**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

**FACULTAD 1**

**CENTRO DE SOFTWARE LIBRE**


# Módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

**Autora**

Mayra de la O Barrientos

**Tutores**

MSc. Yasiel Pérez Villazón

Ing. Hanny Valdés Hernández

**LA HABANA, JUNIO 2019**

**AÑO 61 DE LA REVOLUCIÓN**

**“Podrán morir las personas pero jamás sus ideas”**

**Ernesto Ché Guevara**

## Dedicatoria

Dedico este trabajo a mis padres por darme todo el apoyo que necesité durante los cinco años de la carrera, por todo el amor que me han dado y por confiar en mí y creer que podía llegar hasta aquí.

A mis dos hermanos que los amo, en especial a mi tata Yoandy de la O Barrientos por estar ahí siempre y por apoyarme en todo momento.

## Agradecimientos

Agradezco en primer lugar a mi familia. A mi mamá por ser la mejor mami del mundo, por brindarme su apoyo incondicional, por todo el cariño y confianza y por darme fuerzas para vencer cualquier obstáculo durante estos 5 años. A mi papá que a pesar de la distancia siempre me ha brindado todo su apoyo y cariño, muchas gracias por confiar en mí y creer que podía llegar hasta aquí.

A mi tata Yoandy que lo amo con todo mi corazón, gracias por apoyarme en todo momento y por estar pendiente de todos mis pasos.

A mi cuñada Karol, muchas gracias por cuidar de mi hermano y por todo el cariño que me has dado.

A mis tías Maritza y Mayra que las quiero mucho, gracias por preocuparse siempre por mí.

A mis amigos viejos y nuevos que siempre están a mi lado sin importar la distancia y son la familia que me ha dado la vida.

A mis tutores Yasiel y Hanny por apoyarme en todo lo que estaba a su alcance, por su ayuda incondicional en la creación de este trabajo, por el tiempo dedicado y la experiencia.

A mis amigas Susana y Yailén, por los lindos y malos momentos que hemos pasado, y por ser mis pingüinas.

A mis amistades y personas que he conocido aquí, que a muchas no voy a volver a ver nunca, pero que van a estar en un rinconcito de mi corazón.

A una personita que ha ocupado un lugar muy importante en mi corazón, a mi gran amigo y hermano Reimer Darían Malleza Romero, gracias por formar parte de mi vida, gracias por todos los consejos que me diste, por todos los momentos compartidos, por las tristezas y las alegrías, gracias por ser la persona que eres y por sacarme una sonrisa cuando más triste estaba.

A mi novio Reynier que a pesar de llevar poco tiempo en mi vida se ha convertido en alguien muy especial e importante. Gracias por apoyarme y soportar mis malos días.

En fin a todas esas personas y familiares que de una forma u otra me ayudaron y formaron parte de mi vida en estos 5 años de la carrera.

## Declaración de autoría

Declaro por este medio que yo Mayra de la O Barrientos, con carné de identidad 96022924412 soy la autora principal del trabajo titulado “Módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firman la presente a los __ días del mes de junio del año 2019.

_____

Mayra de la O Barrientos

Autora

_____ _____

MSc. Yasiel Pérez Villazón

Tutor

Ing. Hanny Valdés Hernández

Tutora

## Datos de la autora

**Nombre y apellidos:** Mayra de la O Barrientos

**Correo electrónico:** mdelao@estudiantes.uci.cu

**Institución a la que pertenece:** Universidad de las Ciencias Informáticas.

**Dirección:** Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de la Habana, Cuba, Código Postal 19370.

## Datos de los tutores

**Nombre y apellidos:** Yasiel Pérez Villazón

**Correo electrónico:** yasielpv@uci.cu

**Especialidad de graduación:** Ingeniero en Ciencias Informáticas.

**Institución a la que pertenece:** Universidad de las Ciencias Informáticas.

**Dirección:** Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de la Habana, Cuba, Código Postal 19370.

**Nombre y apellidos:** Hanny Valdés Hernández

**Correo electrónico:** hanny@uci.cu

**Especialidad de graduación:** Ingeniero en Ciencias Informáticas.

**Institución a la que pertenece:** Universidad de las Ciencias Informáticas.

**Dirección:** Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de la Habana, Cuba, Código Postal 19370.

## Resumen

La Distribución Cubana GNU/Linux Nova, cuenta con la plataforma de administración de clientes ligeros Nova-LTSP, que brinda un conjunto de funcionalidades para administrar estos clientes ligeros. Actualmente la herramienta carece de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta, lo que trae como consecuencia vulnerabilidad de la información almacenada en la misma. Debido a esto, el objetivo de este trabajo consiste en desarrollar un módulo que garantice la implementación de mecanismos de bloqueo a conexiones remotas en la plataforma de clientes ligeros Nova-LTSP. La propuesta de solución es guiada por la metodología de desarrollo de software Variación del Proceso Unificado Ágil para la Universidad de las Ciencias Informáticas en su escenario de historias de usuario. Además, se realiza un análisis de las diferentes herramientas, tecnologías, y lenguajes a utilizar para el modelado y desarrollo del módulo. Se realizaron pruebas funcionales, de unidad, integración, aceptación y de seguridad para comprobar el correcto funcionamiento y calidad del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.


**Palabras clave:** bloqueo, fuerza bruta, módulo, Nova-LTSP, seguridad

## Índice

## Índice de tablas

## Índice de figuras

## Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) han tenido un desarrollo acelerado durante las últimas décadas, en este contexto el empleo de sistemas informáticos es clave para cualquier esfera de la sociedad. En el ámbito empresarial su uso se ha incrementado considerablemente, debido en gran medida, al hecho de cada vez contar con un mayor volumen de información a gestionar. La importancia de la información administrada en estos sistemas ha despertado gran interés en toda una gama de intrusos cibernéticos, que se han aprovechado de numerosos problemas de seguridad existentes en las aplicaciones, para llevar a cabo toda una serie de ataques como denegación de servicio, autenticación, entre otros. Es entonces vital disminuir las posibles vulnerabilidades que pudiesen existir en estos sistemas a través de la implementación de medidas y políticas de seguridad.

La seguridad constituye uno de los temas principales en la rama de la informática a nivel mundial. Antes los problemas de seguridad estaban dados por los virus, actualmente son otros los tipos de ataques que preocupan a los usuarios como son: los spamming[1], pharming[2], hacker[3], cracker[4], spyware[5]. Hay que tener en cuenta que la seguridad informática es un proceso dinámico, suele siempre estar encaminado a la actualización permanente de mecanismos, métodos, técnicas y procedimientos que ayudan a contrarrestar los ataques o amenazas informáticas que cada día aparecen en Internet (TARAZONA, 2016).

