes.