FACULTAD 2

TÍTULO: SOLUCIÓN INFORMÁTICA PARA EL DIAGNÓSTICO DE LA SEGURIDAD DESDE LA PLATAFORMA OSSIM, TENIENDO EN CUENTA LA INFORMACIÓN QUE GESTIONA LA PLATAFORMA PLATSI

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Adrián Roldós del Arco
Willian Galvez Fernández

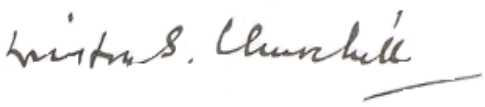
Tutores:

Msc. Yasser Azán Basallo
Ing. Luís E. Gallardo Concepción

La Habana, junio de 2016

"Año 58 de la Revolución"

"El éxito no es definitivo, el fracaso no es fatídico: lo que cuenta es el valor para continuar". - Winston Churchill



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, se firma la presente a los ____ días del mes de _____ del año _____.

_____	_____
Adrián Roldós del Arco	Willian Galvez Fernández
Autor	Autor
_____	_____
Msc. Yasser Azán Basallo	Ing. Luís E. Gallardo Concepción
Tutor	Tutor

Tutor: MSc. Yasser Azán Basallo (yazan@uci.cu): graduado de Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Pertenece al Centro de Telemática (TLM). Es profesor del Departamento de Práctica Profesional en la Facultad 2.

Tutor: Ing. Luís Eduardo Gallardo Concepción (legallardo@uci.cu): graduado de Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Pertenece al Centro de Telemática (TLM).

Consultante: MSc. Annia Arencibia Morales (aarencibia@uci.cu): graduada de Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas en el año 2007. Pertenece al Centro de Informática Médica (CESIM), posee categoría docente de profesor Asistente. Se desempeña como Metodóloga del Centro de Informática Médica. Es máster en Informática Aplicada.

Consultante: Ing. Darien Castellano Pérez (dcperez@uci.cu): graduado de Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas en el año 2009. Es profesor del Departamento de Programación en la Facultad 2, posee categoría docente de profesor Instructor y está desarrollando la maestría en Informática Aplicada.

Dedicatoria

Adrián Roldós del Arco

Dedico esta tesis a toda mi familia por tanto esfuerzo y sacrificio que me han dedicado durante todos estos años, pero en especial a mis dos amores: mis hijos.

Willian Galvez Fernández

Dedico esta tesis, a mis familiares y en especial a mis padres, a mi hermano y a mis abuelas por la educación que me han brindado.

Resumen

Con el presente trabajo de investigación: Solución informática que permite el diagnóstico de la seguridad desde la plataforma OSSIM, teniendo en cuenta la información que gestiona la plataforma PLATSI, se detalla el negocio de gestión de los reportes generados por la plataforma PLATSI hacia la plataforma OSSIM, en ETECSA. El tema de la investigación está enfocado en la integración de información en los sistemas de telecomunicaciones.

En la investigación, se propone el desarrollo de una solución *middleware*, para el diagnóstico de la seguridad desde la plataforma OSSIM, teniendo en cuenta la información que gestiona la plataforma PLATSI, a través del Nivel de Integración Estructural de la Empresa de Telecomunicaciones de Cuba S.A.

De esta manera, se alcanzó a desarrollar un *plugin* que permitió la correcta comunicación entre las plataformas PLATSI y OSSIM en ETECSA, permitiendo la escalabilidad y flexibilidad en el sistema, aportando un servicio de comunicación en herramientas remotas y permitiendo centralizar los reportes de las vulnerabilidades detectadas. Para la realización de este trabajo se utilizó la metodología de desarrollo OpenUP, Framework-OSSIM como Framework de desarrollo, PostgreSQL y MySQL como Sistemas Gestores de Bases de Datos.

Palabras clave: integración, *middleware*, OSSIM, PLATSI, reporte, vulnerabilidade

Índice

Índice de tablas

Índice de figuras

Introducción

El desarrollo alcanzado por las nuevas tecnologías de la informática y las comunicaciones, ha llevado a la sociedad a entrar en lo que se ha denominado la “era de la información”. La información puede existir de muchas formas: impresa o escrita en papel, almacenada digitalmente, transmitida por correo postal o utilizando medios digitales, presentada en imágenes o expuesta en una conversación. Cualquiera sea la forma que adquiera, es un recurso que, como el resto de los activos importantes, tiene un gran valor.

Dentro del mundo empresarial, la digitalización de la información es un paso fundamental que permite la optimización de procesos, elevando el nivel de competitividad y las posibilidades de desarrollo. Para su gestión, existen múltiples herramientas que varían según su plataforma de desarrollo, lenguaje, arquitecturas y otros factores, por lo que las empresas no cuentan con un único sistema centralizado. La heterogeneidad de tecnologías crea las llamadas “islas de información”, donde la comunicación entre estos sistemas informáticos es insuficiente, provocando la fragmentación de los procesos de negocio y que la documentación estratégica de la entidad, esté dividida.

Para lograr la centralización, son de vital importancia la transmisión y la protección de la información digital, debido al creciente auge de los delitos de carácter informático, lo cual conlleva a que la seguridad informática sea un tema crítico para las grandes empresas y compañías de la sociedad moderna mundial. Con el objetivo de garantizar la seguridad de la información, existen diversas herramientas, tanto privativas como de código abierto, que posibilitan realizar auditorías de seguridad en aplicaciones web, así como otras que realizan un monitoreo de la red, con el propósito de detectar vulnerabilidades que puedan poner en peligro la integridad de los datos dentro de la entidad.

A Cuba, como nación en el mundo de las tecnologías y herramientas, le es imprescindible mantenerse involucrada en las transformaciones que se van asumiendo en el sector y propone adaptaciones desde soportes libres, atendiendo sistemáticamente la seguridad informática de sus equipos, plataformas e infraestructuras. La Empresa de Telecomunicaciones de Cuba S.A. (ETECSA), cuyo principal objetivo es “Brindar servicios de telecomunicaciones que satisfagan las necesidades de los clientes y la población, así como respaldar los requerimientos del desarrollo socio-económico del país, con los resultados que de la empresa se demandan”

(ETECSA 2016), siendo la única de su tipo en Cuba, puede ser considerada el punto más estratégico para recibir ataques informáticos.

Debido a los altos volúmenes de información que maneja esta empresa, es crucial garantizar la seguridad de la misma. Para lograrlo, el auditor en Seguridad Informática dispone de dos herramientas libres: la Administración de la Seguridad de la Información de Código Abierto (OSSIM, por sus siglas en inglés), es una distribución de productos de código abierto integrados, ideada para construir una infraestructura de monitorización de seguridad, desarrollada por AlientVault; y la Plataforma de Seguridad en las Tecnologías de la Información (PLATSI), que gestiona las auditorías en aplicaciones web, la cual fue desarrollada por la Universidad de las Ciencias Informáticas (UCI), en el Centro de Telemática (TLM).

Dichas plataformas funcionan de manera independiente, brindando reportes o informaciones de diferentes problemas de la seguridad. Cada una de ellas tiene un usuario tipo auditor, encargado de generar auditorías a aplicaciones web; el de PLATSI, al ejecutar una de las herramientas de la plataforma, obtiene un conjunto de vulnerabilidades y el auditor define cuáles no son falsos positivos, a través de su conocimiento empírico. Una vez terminada esta operación, crea un reporte y lo envía al auditor de OSSIM, el cual realiza las mismas operaciones en su plataforma, además es el encargado de realizar una correlación de los datos suministrados por el auditor de PLATSI, en formato PDF, con los datos suministrados por OSSIM (OSSIM también exportan en otros formatos como xls, doc y otros), tarea que se dificulta por las razones siguientes:

- A pesar de que ambas plataformas exportan los reportes en formato PDF, no cuentan con la misma estructura datos, trayendo consigo que la misma vulnerabilidad puede estar representada con dos estructuras diferentes. El auditor debe revisar cada una de estas vulnerabilidades de forma manual, haciendo una comparación de cada valor de la vulnerabilidad e identificando semejanzas y diferencias. El reporte generado por ambas plataformas puede tener desde una hasta más de doscientas vulnerabilidades, lo que trae consigo demoras por partes del auditor a la hora de realizar el análisis, incumpliendo con las fechas de entrega a las instancias superiores.

- El auditor de OSSIM, debe conocer cómo se representa la información de las vulnerabilidades en cada una de las plataformas (aproximadamente existe un total de

200 vulnerabilidades que pueden ser detectadas entre ambas plataformas), esta tarea conlleva a un elevado esfuerzo físico, por parte de la persona encargada, la misma debe conocer cada una de las existentes.

- Se generan reportes manuales en formato PDF, que no están exentos de equivocaciones humanas, afectando el proceso de toma de decisiones en instancias superiores. Por esta razón, se debe contar con una persona de experiencia para la realización de la labor, tratando de esta forma de minimizar la cantidad de posibles errores que pueda tener la información, pues esto implicaría que las posibles vulnerabilidades puedan llegar a ser amenazas o incurrir en daños a la organización en cuestión.

Por lo anteriormente expuesto, se identifica el **problema a resolver:** se dificulta realizar, desde la plataforma OSSIM, el diagnóstico de la seguridad, teniendo en cuenta la información que gestiona la plataforma PLATSI, de la Empresa de Telecomunicaciones de Cuba S.A.

Una vez identificado el problema, se hace necesario enfocar la investigación en la integración de información en los sistemas de telecomunicaciones, lo cual constituye el **objeto de estudio** de la misma; definiéndose así como **campo de acción:** integración de la información a través del Nivel de Integración Estructural.

Para dar solución al problema planteado surge el **objetivo general:** desarrollar una solución informática para el diagnóstico de la seguridad desde la plataforma OSSIM, teniendo en cuenta la información que gestiona la plataforma PLATSI de la Empresa de Telecomunicaciones de Cuba S.A.

Para dar cumplimiento al objetivo general, se definen las siguientes **tareas de la investigación:**

1. Analizar soluciones existentes en el área de las telecomunicaciones, en el ámbito nacional e internacional, en la actualidad, relacionadas a la integración de los sistemas informáticos, estableciendo similitudes con la investigación en curso.
2. Asimilar herramientas, tecnologías y metodología, utilizadas en el desarrollo de las plataformas PLTASI y OSSIM para la propuesta de solución.
3. Analizar las plataformas PLATSI y OSSIM, identificando las vulnerabilidades, para ser incluidas en la propuesta de solución.

4. Elaborar el diseño de la propuesta de solución haciendo uso de arquitecturas de integración.
5. Desarrollar la propuesta de solución, que permita el diagnóstico de la seguridad desde la plataforma OSSIM, teniendo en cuenta la información gestionada por la plataforma PLATSI.
6. Desarrollar las pruebas necesarias para el correcto funcionamiento de la solución propuesta.

Al concluir la presente investigación, se espera obtener los siguientes beneficios:

- La persona encargada de realizar la correlación de las vulnerabilidades detectadas por ambas plataformas, al término de esta investigación, empleará menos esfuerzo físico en desarrollar su trabajo, pues la información se encontrará centralizada en la plataforma OSSIM.
- Las entregas a instancias superiores, por parte de los auditores, se realizarán en un menor tiempo, permitiendo corregir las vulnerabilidades antes de que se conviertan en amenazas, debido que la plataforma OSSIM automatizará el trabajo que antes se realizaba de forma manual.
- Al realizar la correlación desde la plataforma OSSIM, se disminuirán los errores humanos, por lo que se podrá realizar un análisis más completo del estado de la seguridad de la institución, permitiendo disminuir gastos económicos por las posibles amenazas que puedan ocurrir.

Para el desarrollo de este trabajo se emplearon los siguientes **métodos de investigación científicos**:

Métodos Teóricos

El **método analítico-sintético:** la revisión bibliográfica se realizó sobre un conjunto de libros, publicaciones, monografías y documentos en soporte electrónico, que se encuentran situados en páginas web, internet, trabajos de curso, revistas, entre otros y se utilizó para sintetizar todas las citas, apuntes y datos tomados al respecto.

El **método histórico-lógico:** se utilizó para realizar un análisis del estado del estado del arte de las principales aplicaciones orientadas a la integración, relacionadas con el campo de acción.

El **método modelación:** se utilizó en la confección de los modelos y diagramas que ayudaron a la comprensión de los procesos desarrollados.

Métodos Empíricos

El **método de observación:** sirvió de apoyo para obtener y recopilar información sobre el tema a tratar. Se utilizó para ver cómo funciona actualmente el diagnóstico de la seguridad informática, de los sistemas auditados en ETECSA.

El **método entrevista**, favoreció el proceso de recoger las opiniones necesarias que poseen distintos especialistas y conocedores de la rama tratada, sobre el tema de la investigación.

El documento está estructurado en tres capítulos:

Capítulo 1. Fundamentación teórica de la integración entre las plataformas OSSIM y PLATSI: En este capítulo se estudian los conceptos fundamentales asociados al dominio del problema, para un mejor entendimiento de la investigación. Incluye un estado del arte del tema tratado, así como la descripción de las metodologías, tecnologías y herramientas empleadas en el desarrollo de la solución.

Capítulo 2. Propuesta de la solución informática: En este capítulo se realiza un análisis de las plataformas PLATSI y OSSIM, identificando las vulnerabilidades que deben ser incluidas en la propuesta de solución. Se describen las características de la solución informática a desarrollar. Además, se define el modelo de dominio, los requisitos funcionales y no funcionales, para comprender mejor su funcionamiento, así como, los actores, diagrama de casos de uso del sistema y la descripción de los casos de usos del sistema.

Capítulo 3. Análisis y Diseño, Implementación y Pruebas de la solución informática: Este capítulo está relacionado con el diseño del sistema; se describen los patrones de diseño empleados, se presentan los diagramas de clases del diseño, de despliegue, de secuencia, de componentes; se describen los estándares, codificación y el tratamiento de errores, para la solución informática propuesta.

Capítulo 1: Fundamentación teórica de la integración entre las plataformas OSSIM y PLATSI

En este capítulo se abordan diferentes conceptos asociados al problema, que son útiles para un mejor entendimiento de la información y se caracteriza el objeto de estudio. Se realiza un estudio del estado del arte de algunas herramientas y tecnologías usadas actualmente, se describen algunas de las soluciones existentes en el mundo, enmarcándose en la situación problemática y se hace la selección de las herramientas a utilizar para el desarrollo de la solución.

1.1 Conceptos asociados al dominio del problema

Durante la investigación se identificaron diferentes conceptos, necesarios para el desarrollo de la solución propuesta, a continuación estos se detallan.

OSSIM

La Administración de Seguridad de Información de Código Abierto (OSSIM, por sus siglas en inglés), es una plataforma de seguridad y de código abierto, que ha estado disponible desde 2003. OSSIM incorpora en su solución de Administración de Eventos y Seguridad de la Información (SIEM, por sus siglas en inglés), la administración consolidada, presentación de informes consolidados, y de arrendamiento múltiple de Proveedores de servicios gestionados. OSSIM es totalmente funcional, integra una gran variedad de herramientas de código abierto, que junto con el potente sistema de correlación de eventos, la convierten en una indispensable herramienta para la administración de seguridad, en redes realmente complejas. (AlienVault 2016)

PLATSI

La Plataforma de las Tecnologías para la Seguridad de la Información (PLATSI), gestiona las auditorías en aplicaciones web; estas son realizadas por los especialistas de Seguridad Informática de ETECSA. La auditoría consiste en ejecutar desde una aplicación web, las herramientas que realizarán las pruebas de seguridad a la aplicación web de la entidad

especificada. Una vez realizada la solicitud de la prueba, el especialista selecciona de las herramientas que están integradas en el sistema, la que desea ejecutar y luego esta solicitud es recibida por el servidor, quien la asigna a un agente para su procesamiento. Cuando se realiza la prueba, se obtienen los resultados para analizarlos y procesarlos, para confeccionar un informe general con los resultados finales. La aplicación engloba una serie de funcionalidades que le dan respuesta a las necesidades existentes y se obtiene un reporte con las vulnerabilidades detectadas. (TLM 2014)

El sistema está compuesto por tres (TLM 2014):

- **Administración:** garantiza el control de acceso a la plataforma, la gestión de usuarios y los permisos para el acceso de estos, además de la configuración de herramientas y tareas de administración.
- **Auditorías:** brinda la posibilidad de ejecutar de forma remota las herramientas Acunetix y Nikto, para las pruebas de seguridad, además da a conocer en tiempo real, el estado de las pruebas (ejecución, detenida, cancelada).
- **Reporte:** permite generar un informe técnico y de alta gerencia, con los resultados obtenidos en las pruebas de seguridad realizadas. Esta muestra los tipos de vulnerabilidades que afectan al sistema y el impacto que pueden ocasionar, además evalúa el nivel de seguridad del sistema, según los resultados generados.

Middleware o lógica de intercambio de información entre aplicaciones

Es un *software* que asiste a una aplicación, para interactuar o comunicarse con otras aplicaciones o paquetes de programas, redes, *hardware*, además de sistemas operativos. Este simplifica el trabajo de los programadores, en la compleja tarea de generar las conexiones y sincronizaciones, que son necesarias en los sistemas distribuidos. (ORACLE 2014) Características (ORACLE 2014):

- Capa entre sistema operativo y las aplicaciones distribuidas.
- Oculta la complejidad y heterogeneidad de sistema distribuido.
- Puentes brecha entre las comunicaciones del sistema operativo de bajo nivel y las abstracciones del lenguaje de programación.

- Proporciona la abstracción de programación común y la infraestructura para aplicaciones distribuidas.

1.2 Soluciones existentes en el área de las telecomunicaciones para la integración entre sistemas informáticos

Al analizar el negocio actual que se realiza en ETECSA, para la gestión de la seguridad, con las plataformas PLATSI y OSSIM, se obtuvo un conjunto de características, con las que debían cumplir las soluciones existentes que permiten la integración de plataformas, identificando si las soluciones encontradas pudieran servir como propuesta de solución. Las características son listadas a continuación:

- La información debe ser centralizada en uno de los sistemas que se desee integrar y no en un sistema independiente concebido solo para este fin.
- En el proceso de integración se debe realizar transformación de los datos.
- Deben ser soluciones de *software* libre.

eiConsole IDE

EiConsole IDE es una plataforma de integración de sistemas orientados a la industria. Tiene una intuitiva interfaz gráfica de usuario, con componentes totalmente configurables, desarrollado por la empresa PilotFish. Este motor de integración ayuda a establecer una exitosa integración mediante las técnicas de Extracción, Transformación y Carga (ETL, por sus siglas en inglés), Integración de Información Empresarial (EII, por sus siglas en inglés) e Integración de Aplicaciones Empresariales (EIA, por sus siglas en inglés). Está implementado con el simple nivel de integración punto a punto (PTP, por sus siglas en inglés), además puede hacer uso de tecnologías como la de Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés). Esta puede integrar sistemas sin tener en cuenta la plataforma, sistema operativo, lenguaje de base de datos o protocolos de comunicación. (PILOTFISH 2016)

IBM WebSphere MQ

WebSphere MQ es un producto *middleware* líder de mercado, para integración de mensajería. Este fue introducido en el mercado en 1993 con el nombre IBM MQSeries por la empresa IBM. Ofrece un mecanismo de transporte de alto rendimiento y disponibilidad, confiable, escalable y

seguro, para atender los requerimientos de conectividad de las empresas. Proporciona la red troncal de mensajería universal para la conectividad utilizando SOA. Conecta prácticamente todos los sistemas de la tecnología de la información comercial, brindando soporte a más de 80 plataformas. (IBM 2016)

EiPlataform

EiPlatform es un moderno motor de integración de sistemas orientados a la salud. Este, como su hermano eiConsole IDE, está desarrollado por la empresa PilotFish. Es flexible, extensible y compatible con cualquier base de datos, sistema operativo, plataforma y aplicación de servidor. Su nativo XML de soporte, asegura la compatibilidad con cualquier estándar de integración de salud futuro. EiPlataform se acopla al ambiente de integración de eiConsole IDE, el cual le proporciona la interfaz gráfica para realizar las configuraciones pertinentes. (PILOTFISH 2016)

Microsoft Integration Services

Integration Services es una plataforma para la creación de soluciones empresariales de transformaciones e integración de datos, desarrollado por Microsoft. Es flexible porque trata datos de diversos orígenes como archivos XML o texto plano, para después cargarlos en uno o varios destinos. La herramienta gráfica está enfocada para crear soluciones, sin escribir una sola línea de código. (MICROSOFT 2016)

InterSystems Ensemble

Es una plataforma flexible, que utiliza infraestructuras basadas en el Bus de Servicios Empresariales (ESB, por sus siglas en inglés) y SOA para la conectividad rápida y el desarrollo de nuevas aplicaciones conectables. Permite crear aplicaciones compuestas, que protegen y amplían las inversiones de *software* anteriores. Utiliza proveedores de aplicaciones. Posee una interfaz gráfica de usuario intuitiva, permitiendo a los desarrolladores crear rápidamente nuevas aplicaciones. Es generado basándose en caché, lo que permite gestionar a miles de usuarios simultáneos y terabytes de datos. (CORPORATION 2016)

Luego de realizar un análisis de las soluciones encontradas, para lograr la integración, se concluye que ninguna satisface las necesidades del cliente; estas aplicaciones se encuentran enfocadas a un negocio o área específica. Ninguna ha sido utilizada anteriormente para lograr

la integración de las plataformas OSSIM Y PLATSI. En todos los casos se determinó que las licencias son privativas, lo que contradice las políticas de soberanía digital de nuestro país. En el caso de la plataforma eiConsole, cumple con la mayoría de las características definidas, pero a través de esta no se realiza la integración de plataformas de seguridad informática, sino que está enfocada al sector de la industria. En el ámbito nacional no se encontró referencia de aplicaciones que permitan la integración de soluciones informáticas, en el área de las telecomunicaciones.

1.3 Herramientas, tecnologías y metodologías utilizadas en el desarrollo de la propuesta de solución

En la actualidad, el uso de tecnologías de última generación, es un requisito casi indispensable para garantizar la calidad de un sistema. Se impone la utilización de tecnologías que brinden rapidez, eficacia y que ofrezcan a los desarrolladores herramientas poderosas que permitan realizar un conjunto de procedimientos, ya sean básicos o altamente avanzados.

SSHTunnel 0.0.6

SSHTunnel consiste en un túnel encriptado, a partir del protocolo de conexión denominado Intérprete de Ordenes Seguro (SSH por sus siglas en inglés). Esta herramienta tiene gran utilidad cuando se habla de la integridad de los datos, en términos de seguridad de la información. Esta garantiza que los datos viajen por la red sin ser espiados, cambiados o neutralizados, asegurando así que se garantice el cumplimiento de los pilares básicos de la seguridad informática: integridad, confidencialidad y disponibilidad. Este no es independiente, por lo que necesita del módulo Paramiko. (PHAZ BLINOV 2016)

Se utiliza SSHTunel para cifrar la información que viaja entre los servidores PLATSI y OSSIM. De esta manera se asegura los datos de las vulnerabilidades lleguen a su destino sin interrupciones ni alteraciones.

Paramiko 1.16.0

Paramiko es un módulo para Python, de la versión 2.2 en adelante, del protocolo SSHv2; el cual permite cifrar y autenticar conexiones remotas de forma segura. Este ha sido desarrollado completamente por Python y utiliza su lenguaje de programación. Este no necesita de

certificados jerárquicos otorgados por un agente central para establecer una conexión, diferenciándose del protocolo Capa de Puertos Seguros (SSL por sus siglas en inglés) o Seguridad de la Capa de Transporte (TLS por sus siglas en inglés), lo que le permite tener mayor autonomía cuando se habla de establecer una conexión segura. Además, permite abrir canales arbitrarios a los servicios remotos a través de túneles encriptados. (Jeff Forcier 2016)

PyYalm 3.11

PyYalm es un interpretador YAML ("YAML Ain't Markup Language") escrito específicamente para el lenguaje de programación Python. Este interpreta las especificaciones de los algoritmos YAML, lo que lo convierte en un lenguaje muy sencillo, que permite describir los datos como en XML, pero con una sintaxis mucho más sencilla. (Oren Ben-Kiki 2012)

Plataforma de desarrollo Framework OSSIM

Un framework o marco de trabajo en el desarrollo de *software*, es una estructura de soporte definida, mediante la cual otro proyecto de *software* es organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros *softwares*, para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de *software* que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio. Este es una base de programación que atiende a sus descendientes (manejado de una forma estructural y/o en cascada) posibilitando cualquier respuesta ante las necesidades de sus miembros, o secciones de una aplicación web. (Milanés López, Aldo y Montano García, Reinier 2009)

El uso del framework OSSIM en esta investigación está sujeto a la plataforma OSSIM.

Lenguaje de programación Python 2.6.6

Este lenguaje de programación cuenta con una sintaxis muy limpia y que contribuye a que el código sea legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, que es multiplataforma y orientado a objetos.

Python permite dividir el programa en módulos reutilizables desde otros programas; se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje o probar funciones durante el desarrollo del programa. (Python Software Foundation 2015)

El uso del lenguaje de programación Python en esta investigación, está sujeto principalmente a la plataforma OSSIM, debido a que los *plugin* son implementados con dicho lenguaje.

Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD, por sus siglas en inglés), es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Está compuesto por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, además de garantizar la seguridad e integridad de los mismos. (Date, C.J 1998)

MySQL

Es un sistema multiplataforma de manejo, creación y administración de base de datos (Database Management System, DBMS) *opensource*, para base de datos relacionales. Cuenta con un sistema multihilo, que ofrece un soporte eficiente y veloz, permitiendo acceder a todos los campos que resguardan los datos de trabajo. (MySQL 2015)

Características de MySQL (MySQL 2015):

- Cuenta con la capacidad de realizar tareas multiprocesador, debido a que posee la opción de trabajo multihilo.
- Puede ingresar una enorme cantidad de datos por columna de trabajo.
- Cuenta con varias API, disponibles para los principales lenguajes de programación que existen.
- Aplicación con una portabilidad sobresaliente.
- Capacidad de soportar hasta 32 índices de tablas diferentes.

- Un nivel de seguridad que permite gestionar varios usuarios, mediante contraseñas individuales.

PostgreSQL

Es un SGBD relacional, orientado a objetos y libre, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. (Scribid 2014)

Dentro de sus principales ventajas (Scribid 2014) se encuentran:

- Soporta distintos tipos de datos.
- Posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que atienden un gran número de solicitudes.
- Puede ser instalado un número ilimitado de veces sin temor de sobrepasar la licencia.
- Es extensible a través del código fuente, disponible sin costos adicionales.
- Soporte nativo para los lenguajes más populares del medio: PHP, C++, Perl, Python, entre otros.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos especiales, minería de datos, entre otros.
- Es multiplataforma, disponible en Linux, Unix, MacOsX y Windows, entre otros.

El uso de los gestores de base de datos MySQL y PostgreSQL en esta investigación está sujeto a la plataforma OSSIM y a la plataforma PLATSI, pues son los gestores utilizados por dichas herramientas respectivamente.

Herramienta para el modelado de software

Las herramientas de Ingeniería de Software Asistidas por Computadora (CASE por sus siglas en inglés) brindan soporte para desarrollar y mantener *software*. Son herramientas individuales que ayudan al desarrollador de *software* o administrador de proyecto, durante una o más fases del desarrollo de *software*. (EUI-FI 2015)

Para el desarrollo del presente trabajo se utiliza la herramienta CASE Visual Paradigm 5.0, debido a que es la definida por el centro TLM, para el modelado de las aplicaciones.

Visual Paradigm 5.0

Es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este *software* de Lenguaje de Modelado Unificado (UML por sus siglas en inglés), ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML. (SparxSystems 2014)

Características (SparxSystems 2014):

- Soporte de UML versión 2.1
- Diagramas de Procesos de Negocio
- Ingeniería inversa
- Modelo a código, diagrama a código
- Diagramas de flujo de datos
- Soporte ORM
- Generación de bases de datos
- Transformación de diagramas de Entidad-Relación en tablas de base de datos
- Distribución automática de diagramas

Nivel de integración

Los niveles de integración permiten establecer Proyectos de Integración de Sistemas, al definir cómo está la empresa y hasta dónde quiere llegar, lo que permitirá identificar la estrategia a utilizar. (Microsoft Patterns & Practices 2013)

Nivel de Integración punto a punto

Representa el nivel más pobre de integración, pues las tecnologías utilizadas se comunican a través de interfaces intermedias donde no se contempla la visión del negocio; es decir, establece una infraestructura básica para el intercambio de datos entre aplicaciones. La

interrelación entre los sistemas es baja, por lo que tienen un alto grado de independencia. (Microsoft Patterns & Practices 2013)

Nivel de Integración estructural

En este nivel de integración, se utiliza la tecnología Middleware para estandarizar y controlar el intercambio de información, así como para consolidar las reglas de negocio y transacciones entre aplicaciones. Debe tomarse en cuenta que en este punto existe una estructura central, que controla el intercambio de información y las organizaciones cuentan con interfaces integradas y fuentes de datos comunes, pero no se integra con componentes externos del negocio. (Microsoft Patterns & Practices 2013)

Nivel de Integración de procesos

A este nivel, la organización ha logrado la transición de compartir la información entre aplicaciones para manejar en realidad el flujo de datos entre las aplicaciones. Para conseguirlo, han desarrollado un modelo de negocio común que cubre la totalidad de la empresa. Utiliza tecnología sofisticada para poner en práctica el modelo de negocio en la capa Middleware. Existe gran integración en este nivel, debido a que se toman en cuenta las reglas de negocio y el flujo de procesos. (Microsoft Patterns & Practices 2013)

Nivel de Integración externa

Las compañías en este nivel, están considerando las verdaderas aplicaciones empresariales, la influencia tecnológica, la transformación de los procesos de negocio y las nuevas estructuras, para redefinir la organización desde el punto de vista de servir a los clientes. Estas organizaciones usan tecnologías de Integración de Aplicaciones Empresariales (EAI, por sus siglas en inglés) para transformar el negocio, a menudo uniendo directamente a clientes y proveedores para operaciones internas. Estas compañías contemplan nuevas capacidades para crear innovadoras ofertas en línea, nuevos productos y servicios, y pueden mejorar una marca registrada o crear una nueva identidad en Internet. (Microsoft Patterns & Practices 2013)

Se escoge el nivel de integración estructural, producto de que las plataformas PLATSI y OSSIM se comunican a través de tecnología Middleware, para estandarizar y controlar el intercambio de información. En este punto existe una estructura central que controla el intercambio de datos

entre ambas plataformas; y a nivel de Middleware se consolidan las reglas de negocio y las transacciones entre las aplicaciones. Los dos sistemas cuentan con interfaces integradas y fuentes de datos en común, pero no se integran con componentes externos del negocio.

Técnicas de integración

El esquema tradicional cliente/servidor, permite que diferentes módulos de aplicación, se comuniquen directamente sin una capa intermedia. El problema se presenta en sistemas complejos con componentes de diversos proveedores, donde resulta poco flexible e inoperante la comunicación.

Replicación de Datos

La Replicación de Bases de Datos es una técnica de integración, que se basa en la creación y el mantenimiento, de múltiples copias de una misma base de datos. En la mayoría de las implementaciones de Replicación, un servidor mantiene la copia primaria de la base de datos y servidores adicionales mantienen las copias esclavas de la misma. (Microsoft Patterns & Practices 2013)

Extracción, Transformación y Carga de Datos

La técnica Extracción Transformación y Carga de Datos (ETL, por sus siglas en inglés), como su nombre lo indica, extrae información de un sistema fuente, transforma esos datos para satisfacer los requerimientos del negocio y carga el resultado en el sistema destino. Tanto la fuente como el destino son generalmente bases de datos y archivos. (Microsoft Patterns & Practices 2013)

Integración de Información Empresarial

La Integración de Información Empresarial (EII, por sus siglas en inglés) es otra de las técnicas de integración, la cual es un mecanismo de transformación y acceso a datos, transparente y optimizado para suministrar una única interfaz, a lo largo de los datos de las organizaciones. Dicha interfaz permite acceder a los datos y el resultado de este método es un Sistema de Información Heterogéneo Distribuido, virtualmente integrado. Este tipo de solución consiste en crear un intermediario, que contenga los directorios de la base de datos y que a su vez sirva de canal de consulta y representación de la información recuperada. Los datos son capturados en

tiempo real, lo que implica que las fuentes de datos tengan una estructura tecnológica sólida y bien establecida. EII protege a las aplicaciones de la complejidad de recuperar datos de múltiples localizaciones, donde los datos pueden diferir en semántica y formato, y de emplear diferentes interfaces de datos. (Microsoft Patterns & Practices 2013)

Integración de Aplicaciones Empresariales

Integración de Aplicaciones Empresariales (EAI, por sus siglas en inglés), es el proceso de integrar múltiples aplicaciones desarrolladas independientemente, que utilizan tecnología incompatible y que son gestionadas de forma independiente, permitiendo que se comuniquen e intercambien transacciones de negocio, mensajes, y datos entre sí. Uno de los principales objetivos de EAI, es proporcionar acceso transparente a la amplia gama de aplicaciones que existen en una organización. Las características más importantes de esta tecnología son que se utiliza para la integración de Aplicaciones a Aplicaciones y que proporciona un enfoque de integración orientado a proceso, basado en mensajes XML. (Microsoft Patterns & Practices 2013)

De las técnicas de integración antes mencionadas se decide a utilizar la extracción, transformación y carga de datos. Debido a que las plataformas PLATSI y OSSIM disponen de gestores de bases de datos diferentes para almacenar la información, es necesario realizar la extracción de los reportes de las vulnerabilidades encontradas por las herramientas de PLATSI. Una vez extraídos los datos, se les debe aplicar una limpieza y transformación para que puedan ser manejados desde OSSIM.

Metodologías de desarrollo de software

Las metodologías de desarrollo de *software* surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental, a la hora de desarrollar un producto de *software*. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo *software*, pero los requisitos de un *software* a otro son tan variados y cambiantes, que han dado lugar a que exista una gran variedad de metodologías para la creación del *software*. (Carrillo P., Isaías; Pérez G., Rodrigo; Rodríguez M., Aureliano D. 2015)

Las metodologías para el desarrollo de *software*, se encuentran divididas en dos grupos fundamentales:

Metodologías robustas: dentro de estas se encuentra Rational Unified Process (RUP). Las metodologías robustas son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo, llamadas también metodologías tradicionales o clásicas, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema. (P.Letelier 2015)

Metodologías ágiles: dentro de estas se encuentran Extreme Programing (XP), Open Unified Process (OpenUP). Las metodologías ágiles son aquellas que se utilizan cuando el desarrollo del *software* es incremental (y se deben hacer entregas pequeñas de *software*, con ciclos rápidos), cooperativo (cliente y desarrollador trabajan juntos constantemente con una comunicación cercana), sencillo (el método es sí mismo es fácil de aprender y modificar, bien documentado), y adaptable (permite realizar cambios de último momento). (P.Letelier 2015)

OpenUP

Es un proceso de desarrollo unificado que está basado en RUP. Este mantiene las mismas características de RUP, pues está dirigido por casos de uso, centrado en la arquitectura y además es iterativo e incremental (EPF 2011). El ciclo de vida de un proyecto según esta metodología, se divide en cuatro fases fundamentales:

1. **Concepción**: en esta fase se pretende determinar los objetivos y establecer el alcance del proyecto.
2. **Elaboración**: el propósito de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para el desarrollo de la siguiente fase.
3. **Construcción**: esta fase tiene como propósito completar el desarrollo del sistema, basándose en la arquitectura definida, enfocándose en el diseño, implementación y pruebas de las funcionalidades, para desarrollar un sistema completo.
4. **Transición**: en esta fase se asegura que el *software* esté listo para entregar a los usuarios.

OpenUP propone seis flujos de trabajo:

1. **Requerimiento:** en este flujo de trabajo se realizan entrevistas con el cliente para comprender el problema a resolver y se definen los requerimientos.
2. **Análisis y Diseño:** se realiza el diseño de los requisitos que serán implementados posteriormente.
3. **Implementación**: se realiza la implementación del sistema, basándose en el diseño realizado.
4. **Prueba**: busca los defectos a lo largo del ciclo de vida.
5. **Gestión del Proyecto**: involucra actividades, con las que se busca producir un producto que satisfaga las necesidades de los clientes.
6. **Gestión de Configuración y Cambios**: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto, en cuanto a: utilización y actualización, control de versiones, etcétera.

XP

Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requerimientos para llegar al éxito del proyecto. Es utilizada en proyectos de corto plazo y para equipos de desarrollo pequeños.

¿Qué es lo que propone XP?

1. Empieza en pequeño y añade funcionalidades con retroalimentación continua.
2. El manejo del cambio se convierte en parte sustancial del proceso.
3. El costo del cambio no depende de la fase o etapa.
4. No introduce funcionalidades antes de que sean necesarias.
5. El cliente o el usuario se convierten en miembro del equipo.

Lo fundamental en este proceso de desarrollo, es lograr la comunicación entre los usuarios y los desarrolladores, la simplicidad al desarrollar, codificar los módulos del sistema y la retroalimentación del equipo de desarrollo, el cliente y los usuarios finales. Este proceso de desarrollo de *software* tiene como principales normas: la planificación, el diseño, la codificación y las pruebas. (Canós 2015)

RUP

El Proceso Unificado de Desarrollo, es un proceso de software genérico que provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible. (Roberto Curlango 2011)

RUP propone cómo deben ser ejecutadas las actividades para obtener un producto de software y la forma de documentar todas las acciones que se llevan a cabo en las mismas, soportando el ciclo completo de desarrollo de la aplicación. (Jacobson 2004)

Tabla 1: Comparación entre OpenUP, XP, RUP.

	OpenUP	XP	RUP
Dirigido por casos de uso	SÍ	NO	SÍ
Desarrollo iterativo e incremental	SÍ	NO	SÍ
Participación activa del usurario	NO	SÍ	NO
Adaptación del proceso	SÍ	SÍ	SÍ
Centrado en la arquitectura	SÍ	NO	SÍ

Luego de realizar el análisis de las metodologías anteriormente mencionadas, se decidió escoger OpenUP para el proceso de desarrollo, debido a que es iterativo e incremental. Este es apropiado para proyectos pequeños y de bajos recursos; permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito; y permite detectar errores tempranos a través de un ciclo iterativo. Además, evita la elaboración de documentación, diagramas e iteraciones innecesarias. Por ser una metodología ágil, tiene un enfoque centrado al cliente, con iteraciones cortas; es ligero y proporciona una comprensión detallada del proyecto, beneficia a clientes y desarrolladores sobre el producto a entregar y su formalidad; y es extensible, debido a que el proceso se puede agregar o adaptar, según lo vayan requiriendo los sistemas.

Conclusiones parciales

En el presente capítulo, se abordaron los elementos fundamentales relacionados con la integración de los sistemas informáticos, definiéndose los conceptos básicos asociados al objeto de estudio; esto contribuyó a lograr un buen desarrollo de la propuesta de solución. Con el análisis de las soluciones internacionales y nacionales, enfocadas a la integración de aplicaciones informáticas, se demostró, que ninguna de ellas resuelve el problema en cuestión. Dicho análisis, sirvió de apoyo, logrando un mejor entendimiento de estos sistemas; además se identificaron características para el diseño y desarrollo de la solución informática propuesta.

Capítulo 2: Propuesta de la solución informática

El presente capítulo está dedicado a las características de la solución informática. En él se abordan temas relacionados con la propuesta de solución, para responder a la situación problemática en cuestión; además quedarán definidos los requerimientos no funcionales, así como las principales funcionalidades del sistema. También se identifican los actores, casos de uso y las relaciones existentes entre ellos. Además, se caracterizan las plataformas PLATSI y OSSIM según el manejo de datos, vulnerabilidades, formato de exportación y otros elementos.

2.1 Análisis de las plataformas PLATSI y OSSIM, identificando las vulnerabilidades que deben ser incluidas en la propuesta de solución

La plataforma PLATSI, utiliza diversas herramientas para el realizar auditorías en aplicaciones web, como Zap, Nixto, Acunetix, W3af y Arachni. Una vez finalizada la auditoría, esta es evaluada por un especialista para generar un informe, donde quedan plasmadas las vulnerabilidades reales que existen, a través del conocimiento empírico adquirido durante los años de trabajo.

La plataforma OSSIM, también detecta vulnerabilidades a través de la herramienta Nessus, además permite mediante el conocimiento adquirido, mediante el uso de la inteligencia artificial, durante el transcurso del tiempo, conocer cuándo se está en presencia de vulnerabilidades reales. Esta plataforma también permite realizar correlación con otras informaciones alojadas en ella, garantizando una información más completa del diagnóstico de la seguridad, durante la auditoría. A pesar de que ambas plataformas trabajan en la misma área y realizan reportes en igual formato, presentan diferencias como: variadas representaciones de datos, no manejan de igual forma las vulnerabilidades y trabajan con herramientas diferentes, aunque ocasionalmente utilizan herramientas comunes.

Una vez realizado el análisis, sobre la representación de los datos y herramientas utilizadas por ambas plataformas, se decide incorporar solo las vulnerabilidades que fueron verificadas por el especialista de la plataforma PLATSI, realizándole una transformación de los datos para que puedan ser entendidos y procesados por la plataforma OSSIM.

2.2 Características de la propuesta de solución

La solución propuesta consiste en dos *plugin*: plugin_PLATSI y plugin_OSSIM. Con la utilización de estos *plugin*, se logrará una integración entre ambas plataformas, permitiendo la correlación y elaboración de las vulnerabilidades generadas por la plataforma PLATSI en la plataforma OSSIM. A continuación, se describen ambos *plugin*:

plugin_PLATSI es el encargado de:

- Crear *log* con las vulnerabilidades verificadas
- Eliminar *log* después de enviar
- Enviar *log* al servidor de OSSIM
- Salvar *log* en caso de fallos
- Verificar que exista conexión con el servidor de OSSIM
- Establecer conexión con el servidor de OSSIM

Para crear un *log*, las vulnerabilidades deben estar verificadas por el especialista de PLATSI; una vez realizado este proceso, se verifica la existencia de una conexión con el servidor de OSSIM, y en caso de existir, se establece la conexión mediante un túnel, utilizando el protocolo SSH. Luego de establecida la comunicación cliente/servidor, se envía la información del *log* y esta es obtenida y almacenada en un fichero por el servidor de OSSIM.

El proceso de envío y recepción de información entre cliente/servidor se realiza de la siguiente forma: cuando el *datagrama* con los datos es enviado, se adiciona una función resumen mediante la cual se realiza una comparación, garantizando la integridad de los datos. En el servidor se verifica que los datos recibidos sean iguales a los enviados por el cliente. Partiendo de la verificación anterior, en el caso de existir igualdad entre los datos enviados y recibidos, el servidor envía una notificación al cliente y se elimina el *log* del cliente, quedando solamente almacenado en el servidor; cuando el cliente no recibe la notificación del servidor, se procede a realizar una copia de seguridad del *log* enviado, para un posterior reenvío.

plugin_OSSIM es el encargado de:

- Verificar la conexión
- Transformar *log* en datos

Cuando el servidor de OSSIM recibe los *log* enviados, este verifica que la información es la correcta, generando una función resumen y comparándola con la recibida; si esta es correcta

se le notifica al cliente y procede a guardar los datos en un fichero para posteriores usos; en caso de no ser iguales las funciones resumen, no se notifica y se procede a desechar los datos. Cuando son solicitados los datos, para realizar alguna de las operaciones permitidas por la plataforma OSSIM, como correlacionar la información con otra existente, se traducen los *log* en datos entendibles para dicha plataforma.

2.3 Modelo de Dominio

En un modelo de dominio, se capturan los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde se despliega el sistema, trayendo consigo la ventaja de ayudar a usuarios, clientes y desarrolladores a utilizar un vocabulario común, para entender el contexto en que se desarrolla el sistema. (Pressman, Roger S. 2002)

El modelo de dominio se realiza con el objetivo de definir el flujo de eventos, que se genera a partir de las vulnerabilidades detectadas durante una auditoría web, que se realiza con cualquiera de las herramientas de la plataforma PLATSI, hasta llegar a la correlación en la plataforma OSSIM. Este modelo permite obtener las funcionalidades necesarias para que el sistema propuesto realice la gestión de vulnerabilidades, detectadas por las herramientas de la plataforma PLATSI; además, contribuye a identificar los principales conceptos asociados al problema en cuestión y la relación existente entre ellos.

- **Log**: maneja las vulnerabilidades encontradas por la plataforma PLATSI.
- **Información de vulnerabilidades**: colección de datos que se genera por las plataformas PLATSI y OSSIM, referente a las vulnerabilidades detectadas.
- **PLATSI**: plataforma de seguridad informática para detectar vulnerabilidades en la web.
- **OSSIM**: plataforma de seguridad informática, que usa *plugin* para cargar los *log* de las vulnerabilidades detectadas por algunas de sus herramientas.
- **PLUGIN**: intérprete de *log*, que normaliza y clasifica los eventos de orígenes específicos.

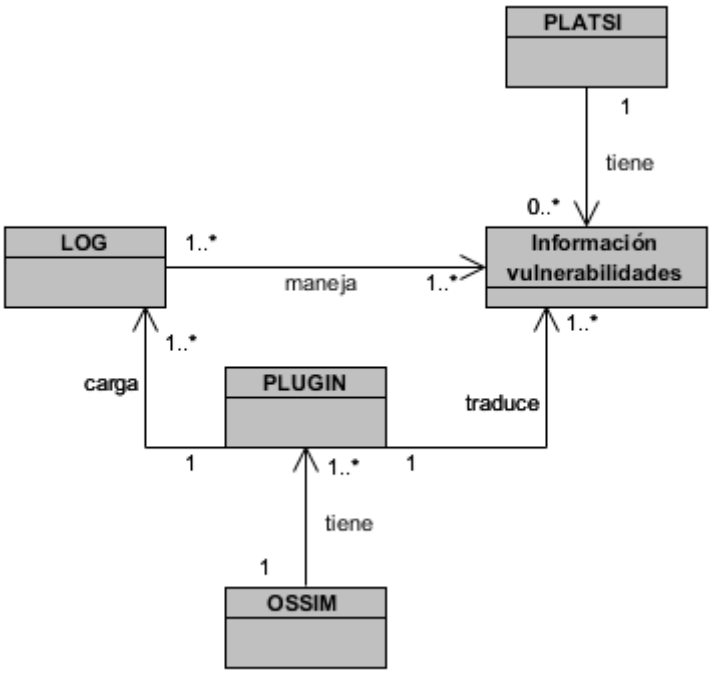


Figura 1: Modelo de Dominio.

2.4 Especificación de los requerimientos del sistema

Los requerimientos de *software* son capacidades o condiciones, que tienen que ser alcanzadas o poseídas por un sistema o componente de un sistema, para satisfacer un contrato, estándar u otro documento impuesto formalmente. Los mismos definen las funciones que el sistema es capaz de emitir, para lo que se identifican las funcionalidades requeridas y las restricciones que se imponen. Estos son el conjunto de propiedades que debe poseer el *software* para ser exitoso, en el entorno en el cual se usará y deben ser comprensibles por clientes, usuarios y desarrolladores, deben tener una sola interpretación y estar definidos de forma medible y verificable. (Pressman, Roger S. 2002)

Requisitos funcionales

Los requisitos funcionales son aquellos que describen qué debe hacer el sistema, o sea, especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto. Desde el punto de vista de las necesidades del usuario, son capacidades o condiciones con las que debe cumplir el sistema y que están fuertemente ligadas a las opciones del programa. (Pressman, Roger S. 2002)

A continuación, se muestran los requisitos funcionales del sistema:

1. RF1. Crear *log* con las vulnerabilidades verificadas
2. RF2. Eliminar *log* después de enviar
3. RF3. Enviar *log* al servidor de OSSIM
4. RF4. Salvar *log* en caso de fallos
5. RF5. Verificar que exista conexión con el servidor de OSSIM
6. RF6. Establecer conexión con el servidor de OSSIM
7. RF7. Transformar *log* en datos

Tabla 2: Descripción de los requisitos funcionales.

No.	Descripción	Prioridad	Complejidad
RF1	Obtiene las vulnerabilidades que han sido verificadas en la plataforma PLATSI y realiza las transformaciones necesarias para obtener la estructura correcta de la plataforma OSSIM.	Alta	Media
RF2	Elimina los *log* verificados.	Alta	Baja
RF3	Envía los *log* al servidor de OSSIM a través de un túnel SSH.	Alta	Alta
RF4	Salva los *log* que no fueron recibidos por el servidor de OSSIM.	Alta	Alta
RF5	Verifica la disponibilidad del servidor de OSSIM.	Alta	Baja
RF6	Establece la conexión con el servidor de OSSIM utilizando el protocolo SSH.	Alta	Media
RF7	Transforma los *log* recibidos por el servidor de PLATSI en datos, para posteriores usos en la plataforma OSSIM.	Alta	Media

Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener, forman una parte significativa de la especificación. Estos tienen la característica de marcar la diferencia, pues una vez comprobado que el producto cumple con lo requerido por el cliente, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable es,

distinguen a un producto bien aceptado de uno con poca aceptación. (Pressman, Roger S. 2002)

Para un correcto funcionamiento, el sistema debe tener como características principales, los siguientes requisitos no funcionales:

RNF1. Seguridad: la comunicación entre las plataformas PLATSI y OSSIM, debe poseer un mecanismo de seguridad, que garantice la confidencialidad de la información durante el envío de datos. Para garantizar que la información no sea capturada durante el proceso de envío se deberá establecer un protocolo de seguridad SSH, permitiendo que los paquetes de información viajen por la red de forma cifrada.

RNF2. Confidencialidad: se debe tratar el manejo de permisos, de forma tal que solamente se envíe al servidor, la información autorizada, de acuerdo a los niveles de permisos del sistema. La configuración de la solución, deberá establecer los parámetros requeridos para el envío de datos, estableciendo el servidor que debe recibir la información, de esta forma se garantizará que la información no sea expuesta a personal indebido.

RNF3. Integridad: se deberá tratar el manejo de la información de forma tal, que la información no sea modificada por personal ajeno. Se establecerá una función resumen (HASH, por sus siglas en inglés) utilizando el Algoritmo de Hash Seguro en el primer nivel (SHA1, por sus siglas en inglés) y se comparará la integridad de la información recibida en el servidor.

RNF4. Eficiencia: la solución deberá atender las solicitudes realizadas entre las plataformas en un tiempo no mayor a 0.05 segundos.

RNF5. Escalabilidad: la solución deberá ser escalable, garantizando incorporar nuevas funcionalidades, así como su mantenimiento. La solución deberá ser capaz de garantizar la adición de nuevas herramientas incorporadas por la plataforma PLATSI.

RNF6. Usabilidad: la solución estará orientada a especialistas con conocimientos de redes, comandos en consola, además de poseer conocimientos de las plataformas PLATSI y/o OSSIM.

RNF7. Hardware:

Servidor: las prestaciones del servidor OSSIM varían en dependencia de la red a la que se desea monitorear y de los recursos que la empresa disponga para destinarle al mismo. Los

datos que se muestran fueron tomados del centro XETID en la UCI, en una red pequeña con pocos activos. En este caso las prestaciones mínimas son:

- Procesador i7 a 3.60 GHz, 4MB caché L3 o similar
- RAM 16GB a 1600 bus
- HDD 500GB SATA 3 7200 RPM 32MB caché
- Realtek PCI 100MB o similar, tarjeta de red para la gestión del servido
- Realtek PCI 100MB o similar, tarjeta de red para la captura de tráfico en modo promiscuo

En el caso del servidor PLATSI, en una red pequeña con pocos activos, las prestaciones mínimas son:

- Procesador P4 a 1.60 GHz, 512M caché L2 o similar
- RAM 512MB a 360 bus
- HDD 100GB IDE 3200 RPM 20MB caché
- Realtek PCI 100MB o similar, tarjeta de red para la gestión del servidor

RNF.8 Software: para la plataforma PLATSI, debe estar montada la versión 1.1 o superior de PLATSI y en el servidor de OSSIM el sistema operativo AlienVault_OSSIM_64bits en su versión 5.2 o superior.

2.5 Actores del sistema

Representan papeles que las personas (o dispositivos) juegan como impulsores del sistema. Definido más formalmente, un actor es algo que comunica con el sistema o producto y que es externo al sistema en sí mismo. (Pressman, Roger S. 2002)

Tabla 3: Descripción de los actores de la solución.

Actor	Descripción
plugin_PLATSI[1]	Actor principal del sistema, debido que es el encargado de verificar las vulnerabilidades detectadas por cualquiera de las herramientas de la plataforma PLATSI y enviarlas a la plataforma OSSIM.

[1] plugin_PLATSI es un servicio montado en la plataforma PLATSI que se encarga de establecer una comunicación con la plataforma OSSIM mediante el protocolo de seguridad SSH, enviando al servidor de OSSIM las vulnerabilidades detectadas.

plugin_OSSIM[2]	Es el encargado de leer los *log* enviados por PLATSI y transformar los datos para que sean usados por la plataforma OSSIM.
Usuario	Generalización entre los actores plugin_PLATSI y plugin_OSSIM

2.6 Casos de uso del sistema

Los casos de uso modelan el sistema desde el punto de vista del usuario y son creados durante la obtención de requisitos; estos describen la manera en que los actores interactúan con el sistema (Pressman, Roger 2001). A continuación, se listan los casos de usos de la solución informática.

Tabla 4: Listado de los casos de uso del sistema.

Nombre del caso de uso	
CUS 1- Crear *log*	CUS 5- Salvar *log*
CUS 2- Establecer conexión	CUS 6- Transformar *log* en datos
CUS 3- Enviar *log*	CUS 7- Verificar conexión
CUS 4- Eliminar *log*	

2.7 Diagrama de Casos de uso del sistema

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema; este representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa (Pressman, Roger 2001). A continuación, se muestra en la Figura 2, el diagrama de Casos de Uso del Sistema.

[2] plugin_OSSIM es un *plugin* adicionado a la plataforma OSSIM, que se encarga de leer los *log* enviados por PLATSI.

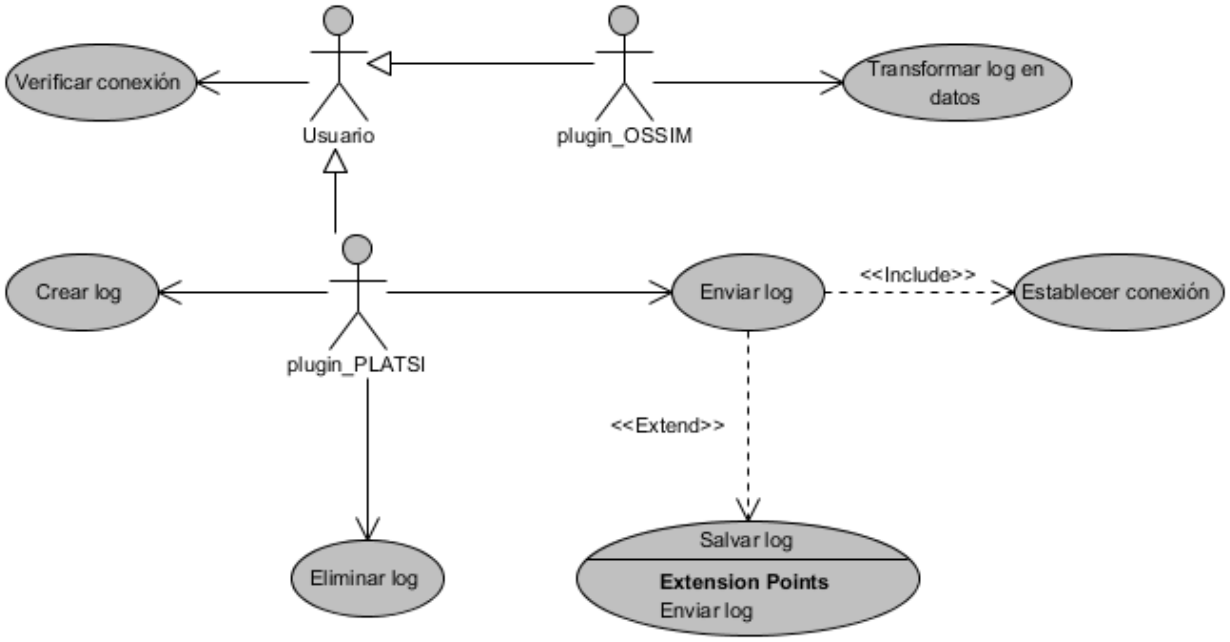


Figura 2: Diagrama de Casos de Uso del Sistema.

2.8 Descripción textual de casos de uso del sistema

A continuación, se muestra la descripción de algunos casos de usos del sistema.

Caso de uso del sistema Enviar *log*

Tabla 5: Descripción del caso de uso Enviar log.

Caso de uso	Enviar *log*
Actores	plugin_PLATSI
Resumen	Se inicia automáticamente. Se verifica la disponibilidad del servidor de OSSIM. Se establece una conexión SSH entre los servidores de PLATSI y OSSIM. Se envía el *log* por el puerto establecido por la conexión. El caso de uso termina cuando se envía el *log*.
Precondiciones	Que se ejecute el sistema PLATSI
Referencias	RF 3, RF 4, RF 5,
Prioridad	Alto
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El caso de uso inicia automáticamente cuando el sistema PLATSI es ejecutado.	2. Se inicia un servicio en PLATSI, verificando la disponibilidad del servidor de OSSIM; si OSSIM está disponible ver **Flujo Alterno 1. OSSIM disponible**; si OSSIM no está disponible ver **Flujo Alterno 2. OSSIM no disponible.**
Flujos Alternos	
Flujo Alterno 1. OSSIM disponible	
Acción del Actor	**Respuesta del Sistema**
	3. Se establece una conexión SSH con el servidor de OSSIM. 4. Se cargan los *log* y se envían al servidor de OSSIM.
Flujo Alterno 2. OSSIM no disponible	
Acción del Actor	**Respuesta del Sistema**
	5. Al no estar disponible OSSIM, plugin_PLATSI intenta la conexión con un intervalo de cinco segundos.
Poscondiciones	Queda establecida la conexión.

Caso de uso del sistema Crear *log*

Tabla 6: Descripción del caso de uso Crear *log.*

Caso de uso	Crear *log*
Actores	plugin_PLATSI
Resumen	Se inicia automáticamente. Se verifica la existencia de nuevas vulnerabilidades detectadas por las herramientas de la plataforma PLATSI. El caso de uso termina cuando se crea el *log*.
Precondiciones	Que se ejecute el sistema PLATSI
Referencias	RF 1
Prioridad	Alto
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1- El caso de uso inicia automáticamente cuando el sistema PLATSI es ejecutado.	2- Se inicia una función disparadora, verificando la existencia de nuevas vulnerabilidades verificadas, si existen ver **Flujo Alterno 1. Vulnerabilidades verificadas**; si no existen ver **Flujo Alterno 2. Vulnerabilidades no verificadas.**
Flujos Alternos	
Flujo Alterno 1. Vulnerabilidades verificadas	
Acción del Actor	**Respuesta del Sistema**
	3- Se realiza una limpieza y transformación a los datos, acorde a la plataforma OSSIM. 4- Se crea una estructura de *log* con los datos de las vulnerabilidades.
Flujo Alterno 2. Vulnerabilidades no verificadas	
Acción del Actor	**Respuesta del Sistema**
	5- Al no haber vulnerabilidades verificadas, no se crea el *log*.
Poscondiciones	Se crea el *log*

Conclusiones parciales

La aplicación de la metodología de desarrollo seleccionada, permitió modelar los artefactos necesarios para lograr un buen desarrollo de la solución informática propuesta.

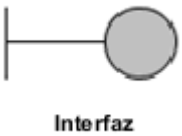
Capítulo 3: Análisis y Diseño, Implementación y Pruebas de la solución informática

En el presente capítulo se abordan los temas relacionados al análisis y el diseño de la solución informática, además se desarrollan los artefactos necesarios de la fase de implementación. Una vez concluida esta última, se aplican las pruebas de seguridad, funcionalidad y eficiencia, validando el correcto funcionamiento del sistema.

3.1 Diagrama de clases del análisis

El diagrama de clase del análisis se realiza para cada caso de uso del sistema; el mismo muestra las clases participantes en dichos casos de uso, así como la relación entre ellas; y en él se identifican tres tipos de clases: Interfaz, Controladora y Entidad (Jacobson 2004). En la siguiente tabla se representan los estereotipos a utilizar en el diagrama de clases del análisis.

Tabla 7: Estereotipos a utilizar en el diagrama de clases del análisis.

Clases	***Descripción***	***Representación***
Entidad	Las clases *entidad* se utilizan para modelar información que posee larga vida y que es a menudo persistente.	
Controladora	Las clases *controladora* representan coordinación, secuencia, transacciones y control de objetos y son utilizadas para encapsular el control de un caso de uso.	
Interfaz	Las clases *interfaz* se utilizan para modelar la interacción entre el sistema y sus actores, además se utilizan para mostrar comunicación entre sistemas.	

A continuación, se muestran los diagramas de clase del análisis de los casos de uso Enviar *log* y Crear *log*. El resto de los diagramas se encuentran en los anexos del presente documento.

Figura 3: Diagrama de Clase del Análisis CU Enviar log.

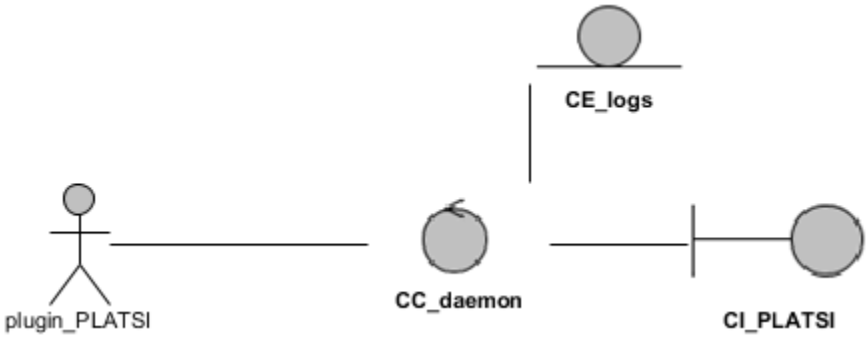


Figura 4: Diagrama de Clase del Análisis CU Crear log.

3.2 Diagrama de clases del diseño (DCD)

El diagrama de clases, es el diagrama principal de diseño de un sistema. En él, se especifica la estructura de clases del sistema y relaciones entre sus clases. (Jacobson 2004)

CU Enviar log

La Figura 5 muestra las clases utilizadas para el envío de *log*. En este caso, la clase logsrv le solicita a la clase daemon que envíe los *log* a la plataforma OSSIM, esta obtiene los parámetros de conexión de la clase parameters con el objetivo de realizar una consulta a la tabla log_db permitiendo el envío de *log*.

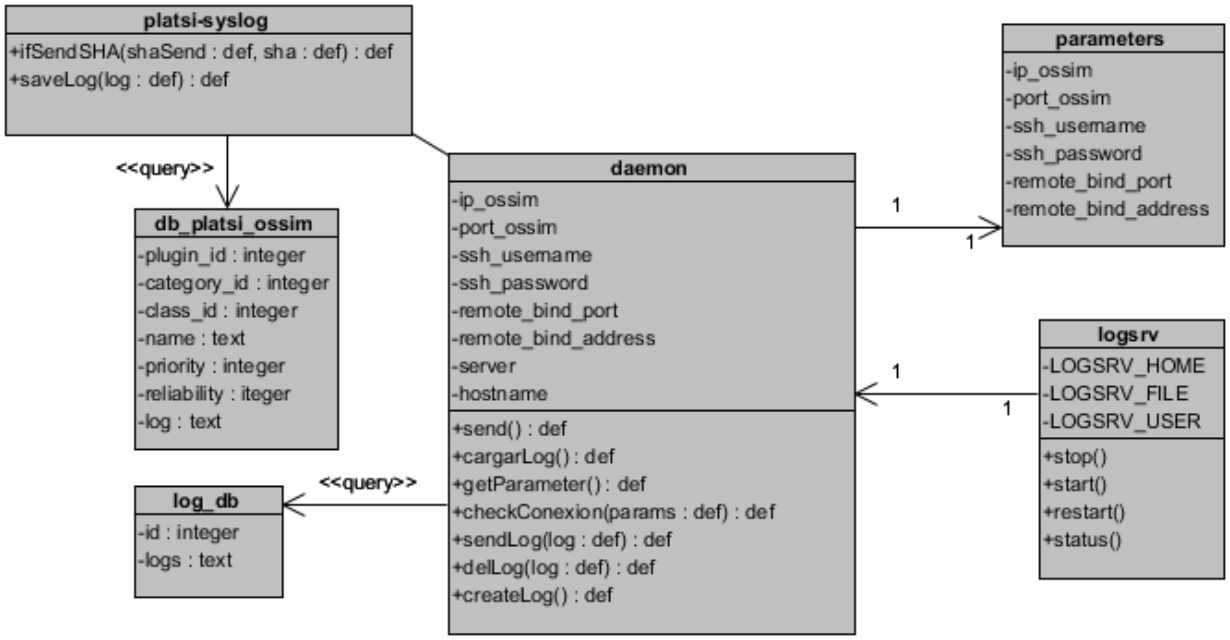


Figura 5: DCD CU Enviar log.

CU Crear log

La Figura 6 muestra las clases utilizadas para crear *log*. En este caso la clase logsrv le solicita a la clase send_ossim_triger que cree los *log* de vulnerabilidades verificadas por el especialista de la plataforma PLATSI, realizando una consulta a la base de datos, concatenando con cada una de las tablas que se muestran en el diagrama.

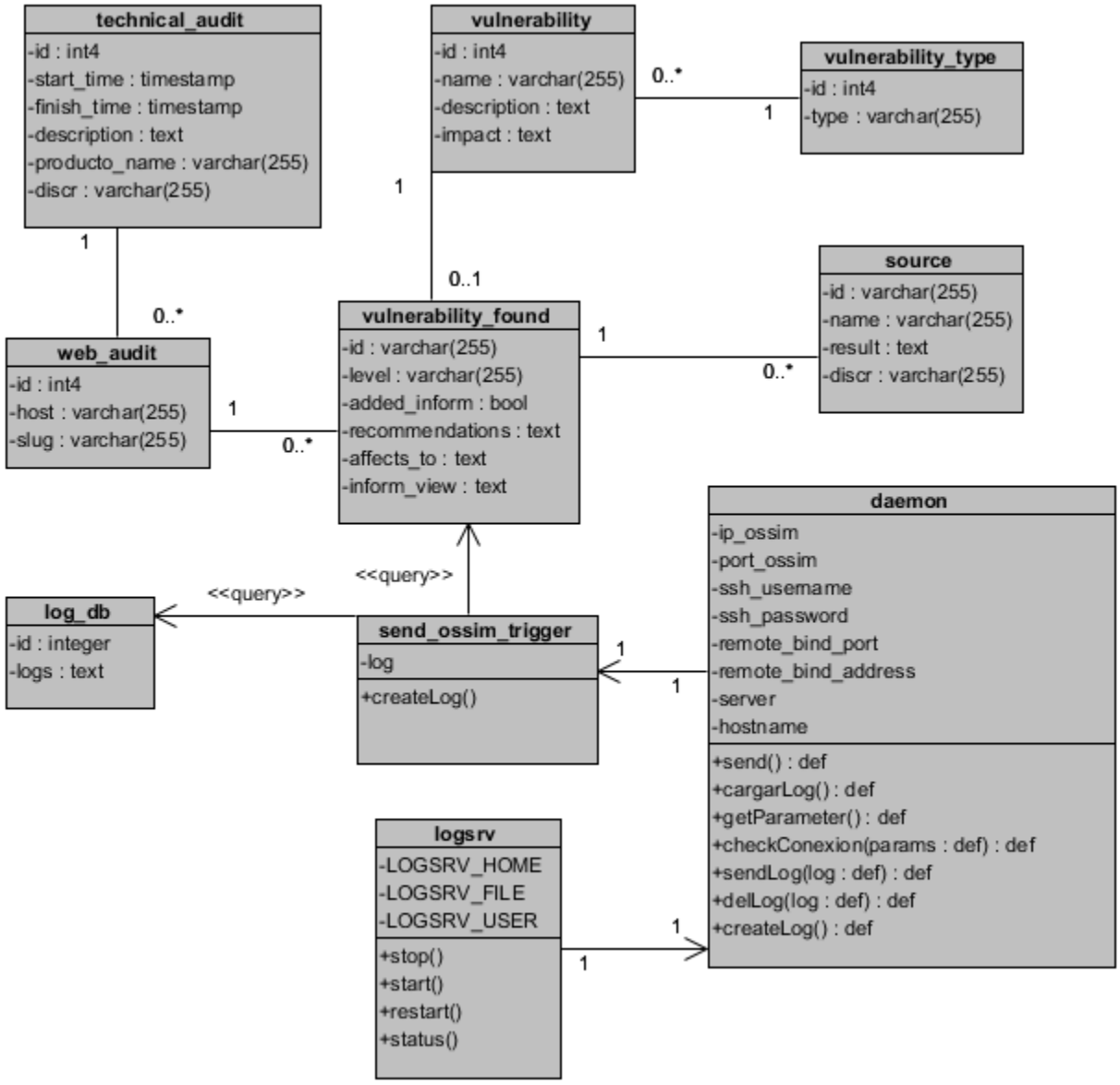


Figura 6: DCD CU Crear log.

Descripción de las clases más relevantes del diseño

Una clase del diseño es similar a una clase de implementación del sistema y tiene prioridades (atributos), comportamientos (métodos) y relaciones con otras clases.

Tabla 8: Descripción de la clase logsrv.

Nombre	Logsrv
Tipo de clase	Servicio
Atributos	LOGSRV_HOME, LOGSRV_FILE, LOGSRV_USER
Principales responsabilidades	
Nombre	stop()
Descripción	Se encarga de detener la ejecución del servicio
Nombre	start()
Descripción	Se encarga de iniciar la ejecución del servicio
Nombre	restart()
Descripción	Se encarga de reiniciar la ejecución del servicio
Nombre	status()
Descripción	Se encarga de identificar el estado del servicio

Tabla 9: Descripción de la clase daemon.

Nombre	Daemon
Tipo de clase	Controladora
Atributos	Ip_ossim, port_ossim, ssh_username, ssh_password, remote_bind_port, remote_bind_address, server, hostname
Principales responsabilidades	
Nombre	send()
Descripción	Se encarga de realizar todas las operaciones previas para enviar los *log* al servidor de OSSIM.
Nombre	cargarLog()
Descripción	Se encarga de obtener los *log* para su posterior envío.
Nombre	getParameter()
Descripción	Se encarga de obtener los parámetros de conexión con el servidor de OSSIM.
Nombre	checkConexion(params : def)
Descripción	Se encarga de verificar la existencia de conexión con el servidor de

	OSSIM.
Nombre	sendLog(log : def)
Descripción	Se encarga de enviar los *log* al servidor de OSSIM.
Nombre	delLog(log : def)
Descripción	Se encarga de eliminar los *log.*
Nombre	createLog()
Descripción	Se encarga de delegar la responsabilidad a la funcionalidad encargada de crear los *log.*

3.3 Diagrama de interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema y muestran un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos (Jacobson 2004). Entre los diagramas de interacción se encuentran los diagramas de secuencia.

Diagrama de secuencia

Un diagrama de secuencia destaca el orden de los mensajes, mostrando una secuencia entre ellos, se ubican a lo largo del eje X, ordenados según suceden en el tiempo. (Jacobson 2004)

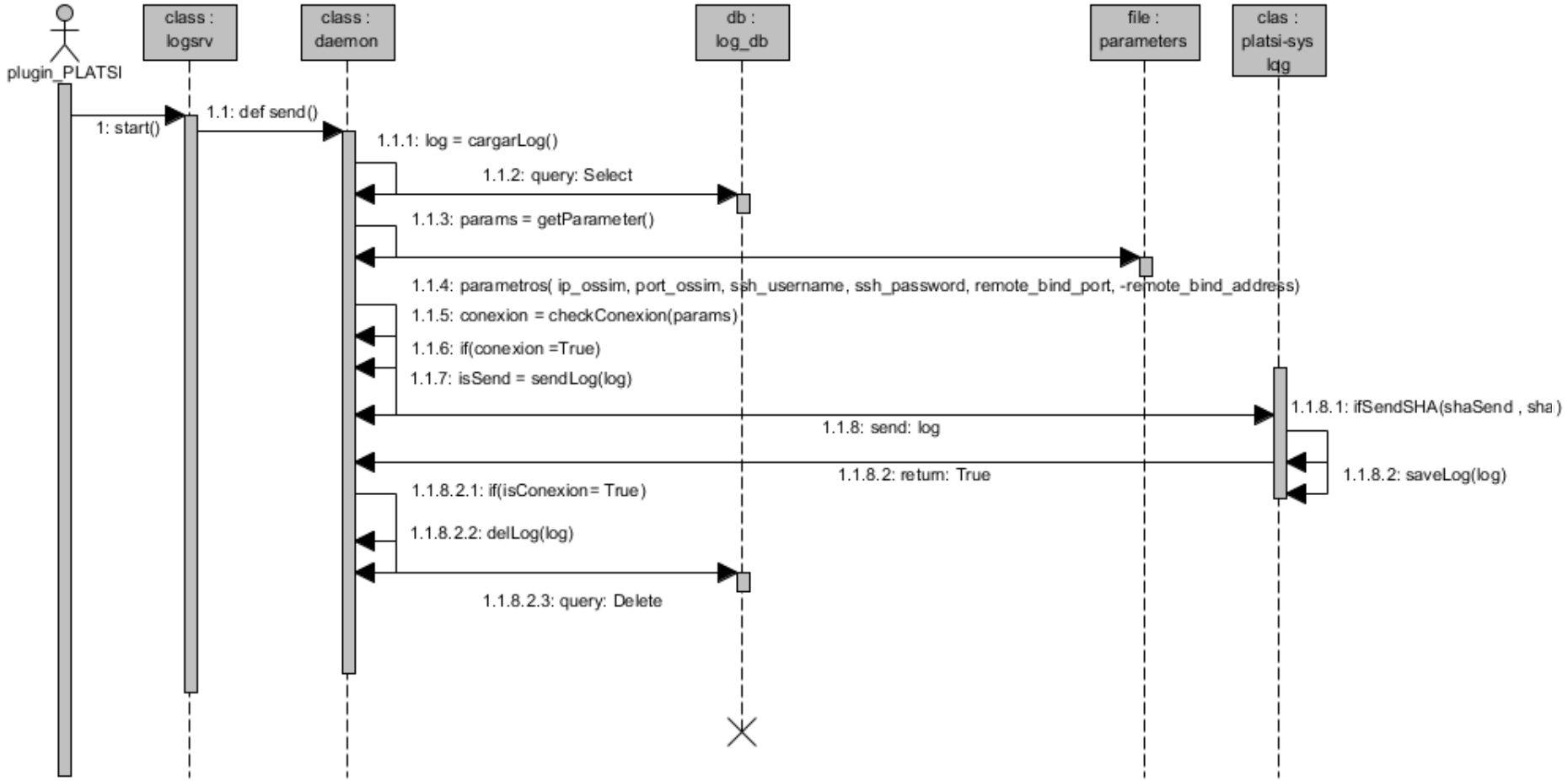


Figura 7: Diagrama de secuencia Enviar log.

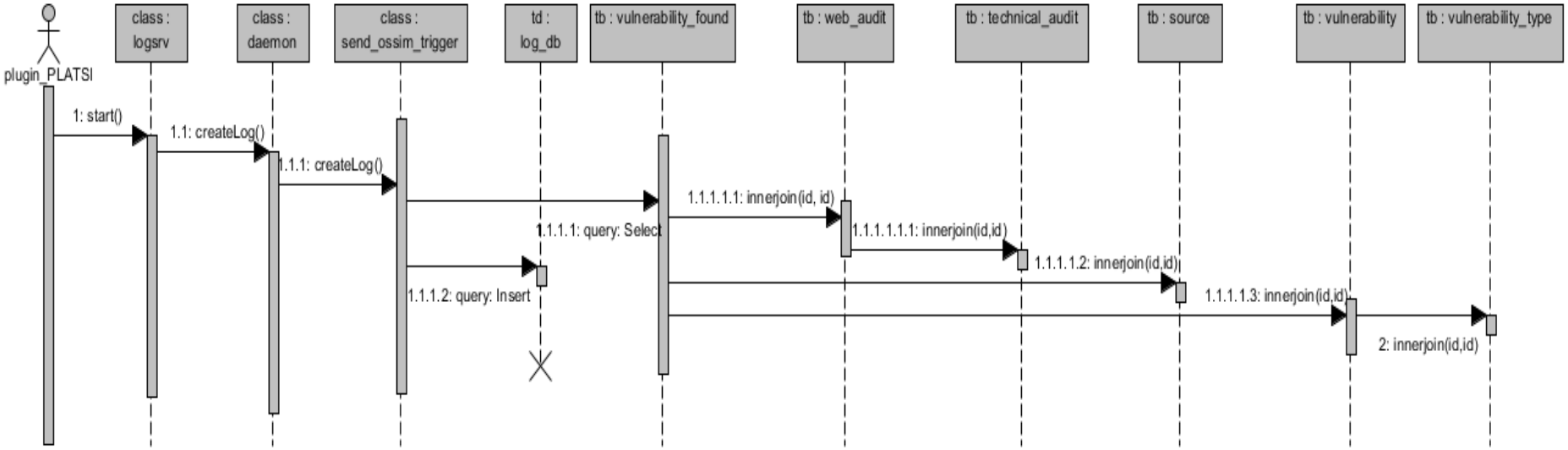


Figura 8: Diagrama de secuencia Crear log.

3.4 Arquitectura del sistema

La arquitectura de *software*[3] se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos. Esta representa un diseño de alto nivel del sistema, que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificable) y servir como guía en el desarrollo. (L. Bass, P. Clements, R. Kazman 2003)

Arquitectura Cliente-Servidor

La arquitectura cliente-servidor, es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes, que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos, y servidor al proceso que responde a las solicitudes. En este modelo, las aplicaciones se dividen de forma tal que el servidor contiene la parte que debe ser compartida por varios usuarios y en el cliente permanece solo lo particular de cada usuario. (TAPSCOTT, ART 2000)

Las principales características (TAPSCOTT, ART 2000) de la arquitectura cliente-servidor son:

- Combinación de un cliente (que interactúa con el usuario) y un servidor (que interactúa con los recursos compartidos). El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de *software*, que maneja recursos compartidos tales como bases de datos, impresoras y módems.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo, como velocidad del procesador, memoria, velocidad y capacidades del disco.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes, distribuidas a lo largo de la red.
- Existe una clara distinción de funciones, basada en el concepto de servicio, que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.

3 es la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo

- Los clientes corresponden a procesos activos, en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo (esperan las peticiones de los clientes).
- No existe otra relación entre clientes y servidores, que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El ambiente es heterogéneo. La plataforma de *hardware* y el sistema operativo del cliente y del servidor, no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura, es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema cliente-servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas, sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

La siguiente imagen muestra el flujo Cliente-Servidor:

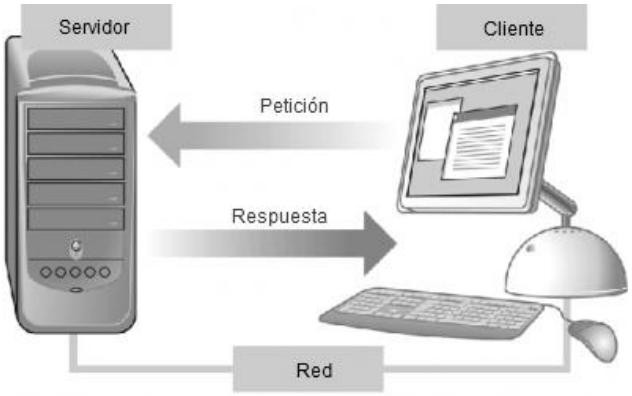


Figura 9: Representación gráfica del flujo cliente-servidor.

Las plataformas PLATSI y OSSIM implementan la arquitectura cliente/servidor, dando como resultado que la solución propuesta también implementará dicha arquitectura. Cuando PLATSI detecta una vulnerabilidad (a través de alguna de sus herramientas) y esta es verificada por algún especialista, se ejecuta una función disparadora, la cual crea un *log*. El servicio que establece la conexión entre las plataformas, verifica la existencia del *log* y crea una conexión con el servidor de OSSIM a través de un túnel SSH, por donde envía el *log*; este le responde

que fue recibido y el servicio procede a eliminarlo. De esta manera queda PLATSI como cliente y OSSIM como el servidor.

Patrones de diseño

Los patrones de diseño son descripciones de clases y objetos relacionados, que están particularizados para resolver un problema de diseño general, en un determinado contexto (Gamma, Erick 2012). Entre los más conocidos se encuentran los patrones de Asignación de Responsabilidades (GRASP, por sus siglas en inglés) y los patrones de la Banda de los Cuatro (GOF, por sus siglas en inglés).

Patrones GRASP

Los Patrones Generales de Asignación de Responsabilidades (GRASP), describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar *software* orientado a objetos. Los mismos no introducen ideas novedosas, sino que son la mera codificación de los principios básicos más usados. (Gamma, Erick 2012)

En el desarrollo de la presente aplicación se evidencia el empleo de los siguientes patrones GRASP:

Experto:

Este patrón constituye el principio básico de asignación de responsabilidades en diseño orientado a objetos. Se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. Se puede ver un ejemplo de la aplicación de dicho patrón a través de la siguiente imagen, que muestra como la clase daemon delega la responsabilidad de establecer una conexión SSH a la clase SSHTunnelForwarder, pues esta reúne la información necesaria para realizar dicha funcionalidad, lo que la hace ser, la clase experta en la información:

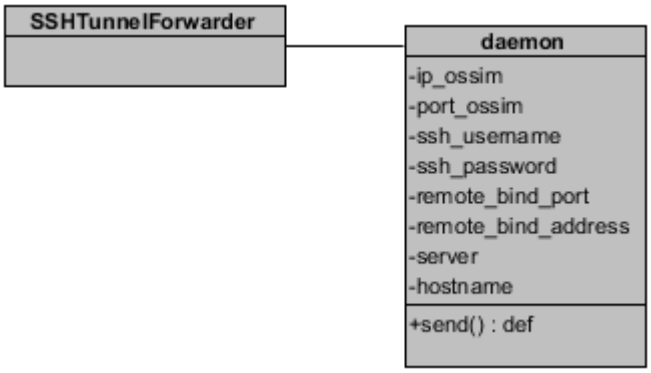


Figura 10: Patrón Experto en la solución.

Creador:

Guía la asignación de responsabilidades relacionadas con la creación de objetos. En el sistema se evidencia mediante la clase send_ossim_trigger, que es la responsable de crear una instancia de la clase log, o sea que send_ossim_trigger es un creador de los objetos *log*, como se muestra en la siguiente imagen:

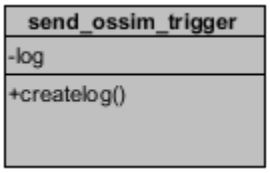


Figura 11: Aplicación del patrón Creador.

Alta cohesión:

La información que almacena una clase, debe de ser coherente y está en la mayor medida de lo posible, relacionada con la clase. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión, compartirá la responsabilidad de una operación, con otras clases. Se evidencia en la clase send_ossim_trigger, puesto que contiene toda la información necesaria para crear un log, que es su razón de ser.

Bajo acoplamiento:

Es la idea de tener las clases lo menos ligadas entre sí. De forma tal que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las

clases. Los beneficios de este patrón son: que un componente no se afecta por cambios de otros componentes, estos son fáciles de entender por separado y fáciles de reutilizar.

Patrones GOF

Dentro de los patrones GOF, utilizados en el diseño de la presente investigación para el desarrollo de subsistemas más pequeños, se encuentra el patrón Solitario o Singleton y el patrón Fachada. Los patrones GOF, se encuentran agrupados en 4 divisiones en dependencia de su responsabilidad, estas son: interacción, comportamiento, creacionales y estructurales. (Gamma, Erick 2012)

Observador observable:

Es un patrón de diseño, que define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. En la comunicación entre la plataforma PLATSI y OSSIM se ve referenciado cuando se establece una conexión entre ambas plataformas. PLATSI le hace una petición de disponibilidad a OSSIM, en caso de estar disponible devuelve uno y en caso contrario devuelve cero.

3.5 Diagrama de despliegue

En el diagrama de despliegue se muestra la distribución física de la solución, incluyendo su *software* y su *hardware*. Para cada uno de los componentes del diagrama, es necesario documentar las características técnicas requeridas. Esta información en el diagrama se representa a través nodos y sus relaciones.

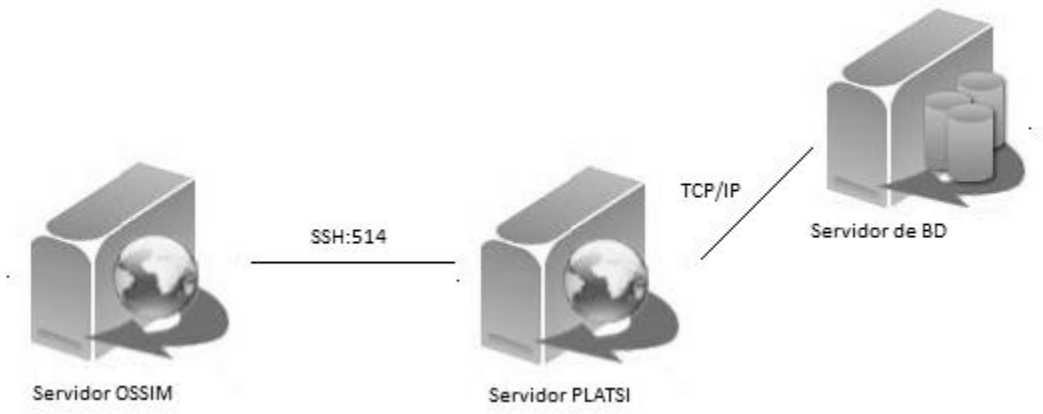


Figura 12: Diagrama de despliegue.

Descripción de los nodos físicos del sistema

- ✓ **Servidor PLATSI:** este nodo representa al servidor donde estará alojado el sistema PLATSI, así como los componentes almacenados en el mismo.
- ✓ **Servidor OSSIM:** este nodo representa al servidor donde estará alojado el sistema OSSIM, así como los componentes almacenados en el mismo.
- ✓ **Servidor de Base de Datos:** este nodo representa al servidor de base de datos PostgreSQL, en el que estará alojada la base de datos del sistema PLATSI.

Descripción de los protocolos utilizados:

- ✓ **SSH:** protocolo para facilitar las comunicaciones seguras entre dos sistemas, usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un *host* remotamente. A diferencia de otros protocolos de comunicación remota, tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.
- ✓ **TCP/IP:** proviene de dos protocolos, Protocolo de Control de Transmisión (TCP, por sus siglas en inglés) y Protocolo de Internet (IP, por sus siglas en inglés). Forma de comunicación básica que usa internet.

3.6 Modelo de implementación

El modelo de implementación, es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros de código fuente, y otros tipos de ficheros

necesarios para la implantación y despliegue del sistema. Describe también, cómo se organizan los componentes, de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación, en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes, unos de otros. (CARABALLO 2012)

Diagrama de componentes

Un diagrama de componentes muestra las dependencias entre los componentes de *software*. Este incluye los clasificadores que los especifican (por ejemplo: clases de implementación) y los artefactos que los implementan, tales como archivos de código fuente, archivos de código binario, archivos ejecutables, scripts. (PATRICIO 2014)

Los diagramas de componentes describen los elementos físicos del sistema como: ejecutables, librerías, clases y las relaciones entre ellos. Para la modelación del diagrama que a continuación se muestra, primeramente se identificaron los componentes, luego se agruparon los mismos por la arquitectura (en este caso cliente-servidor) y se representaron las relaciones entre ellos.

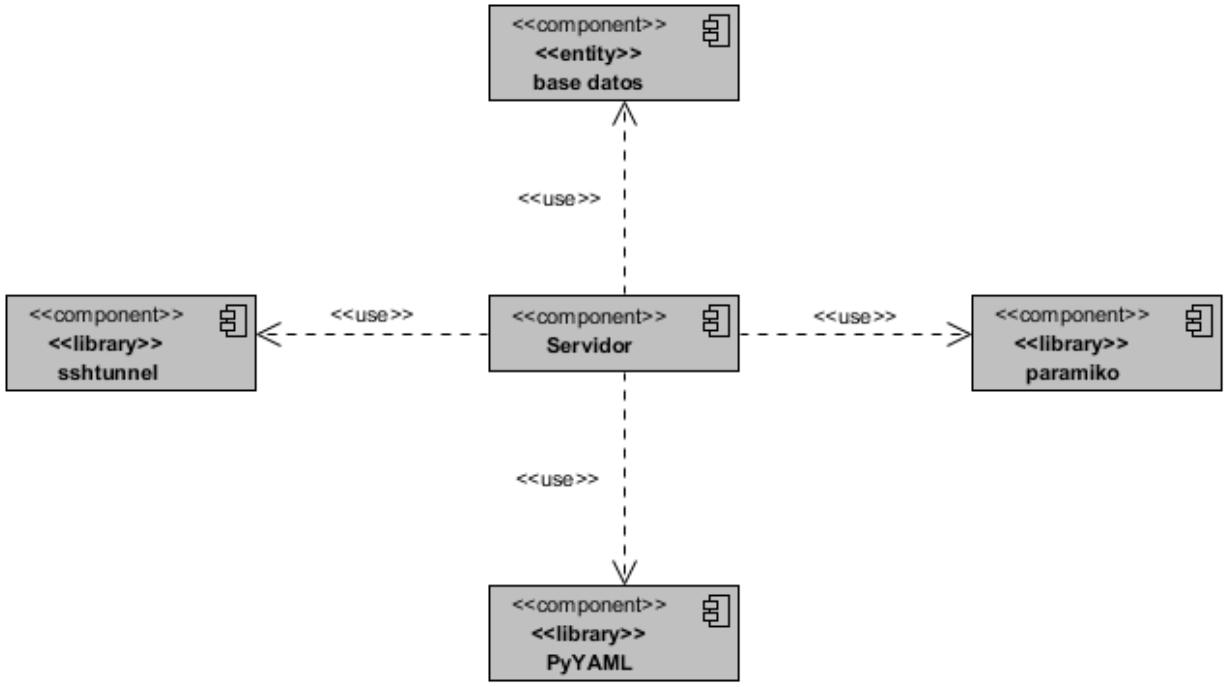


Figura 13: Diagrama de Componentes de la solución.

A continuación, se muestra la descripción de los componentes representados en el diagrama:

Tabla 10: Descripción de los componentes representados en el diagrama.

Componente	***Propósito***
SSHTUNEL	Librería externa para crear un túnel de información con protocolo SSH.
PyYAML	Librería externa para establecer los parámetros de configuración.
PARAMIKO	Librería externa para crear conexiones SSH
Servidor	Representa los servidores OSSIM y PLATSI
Cliente	Representa los clientes
Base de Datos	Representa la base de datos.

3.7 Pruebas

Las pruebas constituyen una actividad, en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

En todo proceso de desarrollo de aplicaciones, es indispensable la presencia de un proceso de pruebas de *software*, para garantizar así el buen funcionamiento y la calidad del producto final. Estas tienen como objetivo general verificar y validar un *software*, independientemente de las características y el entorno donde se desarrollen, además de los recursos y los factores vinculados al proceso de desarrollo. (Carlos, Blanco Bueno 2012)

Pruebas de seguridad

Las pruebas de seguridad tienen como objetivo hacer un análisis, con el fin de encontrar vulnerabilidades en el sistema, o sea, fallos de seguridad tanto en el diseño como en la implementación de la aplicación. Además, buscan medir la confidencialidad, integridad y disponibilidad de los datos. Las pruebas de seguridad del sistema se realizaron con la herramienta Wireshark en su versión 1.2. Como resultado se pudo comprobar que la comunicación entre ambas plataformas se realizaba de forma cifrada, lo cual garantiza la

confidencialidad de la información. En la Figura 14 se muestra un *datagrama* enviado desde la plataforma PLATSI a la plataforma OSSIM, donde se muestra en la parte derecha el contenido cifrado.

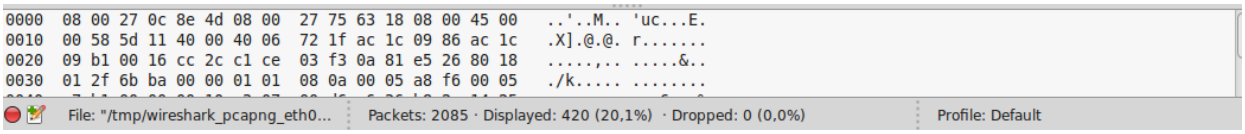


Figura 14: Trama de envío de datos con protocolo SSH.

Pruebas funcionales

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el *software*. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático. Dicho de otro modo, son pruebas específicas, concretas y exhaustivas para probar y validar que el *software* hace lo que debe y, sobre todo, lo que se ha especificado. (Pressman, Roger S. 2002)

A continuación, se muestran los resultados de las tres iteraciones de pruebas, realizadas con una muestra de veintiuna vulnerabilidades, detectadas por las herramientas de la plataforma PLATSI.

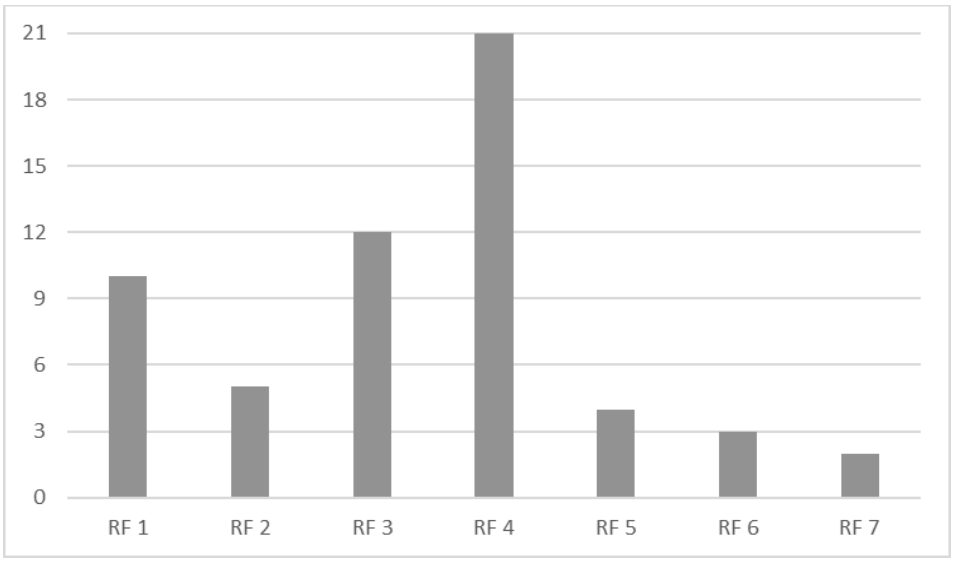


Figura 15: Primera iteración de las pruebas funcionales.

Después de realizar la primera iteración se encuentran un total de 57 errores (Ver Figura 15). Luego de corregidas, se realiza la segunda iteración cuyos resultados se muestran en la Figura 16.

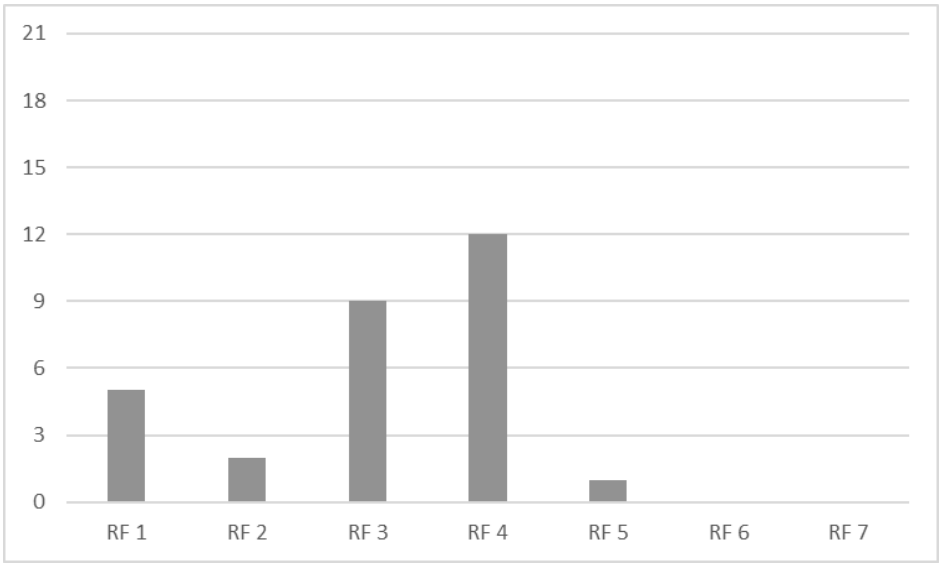


Figura 16: Segunda iteración de las pruebas funcionales.

Luego de realizada la segunda iteración, se detectaron veinticinco no conformidades (Ver Figura 16), las cuales fueron corregidas para posteriormente realizar una tercera iteración, donde se obtuvieron los resultados mostrados en la Figura 17.

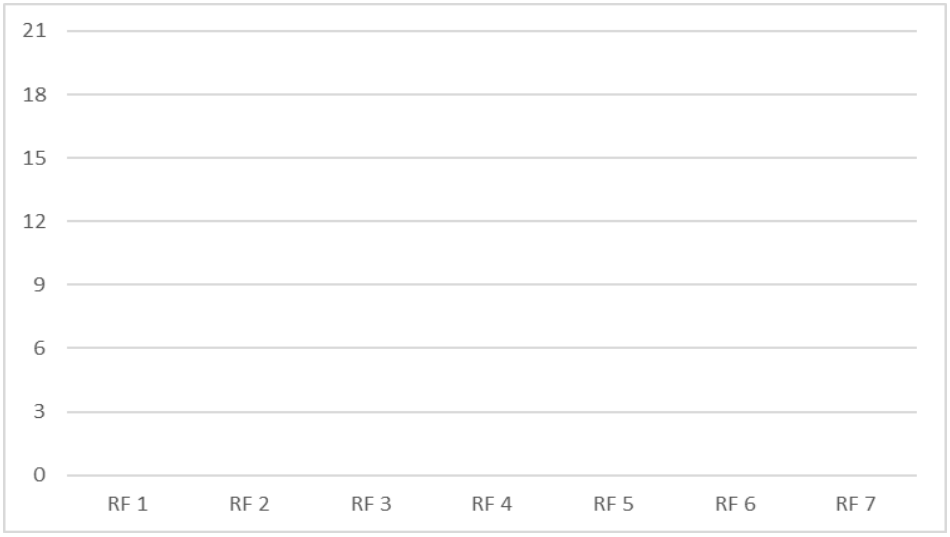


Figura 17: Tercera iteración de las pruebas funcionales.

Luego de aplicada la tercera iteración se alcanzaron valores satisfactorios (Ver Figura 17), demostrando que se cumplió el objetivo de las pruebas funcionales.

Pruebas de rendimiento

Prueban el rendimiento del *software* en tiempo de ejecución, determinando si los tiempos de respuesta, tanto en condiciones normales como en condiciones especiales, se encuentran dentro de los limites predefinidos (Sommerville, Ian 2005). Para realizar las pruebas de rendimiento en la solución, se empleó el intérprete de Python, haciendo uso de la librería “time”. El cálculo del tiempo se realizó empleando el tiempo de inicio y fin de las operaciones. Con una muestra de 100, 50 y 20 vulnerabilidades detectadas por las herramientas de la plataforma PLATSI, el tiempo de respuesta fue de 0.05 segundos, 0.03 segundos y 0.01 segundos respectivamente.

Las pruebas fueron realizadas con ambos servidores representados de forma virtual con las siguientes características: 2gb de memoria RAM y un microprocesador de un núcleo a 1ghz de velocidad en ambas plataformas, por lo que en condiciones reales se deben obtener mejores resultados.

Conclusiones parciales

Al realizar el análisis de las pruebas aplicadas a la solución informática, se concluye que la solución propuesta presenta un correcto funcionamiento, de esta manera quedó demostrado que el protocolo SSH permitió lograr la seguridad de la información.

Conclusiones

Al término de la presente investigación se concluye lo siguiente:

1. Al realizar el análisis de soluciones informáticas a nivel internacional y nacional, se identificó que ninguna resolvía el problema planteado, pero sirvieron como punto de partida para definir las características que debían cumplir la propuesta de solución.
2. La aplicación de patrones de diseño y arquitectónicos, permitió obtener una solución informática flexible.
3. El análisis de las plataformas PLATSI y OSSIM, permitió identificar las vulnerabilidades que debían ser incluidas en la solución informática.
4. La arquitectura de integración seleccionada, permitió la correcta comunicación entre las plataformas PLATSI y OSSIM.
5. Con el desarrollo de la propuesta de solución, se integró la información referente a las vulnerabilidades de la plataforma PLATSI en OSSIM, permitiendo la correlación de la misma.

Recomendaciones

Como recomendaciones al finalizar el proceso de desarrollo de la presente solución se tienen:

- Incorporar una funcionalidad a la plataforma OSSIM, que ordene a la plataforma PLATSI a realizar una auditoría web.

Referencias Bibliográficas

ALIENVAULT, 2016. OSSIM: The Open Source SIEM | AlienVault. *Detect, analyze, & respond to today's threats* [en línea]. [Consulta: 9 junio 2016]. Disponible en: https://www.alienvault.com/products/ossim.

CARABALLO, 2012. *Prolegómenos Sobre el Lenguaje de Modelado Unificado (UML).* [en línea]. 2012. S.l.: s.n. Disponible en: http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/prolego.pdf.

CARLOS, BLANCO BUENO, 2012. Construcción y Pruebas de Software. ,

CARRILLO P., ISAÍAS; PÉREZ G., RODRIGO; RODRÍGUEZ M., AURELIANO D., 2015. Metodología de Desarrollo del Software. [en línea], Disponible en: http://tecnicaytecnologiasc.wikispaces.com/file/view/Metodologias%20de%20desarrollo.pdf/409623082/Metodologias%20de%20desarrollo.pdf.

CORPORATION, 2016. InterSystems Ensemble | Software de Integración. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.intersystems.com/es/nuestros-productos/ensemble/vision-general/.

DATE, C.J, 1998. *Introducción a los sistemas de bases de datos*. S.l.: Person Educación. ISBN ISBM 968-444-419-290000.

EPF, 2011. OpenUP/Basic. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://epf.eclipse.org/wikis/openupsp/.

ETECSA, 2016. ETECSA Empresa de Telecomunicaciones de Cuba S.A. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.etecsa.cu/.

EUI-FI, 2015. *Introdución a herramientas CASE* [en línea]. 2015. S.l.: s.n. Disponible en: http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

GAMMA, ERICK, 2012. *Patrones de Diseño: elementos de software orientados a objetos reutilizables.* S.l.: s.n.

IBM, 2016. IBM MQ. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www-03.ibm.com/software/products/en/ibm-mq.

JACOBSON, 2004. *El Proceso Unificado de Desarrollo de Software* [en línea]. S.l.: s.n. Disponible en: http://quegrande.org/apuntes/EI/OPT/PAI/teoria/07-08/tema_2_-_el_proceso_unificado.pdf.

JEFF FORCIER, 2016. Welcome to Paramiko. [en línea]. Disponible en: http://paramiko.org.

L. BASS, P. CLEMENTS, R. KAZMAN, 2003. *Software Architecture in Practice 2nd Edition*. S.l.: Addison Wesley.

MICROSOFT, 2016. SQL Server Integration Services. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://msdn.microsoft.com/es-es/library/ms141026%28v=sql.120%29.aspx.

MICROSOFT PATTERNS & PRACTICES, 2013. *Integration Patterns*. US: Microsoft Patterns & Practices. ISBN ISBN-10/ISBN-13.

MILANÉS LÓPEZ, ALDO Y MONTANO GARCÍA, REINIER, 2009. *Sistema para la Gestión de la Tecnología en la Universidad de las Ciencias Informáticas*. La Habana: s.n.

MYSQL, 2015. MySQL. *MySQL* [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.mysql.com/.

ORACLE, 2014. Oracle Fusion Middleware | Oracle ES | Oracle España. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://www.oracle.com/es/middleware/index.html.

OREN BEN-KIKI, 2012. YAML Ain't Markup Language (YAML) version 1.2. [en línea]. Disponible en: http://yaml.org/spec/1.2/spec.html.

PATRICIO, 2014. Introducción a RUP. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://es.scribd.com/doc/51852648/Introduccion-a-RUP.

PHAZ BLINOV, 2016. Welcome to SSHTunnel's documentation. [en línea]. Disponible en: https://sshtunnel.readthedocs.org/en/latest.

PILOTFISH, 2016. Enterprise Integration Platform. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.pilotfishtechnology.com/enterprise-integration-platform/.

P.LETELIER, 2015. *Introducción Proceso de Desarrollo de SW1* [en línea]. S.l.: s.n. Disponible en: https://pid.dsic.upv.es/C1/Material/default.aspx.

PRESSMAN, ROGER, 2001. *Ingenieía de Software: Un Enfoque Práctico*. S.l.: s.n.

PRESSMAN, ROGER S., 2002. *Ingeniería de Software, un enfoque práctico. Quinta edición*. S.l.: McGraw-Hill Companies. ISBN ISBN: 8448132149.

PYTHON SOFTWARE FOUNDATION, 2015. Welcome to Python.org. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://www.python.org/.

ROBERTO CURLANGO, 2011. Proceso Unificado de Desarrollo de Software (RUP). [en línea]. Disponible en: http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.

SCRIBID, 2014. Manual del usuario de PostgreSQL. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://es.scribd.com/doc/5703210/Manual-del-usuario-de-PostgreSQL.

SOMMERVILLE, IAN, 2005. *Ingeniería del Software*. S.l.: s.n.

SPARXSYSTEMS, 2014. Herramienta de diseño UML y herramienta CASE UML para desarrollo de software. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.sparxsystems.com.ar/products/ea.html.

TAPSCOTT, ART, 2000. *CAMBIO DE PARADIGMAS EMPRESARIALES*. S.l.: Unidad Académica Santa Cruz. ISBN 958-600-312-4.

TLM, 2014. *Plataforma de Seguridad en las Tecnologías de la información. Módulo para el diagnóstico en las aplicaciones web (PLATSI)*. 2014. S.l.: s.n.

Bibliografía

ALIENVAULT, 2016. OSSIM: The Open Source SIEM | AlienVault. *Detect, analyze, & respond to today's threats* [en línea]. [Consulta: 9 junio 2016]. Disponible en: https://www.alienvault.com/products/ossim.

CARABALLO, 2012. *Prolegómenos Sobre el Lenguaje de Modelado Unificado (UML).* [en línea]. 2012. S.l.: s.n. Disponible en: http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/prolego.pdf.

CARLOS, BLANCO BUENO, 2012. Construcción y Pruebas de Software

CARRILLO P., ISAÍAS; PÉREZ G., RODRIGO; RODRÍGUEZ M., AURELIANO D., 2015. Metodología de Desarrollo del Software. [en línea], Disponible en: http://tecnicaytecnologiasc.wikispaces.com/file/view/Metodologias%20de%20desarrollo.pdf/409623082/Metodologias%20de%20desarrollo.pdf.

CORPORATION, 2016. InterSystems Ensemble | Software de Integración. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.intersystems.com/es/nuestros-productos/ensemble/vision-general/.

DATE, C.J, 1998. *Introducción a los sistemas de bases de datos*. S.l.: Person Educación. ISBN ISBM 968-444-419-290000.

EPF, 2011. OpenUP/Basic. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://epf.eclipse.org/wikis/openupsp/.

ETECSA, 2016. ETECSA Empresa de Telecomunicaciones de Cuba S.A. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.etecsa.cu/.

EUI-FI, 2015. *Introdución a herramientas CASE* [en línea]. 2015. S.l.: s.n. Disponible en: http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

GAMMA, ERICK, 2012. *Patrones de Diseño: elementos de software orientados a objetos reutilizables.* S.l.: s.n.

IBM, 2016. IBM MQ. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www-03.ibm.com/software/products/en/ibm-mq.

JACOBSON, 2004. *El Proceso Unificado de Desarrollo de Software* [en línea]. S.l.: s.n. Disponible en: http://quegrande.org/apuntes/EI/OPT/PAI/teoria/07-08/tema_2_-_el_proceso_unificado.pdf.

JEFF FORCIER, 2016. Welcome to Paramiko. [en línea]. Disponible en: http://paramiko.org.

L. BASS, P. CLEMENTS, R. KAZMAN, 2003. *Software Architecture in Practice 2nd Edition*. S.l.: Addison Wesley.

MICROSOFT, 2016. SQL Server Integration Services. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://msdn.microsoft.com/es-es/library/ms141026%28v=sql.120%29.aspx.

MICROSOFT PATTERNS & PRACTICES, 2013. *Integration Patterns*. US: Microsoft Patterns & Practices. ISBN ISBN-10/ISBN-13.

MILANÉS LÓPEZ, ALDO Y MONTANO GARCÍA, REINIER, 2009. *Sistema para la Gestión de la Tecnología en la Universidad de las Ciencias Informáticas*. La Habana: s.n.

MYSQL, 2015. MySQL. *MySQL* [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.mysql.com/.

ORACLE, 2014. Oracle Fusion Middleware | Oracle ES | Oracle España. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://www.oracle.com/es/middleware/index.html.

OREN BEN-KIKI, 2012. YAML Ain't Markup Language (YAML) version 1.2. [en línea]. Disponible en: http://yaml.org/spec/1.2/spec.html.

PATRICIO, 2014. Introducción a RUP. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://es.scribd.com/doc/51852648/Introduccion-a-RUP.

PHAZ BLINOV, 2016. Welcome to SSHTunnel's documentation. [en línea]. Disponible en: https://sshtunnel.readthedocs.org/en/latest.

PILOTFISH, 2016. Enterprise Integration Platform. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.pilotfishtechnology.com/enterprise-integration-platform/.

P.LETELIER, 2015. *Introducción Proceso de Desarrollo de SW1* [en línea]. S.l.: s.n. Disponible en: https://pid.dsic.upv.es/C1/Material/default.aspx.

PRESSMAN, ROGER, 2001. *Ingenieía de Software: Un Enfoque Práctico*. S.l.: s.n.

PRESSMAN, ROGER S., 2002. *Ingeniería de Software, un enfoque práctico. Quinta edición*. S.l.: McGraw-Hill Companies. ISBN ISBN: 8448132149.

PYTHON SOFTWARE FOUNDATION, 2015. Welcome to Python.org. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://www.python.org/.

ROBERTO CURLANGO, 2011. Proceso Unificado de Desarrollo de Software (RUP). [en línea]. Disponible en: http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.

SCRIBID, 2014. Manual del usuario de PostgreSQL. [en línea]. [Consulta: 20 junio 2016]. Disponible en: https://es.scribd.com/doc/5703210/Manual-del-usuario-de-PostgreSQL.

SOMMERVILLE, IAN, 2005. *Ingeniería del Software*. S.l.: s.n.

SPARXSYSTEMS, 2014. Herramienta de diseño UML y herramienta CASE UML para desarrollo de software. [en línea]. [Consulta: 20 junio 2016]. Disponible en: http://www.sparxsystems.com.ar/products/ea.html.

TAPSCOTT, ART, 2000. *CAMBIO DE PARADIGMAS EMPRESARIALES*. S.l.: Unidad Académica Santa Cruz. ISBN 958-600-312-4.

TLM, 2014. *Plataforma de Seguridad en las Tecnologías de la información. Módulo para el diagnóstico en las aplicaciones web (PLATSI)*. 2014. S.l.: s.n.