La seguridad es uno de los mecanismos que necesita una mejora continua, constituye uno de los factores más vulnerables que tienen los sistemas informáticos, que es cada vez más difícil de controlar. Los mecanismos de seguridad son también llamadas herramientas de seguridad y son todos aquellos que permiten la protección de los bienes y servicios informáticos. Dichos mecanismos se clasifican en diferentes tipos de acuerdo con el objetivo principal de los mismos: preventivos, consisten en prevenir la ocurrencia de

[1] ***Spamming:*** *es el hecho de enviar mensajes electrónicos (spam) no solicitados y en cantidades masivas.*

[2] ***Pharming:*** *es la explotación de una vulnerabilidad en el software de los servidores DNS (Domain Name System).*

[3] ***Hacker: p****ersona con grandes conocimientos de informática que se dedica a acceder ilegalmente a sistemas informáticos ajenos y a manipularlos.*

[4] ***Cracker****: el término cracker fue acuñado por primera vez hacia 1985 por hackers que se defendían de la utilización inapropiada por periodistas del término hacker.*

[5] ***Spyware:*** *es software o hardware instalado en una computadora, generalmente sin el conocimiento del usuario, que recoge información de dicho usuario para más tarde enviarla por Internet a un servidor.*

un ataque informático; detectores, tienen como objetivo detectar todo aquello que pueda ser una amenaza para los bienes; correctivos, se encargan de reparar los errores o daños causados una vez que se haya cometido un ataque, modifican el estado del sistema de modo que vuelva a su estado original, y los disuasivos, se encargan de desalentar a los perpetradores de que cometan su ataque para minimizar los daños que puedan tener los bienes (DHAWAN, 2002).

Cuba, en aras de ganar en soberanía tecnológica y seguridad, así como garantizar la informatización de todas las esferas de la sociedad, inició su incursión en el año 2002 en el desarrollo del Proyecto Futuro, nombre dado por el Comandante en Jefe Fidel Castro Ruz a la Universidad de las Ciencias Informáticas (UCI). En la actualidad la Universidad cuenta con un total de 15 centros de desarrollo de software, dentro de ellos se encuentra el Centro de Software Libre (CESOL), cuyo objetivo es el desarrollo de la distribución cubana de GNU/Linux Nova.

La Distribución Cubana GNU/Linux Nova, cuenta con la plataforma de administración de clientes ligeros Nova-LTSP, que presenta módulos para la gestión de los clientes ligeros, imágenes de los sistemas operativos y perfiles de comportamiento de hardware y software, así como la administración del protocolo de configuración dinámica de host (DHCP, del inglés *Dynamic Host Configuration Protocol*) y del sistema. Constituye también una herramienta que permite automatizar la administración de los clientes ligeros, a través de la tecnología *Linux Terminal Server Project* (LTSP) (ALONSO, 2017).

En una entrevista realizada a varios especialistas encargados del despliegue de la herramienta Nova-LTSP, como se aprecia en el Anexo 1, se identificó qué, aunque esta herramienta cuenta con una interfaz de monitoreo para el seguimiento de recursos de hardware, la misma carece de un mecanismo de bloqueo a las conexiones remotas que intentan accesos por fuerza bruta, lo que trae como consecuencia la vulnerabilidad de la información y los servicios que se ejecutan en el servidor.

En correspondencia con lo antes expuesto se plantea como **problema de investigación**: ¿Cómo contribuir a la gestión de mecanismos de bloqueo a las conexiones remotas que intentan accesos por fuerza bruta desde Nova-LTSP?

Se identifica como **objeto de estudio** de la investigación: Mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.

Para darle solución al problema anteriormente planteado, se define como **objetivo general**: Desarrollar un módulo para Nova-LTSP que permita la gestión de mecanismos de bloqueo a las conexiones remotas que intentan accesos por fuerza bruta.

Se define como **campo de acción**: Herramientas que implementan mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.

Para dar cumplimiento al objetivo general se han definido los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación sobre los mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.
2. Diseñar un módulo para Nova-LTSP para la gestión de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.
3. Implementar un módulo para Nova-LTSP para la gestión de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.
4. Evaluar el módulo para Nova-LTSP para la gestión de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.

Se definen como **preguntas científicas**:

1. ¿Cuáles son los presupuestos teóricos que fundamentan el proceso de bloqueo de conexiones remotas que intentan acceso por fuerza bruta en Nova-LTSP?
2. ¿Qué aspectos se deben tener en cuenta para diseñar un módulo para el bloqueo de conexiones remotas que intentan ataques por fuerza bruta en Nova-LTSP?
3. ¿Cuáles son las herramientas más adecuadas para implementar un módulo para el bloqueo de conexiones remotas que intentan accesos por fuerza bruta en Nova-LTSP?
4. ¿Qué métodos, técnicas y pruebas aplicar para la evaluación del módulo para el bloqueo de conexiones remotas que intentan ataques por fuerza bruta en Nova-LTSP?

Con el propósito de dar cumplimiento a los objetivos se trazan las siguientes **tareas de la investigación:**

1. Estudio de los conceptos asociados al marco teórico de la investigación.

2. Caracterización de las herramientas que implementan mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.
3. Definición de la arquitectura de la propuesta de solución.
4. Elaboración de los artefactos requeridos por la metodología de desarrollo seleccionada.
5. Desarrollo de las funcionalidades del módulo para la implementación de mecanismos de bloqueo a las conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP.
6. Validación de la propuesta de solución.

Para facilitar el cumplimiento del objetivo propuesto y de las tareas de investigación se emplean los siguientes métodos científicos:

**Métodos teóricos**

- **Histórico-Lógico:** Es utilizado en el análisis de los sistemas homólogos, de manera que permita buscar elementos que los caractericen y aspectos para fundamentar la propuesta de solución a la problemática planteada.
- **Analítico-sintético:** Con el fin de descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.
- **Inductivo-Deductivo:** se emplea para arribar a razonamientos que puedan ser aplicables al problema a resolver luego de adquirir una serie de elementos referentes a los mecanismos de bloqueo que intentan accesos por fuerza bruta.

**Métodos empíricos**

- **Entrevista:** se le realizó una entrevista al MSc Yasiel Pérez Villazón perteneciente al Centro de Software Libre con el objetivo de conocer las vulnerabilidades presentes en Nova-LTSP.
- **Observación:** se utilizó a través del estudio realizado a las herramientas informáticas que existen, que permiten implementar mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta, para determinar los elementos más comunes que están presentes en las mismas.

- **Encuesta:** se aplicó para evaluar la propuesta de solución desarrollada y conocer el índice de satisfacción de los usuarios potenciales

El presente trabajo de diploma consta con una introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

**Capítulo 1: Fundamentación teórica sobre mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta**

En este capítulo se explican los principales conceptos que se tratan para lograr un mayor entendimiento del tema tratado. Se realiza un estudio sobre los mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta y las herramientas que los implementan. Además, se definen las herramientas y tecnologías que se van a utilizar para el desarrollo del módulo de implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP.

**Capítulo 2: Análisis y diseño del módulo para la implementación de mecanismos de bloqueo a conexiones remotas para Nova-LTSP**

En este capítulo se especifican los requisitos tanto funcionales como no funcionales para lograr que el módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta se integre correctamente a la plataforma Nova-LTSP. También se describen las historias de usuario correspondientes a los requisitos funcionales, la arquitectura y los patrones de diseño que se utilizan durante la implementación.

**Capítulo 3: Implementación y pruebas al módulo de bloqueo a conexiones remotas que intentan accesos por fuerza bruta en Nova-LTSP**

En este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado, así como el diagrama de componentes, y se describen las pruebas a realizar, con el objetivo de comprobar el correcto funcionamiento del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.

# Capítulo 1. Fundamentación teórica sobre mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta

El objetivo de este capítulo consiste en abordar algunos aspectos teóricos que servirán de soporte para la elaboración del módulo para la implementación de mecanismos de bloqueo a las conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP. Se exponen los conceptos fundamentales para lograr una mejor comprensión sobre los términos tratados en la presente investigación. Por último, se presentan tanto la metodología como las herramientas y tecnologías a utilizar en el desarrollo del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.

## 1.1 Conceptos fundamentales

Para lograr un mejor entendimiento del problema a investigar se tuvieron en cuenta conceptos importantes tales como: ataques de fuerza bruta, seguridad informática, amenaza, vulnerabilidad, intrusión.

### 1.1.1 Ataques por fuerza bruta

Es el método de averiguar una contraseña probando todas las combinaciones posibles hasta dar con la correcta. Los ataques por fuerza bruta son una de las técnicas más habituales de robo de contraseñas en Internet dado que no es necesario tener grandes conocimientos en seguridad informática para realizar uno y existen programas que realizan de forma automática todo el trabajo (CRESPO, 2017).

### 1.1.2 Seguridad Informática

La seguridad informática es el conjunto de métodos y herramientas destinados a proteger los bienes o activos informáticos de una institución. La información es el activo más preciado y la seguridad en la información tiene el objetivo de garantizar:

- Confidencialidad: la información o los activos informáticos son accedidos solo por las personas autorizadas para hacerlo.
- Integridad: los activos o la información solo pueden ser modificados por las personas autorizadas y de la forma autorizada.
- Disponibilidad: los activos informáticos son accedidos por las personas autorizadas en el momento requerido (DHAWAN, 2017).

### 1.1.3 Amenaza

Una amenaza es una violación potencial a la seguridad en cómputo. La violación no necesita realmente ocurrir para que haya una amenaza, ya que estas se encuentran presentes en todo momento, aunque no lleguen a ser consumadas. Las amenazas son factores, circunstancias o eventos que pueden causar daños a un sistema de cómputo.

Se puede clasificar como amenaza todo aquello que pretenda e intente destruir o alterar los activos[6] de la organización (MOLINA, 2008).

### 1.1.4 Vulnerabilidad

Una vulnerabilidad es una debilidad del software, hardware o de procedimientos que puede llegar a causar que un atacante pueda abrir una puerta hacia el interior de la red o de los sistemas. A las vulnerabilidades se les conoce comúnmente como hoyo de seguridad. Son el recurso que utilizan los atacantes para poder penetrar dentro de un sistema de cómputo (MOLINA, 2008).

### 1.1.5 Intrusión

Una intrusión es una secuencia activa de eventos relacionados que tratan deliberadamente de causar daños. La definición se refiere a ambas tentativas, las que se consumaron y las que fracasaron (MOLINA, 2008).

## 1.1 Descripción de mecanismos que implementan mecanismos de bloqueo a conexiones remotas

A continuación, se realiza un estudio sobre los mecanismos de prevención y detección de intrusos.

### 1.2.1 Sistema de detección de intrusos (IDS)

En un nivel básico la detección de intrusos en una red representa exactamente el proceso de determinar cuándo algunas personas no autorizadas realizan intentos o logran consumar una intrusión. Un IDS observa todo el tiempo las actividades realizadas en la red y es capaz de decidir cuándo algo le parece sospechoso.

Los IDS se componen básicamente de tres elementos: sensor, consola y motor central.

[6] *Un activo es algo de valor para la organización*.

- Sensor: su labor es detectar eventos de seguridad.
- Consola: se encarga de monitorizar los eventos emitidos y de administrar los sensores.
- Motor central: registra los eventos generados por los sensores en bitácoras y genera las alertas de los eventos de seguridad recibidos.

Según su arquitectura existen dos tipos de IDS: basado en host y basado en red.

- **IDS basado en host**

Los IDS basados en host (HIDS, del inglés *Host-based Intrusion Detection System*) implican utilizar programas instalados en los sistemas que se requiere sean controlados. Dichos programas están diseñados para detectar y responder a la actividad de los usuarios o del sistema, así como ante eventuales ataques.

Los HIDS están orientados a combatir amenazas internas, debido a la capacidad que tienen de controlar y responder a acciones específicas de los usuarios y el acceso a los archivos del sistema. También se encargan de revisar qué aplicaciones del sistema tienen acceso a qué recursos del mismo, así como controlar que no se modifique el estado interno del sistema como es el sistema de archivos, la cantidad de Memoria de Acceso Aleatorio (RAM, del inglés *Random Access Memory*) entre otras cosas

**IDS basado en red**

Los sistemas de detección de intrusos basados en la red operan de una forma diferente que aquellos IDS basados en host. La filosofía de diseño de un IDS basado en la red es escanear los paquetes de red al nivel del enrutador o host, auditar la información de los paquetes y registrar cualquier paquete sospechoso en un archivo de registros especial con información extendida. Basándose en estos paquetes sospechosos, un IDS basado en la red puede escanear su propia base de datos de firmas de ataques a la red y asignarles un nivel de severidad para cada paquete. Si los niveles de severidad son lo suficientemente altos, se enviará un correo electrónico de advertencia a los miembros del equipo de seguridad para que ellos puedan investigar la naturaleza de la anomalía

Los IDS basados en la red se han vuelto muy populares a medida en que la Internet ha crecido en tamaño y tráfico. Los IDS que son capaces de escanear grandes volúmenes de actividad en la red y exitosamente etiquetar transmisiones sospechosas, son bien recibidos dentro de la industria de seguridad (RICHARDS, 2007).

### 1.2.2 Sistema de prevención de intrusos (IPS)

Un sistema de prevención de intrusos (IPS, por sus siglas en inglés), es un dispositivo de seguridad para redes, que se encarga de monitorizar actividades a nivel de red y a nivel de aplicación, con el fin de identificar comportamientos maliciosos, sospechosos e indebidos. De esta manera busca reaccionar ante ellos en tiempo real mediante una acción de contingencia. Un IPS puede tomar decisiones de control de acceso basado, al igual que un *firewall* o un IDS, en monitoreo de direcciones del Protocolo de Internet (IP, del inglés *Internet Protocol*) o puertos; pues también puede hacerlo basándose en contenido de capa de aplicación. Los IPS se pueden clasificar de diferentes maneras: según su método de detección y según la tecnología que los implementan.

**Según su método de detección**

- IPS basado en firmas: para que un IPS realice la detección basada en firmas, se debe contar con una base de datos, en la cual se contengan todos los patrones conocidos de un ataque particular. Esta información se adhiere al dispositivo que realizará la detección para que así, mediante una búsqueda de coincidencias, se pueda establecer si existe o no un ataque.
- IPS basado en anomalías: también conocido como “basado en perfil”. Este tipo de IPS intenta identificar un comportamiento diferente o que se desvíe de lo que, de alguna forma, se ha predefinido como un “comportamiento normal” de la red. Un IPS basado en anomalías es también basado en estadísticas.
- IPS basado en análisis de protocolos: es similar al IPS basado en firmas, pero este realiza inspecciones mucho más profundas en los paquetes y además son flexibles encontrando algunos tipos de ataques.

**Según su tecnología**

- Basado en host: este tipo de IPS monitorea las características de un host en particular, así que detecta actividades dentro del mismo. Entre algunos ejemplos de las características que este dispositivo es capaz de monitorear, se encuentran: tráfico de red cableada o inalámbrica, registros del sistema, procesamiento del equipo, acceso y modificación de archivos, entre otros. Así mismo, las acciones de contingencia, lanzadas por este tipo de IPS, actúan sobre el host en

el cual trabaja. Se enfocan también en la protección de servidores, escritorios o laptop, o una aplicación de servicio.

- Basado en red: con esta tecnología de IPS se realiza monitoreo sobre el tráfico de red que fluye a través de segmentos particulares, y analizan protocolos de red, de transporte y de aplicación para identificar actividades sospechosas. La principal diferencia entre esta tecnología, y la anterior, es la ubicación de sus sensores. Este, los distribuye a lo largo de varios segmentos de red.

**Ventajas que ofrecen los sistemas de prevención de intrusos**

- Protección preventiva antes de que ocurra el ataque.
- Defensa completa (vulnerabilidades del sistema operativo, puertos, tráfico de IP, códigos maliciosos e intrusos).
- Maximiza la seguridad y aumenta la eficiencia en la prevención de intrusiones o ataques a la red de una empresa.
- Fácil instalación, configuración y administración.
- Es escalable y permite la actualización de dispositivos a medida que crece la empresa.
- No requiere tanta dedicación como un IDS tradicional; esto en consecuencia requeriría menos inversión en recursos para administrar y operar estos sistemas (en comparación con un IDS)

**Características de los sistemas de prevención de intrusos**

- Disminución de falsas alarmas de ataques de red.
- Bloqueo automático frente a ataques efectuados en tiempo real.
- Protección de sistemas no parchados.
- Optimización en el rendimiento del tráfico de la red.
- Aplicación de nuevos filtros conforme detecta ataques en progreso (RICHARDS, 2007).

## 1.2 Análisis de las herramientas que implementan mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta

A continuación, se describen algunas herramientas informáticas que implementan mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta.

### 1.3.1 DenyHost

Es una herramienta de seguridad de prevención de intrusiones basada en registros para servidores con protocolos de administración segura (SSH, del inglés *Secure Shell*) escrita en Python. Está pensada para evitar ataques de fuerza bruta en servidores SSH mediante el monitoreo de intentos de inicio de sesión no válidos en el registro de autenticación y el bloqueo de las direcciones IP de origen.

Características:

- Realiza seguimiento de /var/log/secure para encontrar todos los intentos y los filtros de acceso correctos e incorrectos.
- Mantiene la vigilancia en todos los intentos fallidos de inicio de sesión de usuario y host.
- Opcionalmente envía un correo electrónico sobre las notificaciones de nuevos hosts bloqueados y conexiones sospechosas.
- Mantiene todos los intentos fallidos de inicio de usuario válidos o no válidos en archivos separados, por lo que se hace fácil identificar qué usuario válido o no se encuentra bajo ataque ( OROVENGUA, 2013).

### 1.3.2 Snort

Es un sistema de detección y prevención de intrusos Open Source basado en red que utiliza patrones de búsqueda, llevando a cabo registros de paquetes, análisis de protocolo y búsqueda o comparación de contenido en tiempo real. Utiliza reglas descriptivas para determinar qué tráfico debe ser monitorizado y un motor de detección diseñado modularmente para identificar ataques en tiempo real.

Características:

- Distribución libre bajo licencia GNU/GPL[7], gratuito y disponible para Unix/Linux y Windows.
- La escritura de reglas hace uso de un lenguaje propio, además de otros lenguajes de programación.
- Actualización de reglas.
- Puede ejecutarse en múltiples interfaces.

[7] *GNU/GPL: es una licencia que da la libertad de usar, estudiar, compartir (copiar) y modificar el software.*

- Dispone de sistemas de registro y alertas (DHAWAN, 2002).

### 1.3.3 Fail2ban

Es una herramienta (aplicación) de código abierto que permite bloquear a aquellos que intenten vulnerar servicios como SSH, protocolo para transferencia simple de correo (SMTP, del inglés *Simple Mail Transfer Protocol*), protocolo de transferencia de hipertextos (HTTP, del inglés *Hypertext Transfer Protocol*), etcétera, mediante el intento de fuerza bruta. Cuando Fail2ban encuentra reiterados intentos de sesión fallidos desde una misma IP rechaza estos intentos de conexión y los bloquea con reglas de Iptables, es decir, cuando alguien intenta conectarse muchas veces Fail2ban lo bloquea y lo pone en su tabla entonces no deja que vuelva a intentar conectarse.

Características:

- Altamente configurable.
- Soporte para archivos de log de rotación y puede manejar múltiples servicios como (sshd[8], vsftpd[9], apache, etcétera).
- Archivos de registro de monitores y busca patrones conocidos y desconocidos.
- Utiliza Netfilter / iptables y el archivo hosts.deny del protocolo de control de transmisión (TCP, del inglés *Transmission Control Protocol*) Wrapper para rechazar la dirección IP de un atacante por una cantidad fija de tiempo.
- Ejecuta secuencias de comando cuando un determinado patrón ha sido identificado para la misma dirección IP durante más de X veces (EUGENIN, 2015).

### 1.3.4 Ossec

Ossec es un sistema HIDS, es decir, un sistema de detección de intrusos que también opera como un sistema de gestión de incidentes de seguridad (SIM, del inglés *Security Incident Managament*).

[8] *sshd: solid state hybrid drive, unidad híbrida de estado sólido*

[9] *vsftpd: very secure ftp daemon (demonio ftp muy seguro), un pequeño pero eficiente servidor FTP.*

Ossec permite a los clientes implementar un sistema integral de detección de intrusos basado en la supervisión del host con políticas específicas de aplicaciones en el lado servidor. Como software, funciona con la mayoría de los sistemas operativos, incluyendo Linux, OpenBSD, FreeBSD, Mac OS X, Windows.

OSSEC ofrece la posibilidad a través de configuración de definir las adaptaciones específicas en el servidor para la definición de políticas propias más finas. También es capaz de integrarse con las inversiones actuales de clientes como la SIM/SEM (Gestión de incidentes de seguridad y la Gestión de eventos de seguridad) para los informes centralizados y la correlación de eventos (MANCOMÚN, 2017).

Características:

- Comprobación de integridad de archivos
- Detección de rootkits
- Monitoreo de registro
- Respuesta activa

## 1.4 Análisis comparativo de las herramientas informáticas que implementan mecanismos de bloqueo a conexiones remotas

En el siguiente epígrafe se realiza un análisis comparativo de las herramientas informáticas que implementan mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta, estudiadas anteriormente, mediante el método Calificación y Selección de Código Abierto (QSOS, del inglés *Qualification and Selection of Open Source software*). La finalidad de este análisis tiene como objetivo conocer si las aplicaciones analizadas cumplen con las necesidades existentes para el bloqueo de conexiones remotas que intentan accesos por fuerza bruta en Nova-LTSP, expuestas en la problemática de la investigación. A continuación, se muestra la aplicación del método.

**Aplicación del método QSOS**

El método propone cuatro etapas: definición, evaluación, clasificación y selección. Se establece un método de calificación de software para cuantificar y medir las posibilidades reales de implantación del software ofreciendo posibilidad de comparación al establecer criterios ponderados, en base a los cuales calificar el software y hacer una selección final de la manera más objetiva y beneficiosa (RAMOS, 2011).

- Fase de definición: se establece el marco de referencia para la búsqueda de la información relacionada con las necesidades existentes en el proyecto de software a desarrollar. El estudio realizado en la investigación sobre aplicaciones informáticas que implementan mecanismos de bloqueo a conexiones remotas para Nova-LTSP está enmarcado en herramientas basadas en GNU/Linux.
- Fase de evaluación: consiste en realizar una caracterización del software analizado. En el epígrafe 1.3, se describen las herramientas informáticas estudiadas.
- Calificación: consiste en la ponderación de los criterios definidos para realizar la comparación de las herramientas analizadas. En la Tabla 1 se describen los criterios establecidos, en correspondencia con las necesidades del cliente y las características del entorno de desarrollo del proyecto Nova-LTSP.

*Tabla 1: Definición de criterios y su ponderación. Elaboración propia*

| Criterios de análisis | Puntuación | |
|---|---|---|
| | No cubierto<br>0 | Totalmente cubierto<br>1 |
| Soporte para servicio | No presenta soporte para servicios | Presenta soporte para servicios |
| Basado en reglas | No está basado en reglas | Está basado en reglas |
| Basado en red | No está basado en red | Está basado en red |
| Bloqueo de ataque en tiempo real | No bloquea los ataques en tiempo real | Bloquea los ataques en tiempo real |
| Mecanismo de notificación | No presenta mecanismos de notificación | Presenta mecanismos de notificación |

- Fase de selección: se realiza la comparación de diferentes herramientas analizadas mediante los criterios definidos en la fase de calificación. En la Tabla 2 se evidencia la comparación de las herramientas informáticas que implementan mecanismos de bloqueo a conexiones remotas.

*Tabla 2: Tabla comparativa de las herramientas informáticas que implementan mecanismos de bloqueo a conexiones remotas. Elaboración propia.*

| Criterios de análisis | Herramientas informáticas que implementan mecanismos de bloqueo a conexiones remotas | | | |
|---|---|---|---|---|
| | DenyHost | Fail2ban | Snort | Ossec |
| Soporte para servicio | 1 | 1 | 1 | 1 |
| Basado en reglas | 0 | 1 | 0 | 0 |
| Basado en red | 1 | 1 | 1 | 1 |
| Bloqueo de ataque en tiempo real | 0 | 1 | 1 | 0 |
| Mecanismo de notificación | 1 | 1 | 0 | 0 |

Al realizar la comparación mediante QSOS de las tecnologías homólogas existentes que contribuyen a la seguridad de los datos informáticos, se define como solución a emplear Fail2ban.

## 1.5 Funcionamiento de Fail2ban

Fail2ban es un módulo de servidor que se puede utilizar para sistemas Linux y POSIX, escrito en el lenguaje de programación de Python, y que sirve como *framework* de seguridad del tipo de un cortafuegos o filtro de paquetes. Fail2ban detecta aquellas direcciones IP cuyo comportamiento resulta inusual, por ejemplo, las que han intentado acceder varias veces con una contraseña incorrecta a los archivos de registro del servidor. Un cierto número de intentos fallidos, asegurará automáticamente que ese usuario sea "baneado", es decir, que su IP sea bloqueada durante un período determinado de tiempo. El administrador también puede configurar Fail2ban para recibir notificaciones de este tipo de accesos por correo electrónico.

Por defecto, Fail2ban cuenta con una serie de filtros, por ejemplo, para Apache, Postfix o Courier Mail Server, con los que se pueden reconocer cadenas específicas en los archivos de registro. Con la ayuda de estos filtros es posible desencadenar las así llamadas acciones, comandos que se ejecutan en un tiempo predeterminado.

Fail2ban es una solución flexible y eficaz para prevenir acciones de *bots*[10], *scripts* u otro tipo de ataques informáticos a un servidor. Este *framework* hace posible el seguimiento de archivos de registro con patrones sospechosos y permite bloquear o desbloquear sus direcciones IP temporal o indefinidamente. El usuario es libre a la hora de determinar aquellos aspectos que deben ser controlados, así como los parámetros exactos que se aplicarán durante la búsqueda.

### Configuración

Editar el archivo de configuración de Fail2ban para establecer ciertos parámetros, que son muy importantes. Como: tiempo de baneo, máximos intentos, ignorar ciertas IP.

```
# Option: background.
# Notes.: start fail2ban as a daemon. Output is redirect to logfile..
# Values: [true | false] Default: false.
#.
background = true
```

Copiar el archivo de configuración por defecto de fail2ban como jail.local que será ejecutado y nos sobrescribirá las configuraciones que tengamos en jail.conf

```
$ cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.conf
```

Abrir el fichero:

```
$ nano /etc/fail2ban/jail.conf
```

Modificar parámetros SSH:

[10] *bots es un programa informático que efectúa automáticamente tareas repetitivas a través de Internet, cuya realización por parte de una persona sería imposible o muy tediosa*

Una pequeña explicación de los principales parámetros:
*ignoreip*: esta es una lista de IPs separadas por -espacio en blanco- que no serán bloqueadas por fail2ban.
*bantime*: es el tiempo (expresado en segundos) que un host es bloqueado por Fail2ban.
*maxretry*: es el número máximo de intentos de inicios de sesión fallidos antes de que el host se ha bloqueado por Fail2ban.
*filter*: se refiera a la utilización de un filtro apropiado en /etc/fail2ban/filter.d.
*logpath*: el archivo de log que fail2ban utiliza para revisar los intentos de inicios de sesión.

Una vez hayamos terminado de configurar el archivo de fail2ban a nuestro gusto, reiniciamos el servicio.

```
Service fail2ban restart
```

Y comprobamos si el funcionamiento es correcto como solemos hacer con todos los servicios

```
Service fail2ban status
```

Fail2ban viene con una serie de filtros ya predefinidos, pero es posible desarrollar los propios, o ir cambiando las expresiones regulares para adaptarlo a las propias necesidades (ROCHA, 2017).

### 1.6 Metodología de desarrollo de software

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo. La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito, comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado (GABRIEL, 2015).

#### 1.6.1 Metodología de desarrollo Variación de AUP para la UCI

En el desarrollo de la propuesta de solución se utilizará la variación de la metodología de Proceso Unificado Ágil (AUP, del inglés *Agile Unified Process*) para la UCI, una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP y definida por la universidad como guía para la actividad productiva ya permite estandarizar los diferentes productos de trabajo que se generan en sus centros

productivos y se adapta al ciclo de vida definido para los proyectos de la UCI (RODRÍGUEZ, 2014). Esta metodología propone las siguientes fases:

Inicio: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto.

Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura, el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.

Cierre: en el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

El desarrollo de la investigación se centrará en la fase de Ejecución y transitará por las siguientes disciplinas: requisitos, análisis, diseño, implementación, pruebas internas y pruebas de aceptación, definidas en la metodología.

En la elaboración de la herramienta informática propuesta, se emplea el escenario 4 de metodología, para la descripción de los requisitos ya que se aplica a los proyectos que tienen el negocio bien definido, el cliente acompaña al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos, y el proyecto no es extenso.

## 1.7 Tecnologías de desarrollo

A continuación, se describen todas las tecnologías utilizadas en el desarrollo de la propuesta de solución, fueron seleccionadas por políticas del proyecto al que pertenece el módulo que se pretende desarrollar.

### 1.7.1 Lenguaje de modelado

El modelado constituye una simplificación de la realidad donde se define lo esencial para la construcción del software con los objetivos de comunicar la estructura de un sistema complejo, especificar el

comportamiento deseado del sistema, comprender mejor lo que se está desarrollando y descubrir oportunidades de simplificación y reutilización (HERNANDEZ, 2014).

**Lenguaje unificado de modelado**

El Lenguaje Unificado de Modelado (UML, del inglés *Unified Modeling Language*) es un lenguaje de modelado visual que permite especificar, construir y documentar artefactos de un software. Se puede usar en las diferentes etapas del ciclo de vida de un proyecto. Incluye conceptos semánticos, notación y principios generales de un sistema. Contiene además construcciones organizativas para agrupar los modelos en paquetes, lo que permite dividir grandes sistemas en piezas de trabajo más simples. Este lenguaje es lo suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo en una empresa, diseño de la estructura de una organización y el diseño del hardware. Un modelo UML está compuesto por tres clases de bloques de construcción (HERNÁNDEZ, 2010):

- Elementos: los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etcétera).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: son colecciones de elementos con sus relaciones.

### 1.7.2 Herramienta para el modelado de la propuesta de solución

Dentro de las herramientas claves para el desarrollo de aplicaciones informáticas se encuentran las herramientas de Ingeniería de *Software* Asistidas por Computadoras (CASE, del inglés *Computer Aided Software Engineering*). Este tipo de herramienta se utiliza para ayudar a actividades del proceso de *software* tales como la ingeniería de requisitos, el diseño, el desarrollo de aplicaciones y las pruebas.

**Visual Paradigm**

*Visual Paradigm* versión 8.0 es una herramienta de modelado que soporta el UML y proporciona asistencia al equipo de desarrollo, durante todo el ciclo de vida de la elaboración de un software: análisis, diseños orientados a objetos, construcción, pruebas y despliegue. Se integra con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad. Genera código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software. Posibilita la obtención de diferentes informes a partir de la información introducida en la herramienta. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza

la construcción de aplicaciones con calidad y a un menor costo. Posibilita la generación de bases de datos, transformación de diagramas de entidad-relación en tablas de base de datos, así como ingeniería inversa de bases de datos (QUINTANA, 2011).

### 1.7.3 Tecnologías frontend

**HTML**

Lenguaje Marcado de Hipertextos (HTML, del inglés *Hypertext Markup Language*), es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. El lenguaje HTML es muy sencillo y fácil de aprender, este permite el enlace que conecta una página web con otra, ya sea dentro de una página o entre diferentes sitios web. HTML usa "*markup*" o marcado para anotar textos, imágenes, y otros contenidos que se muestran en el Navegador Web. El lenguaje de marcado HTML incluye elementos especiales tales como las etiquetas para el desarrollo del diseño gráfico de la página web HTML 5 es la última versión de HTML y se utiliza para el desarrollo de este sistema, es una versión que contiene todos los elementos de sus versiones anteriores y mejora la compatibilidad entre los navegadores, dispositivos móviles y plataformas (GAUCHAT, 2012).

**CSS**

Hojas de cascada (CSS, del inglés *Cascading Style Sheets*) es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML. Es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Este lenguaje es usado para definir los estilos de los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la tabulación con la que se muestran los elementos de una lista y la separación entre titulares y párrafos para agregar belleza, diseño, orden a las páginas web ya estructuradas con HTML y así hacer más agradable la experiencia de uso de una página web (NAVAJA, 2012).

**JavaScript**

JavaScript es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de *script* para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js, Apache, CouchDB y Adobe Acrobat. Es un lenguaje *script* multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa. JavaScript es la implementación Microsoft de JavaScript para Internet Explorer.

JavaScript se ejecuta en el navegador (cliente) y no requiere ningún software de servidor. Por lo tanto, es un lenguaje de script del lado del cliente. Desde toda la ejecución se lleva a cabo en el navegador, JavaScript es responsable de la mayor parte de la interactividad en una página web: cambio de imagen o el texto, del cambio de color al pasar el ratón, creando senderos ratón. El lenguaje también ha sido ampliamente utilizado para la validación de forma básica. Esto parece lógico, ya que es mejor para validar un formulario en el lado del cliente que hacer varias peticiones al servidor; JavaScript está incrustado comúnmente dentro de la página HTML y es así visible para el visitante. También puede escribirse para ejecutarse en un servidor (PÉREZ, 2008).

### 1.7.4 Tecnologías backend

**Python**

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Es un lenguaje ideal para el desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas de acuerdo a su elegante sintaxis y su tipado dinámico, junto con su naturaleza interpretada.

**Django**

Es un marco de trabajo (*framework)* para el desarrollo web de código abierto. Posee la característica de que es fácil de instalar y puede ser usado tanto para *Windows* como para Linux. Es compatible con varios sistemas gestores de base de datos como son PostgreSQL, Mysql, Sqlite3 y Oracle. Este *framework* promueve el desarrollo rápido permitiendo crear aplicaciones muy rápidas y potentes, sigue el principio DRY (conocido también como una vez y solo una) utiliza esta filosofía para no crear bloques de código iguales y fomentar la reutilización del mismo. Usa una modificación de la arquitectura Modelo Vista-Controlador (MVC), llamada MTV (*Model-Template-View*), que sería Modelo-Plantilla-Vista, esta forma de trabajar permite que sea práctico (MONTERO, 2012).

**Bash**

Bash (*Bourne again shell*) es un programa informático cuya función consiste en interpretar órdenes. Está basado en el *shell* de Unix y es compatible con POSIX. Fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux. Su nombre es un acrónimo de *Bourne-Again Shell* (otro *shell bourne*) el *Bourne shell (*sh), fue uno de los primeros intérpretes importantes

de Unix, la mayoría de los scripts sh pueden ser ejecutados por *Bash* y sin modificación. *Bash* es una de las *shells* más populares de distribuciones de GNU/Linux, la mayoría de los scripts de configuración están programados en este lenguaje (GÓMEZ, 2015).

### 1.7.5 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE, del inglés *Integrated Development Environment*), es un software compuesto por un conjunto de herramientas de programación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (QUIROZ, 2012 ).

**PyCharm**

PyCharm es un entorno de desarrollo integrado para el lenguaje de programación Python que posee un juego completo de herramientas para el desarrollo productivo. Ofrece las siguientes características:

- Compatibilidad con múltiples marcos de desarrollo web como Django y otros, lo que lo convierte en un completo IDE de desarrollo de aplicaciones rápidas.
- Soporte de bases de datos.
- Multiplataforma, utilizado para desarrollar en el lenguaje Python.
- Mantiene el código bajo control de chequeos, asistencia de pruebas, refactorizaciones y un conjunto de inspecciones que posibilitan codificar de forma limpia y sostenible.

**Librería para acceso a fichero**

**Augeas:** Es una herramienta de edición de configuración. Analiza los archivos de configuración en sus formatos nativos y los transforma en un árbol. Los cambios de configuración se realizan al manipular este árbol y guardarlo nuevamente en los archivos de configuración nativos.

Manipula los archivos de configuración de forma segura, más seguro que las técnicas ad-hoc generalmente utilizadas con grep, sed, awk y mecanismos similares en lenguajes de scripting.
Proporciona una API local para la configuración de Linux.
Facilita la integración de nuevos archivos de configuración en el árbol de Augeas.

## 1.8 Conclusiones del capítulo

El estudio de los principales conceptos permitió tener una idea más clara del objeto de estudio y ayudar en una mejor comprensión de los elementos esenciales a tener en cuenta en la posible solución. La selección de la metodología de desarrollo AUP-UCI permitió tener una guía adecuada del proceso de desarrollo de la solución y fueron seleccionadas las herramientas de desarrollo acorde a la propuesta de solución. La entrevista realizada al especialista encargado del despliegue de la herramienta permitió definir los aspectos más importantes a tener en cuenta para realizar el módulo que se propone implementar. El estudio del arte permitió concluir que la herramienta fail2ban es la más adecuada para realizar el bloqueo de las conexiones remotas que intentan accesos por fuerza bruta.

# Capítulo 2. Análisis y diseño del módulo para la implementación de mecanismos de bloqueo a conexiones remotas para Nova-LTSP

En el presente capítulo se aborda sobre el análisis y diseño del módulo para la implementación de mecanismos de bloqueo a las conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP. Se identifican sus características a través de los requisitos funcionales y no funcionales y se describen las historias de usuario que especifican cada requisito funcional. Además, se incluye la arquitectura del software, los patrones de diseño y los diferentes artefactos de ingeniería de software correspondientes a las funcionalidades. A continuación, se expone la propuesta del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP, donde se describen las principales funcionalidades.

## 2.1 Descripción del contexto del negocio de la propuesta de solución

En la descripción de la propuesta de solución se elaboró un modelo conceptual como se aprecia en la *Figura 1*. Este modelo permitió identificar los conceptos existentes en el proceso a informatizar, así como sus atributos y relaciones, lo que permitió conocer las características del entorno en el que se desarrolla la solución.

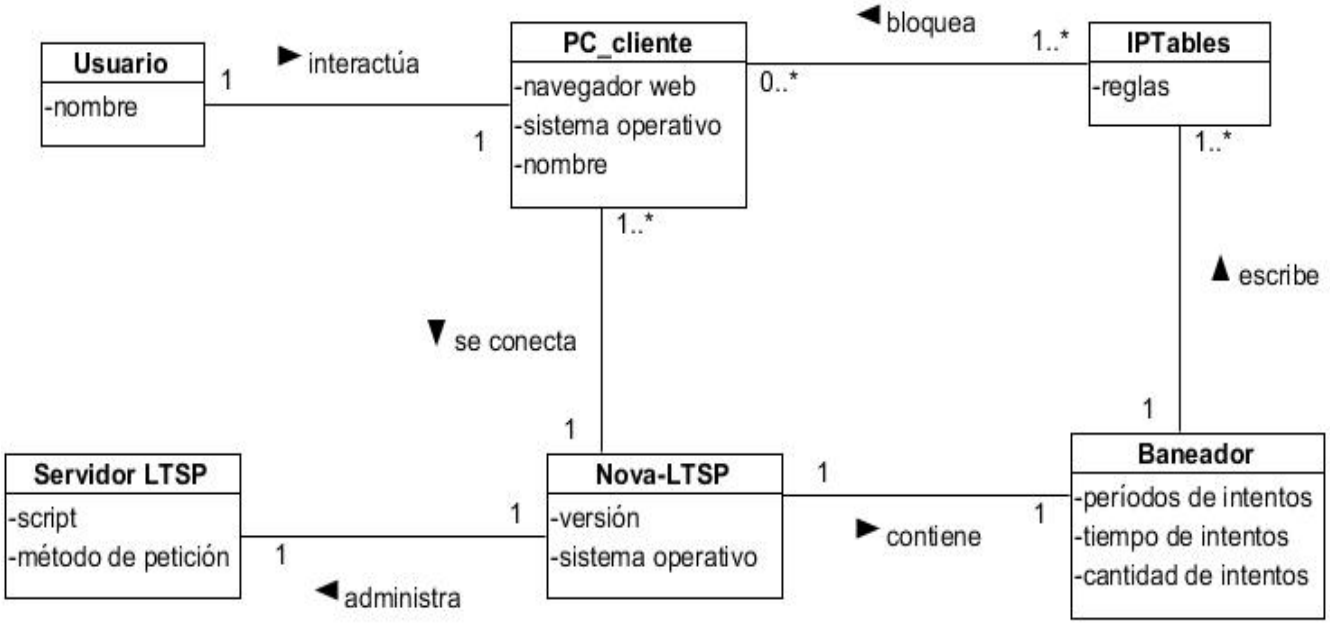



*Figura 1: Modelo conceptual. Elaboración propia*

A continuación, se detallan los diferentes objetos y relaciones que conforman el proceso

**Usuario:** persona que interactúa con una PC_cliente**.**

**PC_cliente:** computadora que se conecta a la plataforma Nova-LTSP vía localizador uniforme de recursos (URL, del inglés *Uniform Resourse Locator*).

**Baneador:** herramienta que se encarga del bloqueo del acceso a los clientes al servidor a partir de reglas.

**Iptables:** es una herramienta de cortafuegos que permite no solamente filtrar paquetes, sino también realizar traducción de direcciones de red (NAT) para IPv4 o mantener registros de log (JIMENEZ, 2015).

**Nova-LTSP**: plataforma web que cuenta con un conjunto de módulos para la administración del servicio de directorio pasivo, LTSP y DNSMASQ y la gestión de imágenes de sistemas operativos, clientes ligeros y usuarios del sistema.

**Servidor LTSP:** representa al servidor LTSP de la plataforma.

## 2.2 Requisitos

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un software, esto ayuda a que exista una mayor comunicación entre el cliente y el equipo de desarrollo. Existen dos tipos de requisitos: los requisitos funcionales y los no funcionales (SOMMERVILLE, 2007). A continuación, se describe cómo se obtuvieron los requisitos de la propuesta de solución, su especificación, descripción y validación.

### 2.2.1 Fuentes para la obtención de requisitos

Las fuentes de obtención de requisitos utilizadas fueron:

- Propuesta de solución del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP (Ver ***Figura 1***).
- Análisis de las herramientas existentes (Ver epígrafe 1.4).
- Especialistas de CESOL.

### 2.2.2 Técnica de identificación de requisitos

Las técnicas de identificación de requisitos de *software* permiten identificar las necesidades de negocio de clientes y usuarios. Son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos de una aplicación, permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle, requieren ser adecuadamente orientadas para cubrir la información que se requiere capturar (PRESSMAN, 2002).

**Entrevista**

Se usó como técnica de extracción de requisitos la entrevista, que es la más tradicional de las técnicas de obtención y consiste en reuniones analista-interesado en las cuales suceden preguntas y respuestas para extraer el dominio de la aplicación. La forma en que se realizó esta técnica fue: el autor de la investigación se entrevistó con el cliente y el jefe del proyecto, a partir de la entrevista realizada se detectaron 9 requisitos funcionales y 7 requisitos no funcionales como se aprecia en el Anexo 1.

### 2.2.3 Especificación de requisitos de software

Una especificación de requerimientos de software (ERS) es un documento que se crea cuando debe especificarse una descripción detallada de todos los aspectos del software que se va a elaborar (PRESSMAN, 2002). A continuación, se describen los requisitos funcionales y no funcionales de la propuesta de solución.

**Requisitos funcionales**

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, describe lo que este debe hacer. En la Tabla 3 se especifican los requisitos funcionales que debe cumplir la propuesta de solución. La complejidad de los requisitos funcionales se obtuvo mediante el producto de trabajo Evaluación de requisitos, propuesto en el expediente de proyecto 5.0 para la actividad productiva de la universidad.

*Tabla 3: Requisitos funcionales del sistema. Elaboración propia.*

| No. | Nombre | Descripción | Prioridad |
|---|---|---|---|
| RF_1 | Registrar intentos de inicio de sesión | El sistema debe permitir registrar todos los intentos de inicio de sesión que se | Alta |

| | | | |
|---|---|---|---|
| | | efectúen a la plataforma de clientes ligeros Nova-LTSP. | |
| RF_2 | Listar intentos de inicio de sesión | El sistema debe permitir mostrar todos los intentos de inicio de sesión que hayan sido registrados previamente en la plataforma de clientes ligeros Nova-LTSP. | Media |
| RF_3 | Buscar intentos de inicio de sesión | El sistema debe permitir buscar uno o más inicios de sesión que hayan sido registrados en la plataforma de clientes ligeros Nova-LTSP. | Baja |
| RF_4 | Mostar estadísticas de los intentos de inicio de sesión de Nova-LTSP | El sistema debe permitir mostrar las estadísticas de intentos de inicio de sesión realizadas a la plataforma de clientes ligeros Nova-LTSP. | Media |
| RF_5 | Mostar estadísticas de los intentos de inicio de sesión mediante SSH | El sistema debe permitir mostrar todas las estadísticas de intentos de inicio de sesión mediante SSH. | Media |
| RF_6 | Mostrar configuración de fail2ban | El sistema debe permitir mostrar la configuración de Fail2ban. | Media |
| RF_7 | Editar configuración de fail2ban | El sistema debe permitir editar la configuración de Fail2ban. | Media |
| RF_8 | Habilitar servicios de fail2ban | El sistema debe permitir habilitar los servicios de Fail2ban. | Alta |
| RF_9 | Deshabilitar servicios de fail2ban | El sistema debe permitir deshabilitar los servicios de fail2ban. | Media |

### Requisitos no funcionales

Los requisitos no funcionales son aquellos requisitos que no describen información a guardar, ni funciones a realizar, sino que son propiedades que hacen al producto usable, rápido o confiable. Además, se conocen

como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software (PRESSMAN, 2002).

*Tabla 4: Listado de los requisitos no funcionales. Elaboración propia.*

| Número | Requisito no funcional | Descripción |
|---|---|---|
| **RNF1** | Restricciones del diseño de implementación | Se utilizan el lenguaje de programación *Python* y el *framework* Django. |
| **RNF2** | Restricciones del diseño de implementación | La interfaz visual debe mantener el estilo de la plataforma de administración de clientes ligeros Nova-LTSP. |
| **RNF3** | Restricciones del diseño de implementación | El sistema cumple con los patrones de diseño experto, creador y Singleton. |
| **RNF4** | Usabilidad | La aplicación debe adaptarse a las estrategias marcarias de la UCI. |
| **RNF5** | Portabilidad | Se puede acceder al sistema desde navegadores web que soporten HTML5, CSS y JavaScript. |
| **RNF6** | Seguridad | La seguridad de los datos se garantiza mediante conexiones seguras empleando el protocolo HTTPS. |

### 2.2.4 Descripción de requisitos de software

La descripción de los requisitos funcionales de la propuesta de solución se realiza mediante las historias de usuario. Este producto de trabajo permite describir las funcionalidades que el sistema debe poseer, sean requisitos funcionales o no funcionales. A continuación, se muestran las historias de usuario correspondientes al RF1 y RF2 que presiden de una prioridad Alta, las restantes historias de usuario se mostrarán en el Anexo 2.

*Tabla 5: Historia de usuario Registrar intentos de inicios de sesión. Elaboración propia*

| **Historia de usuario** | |
|---|---|
| **Número:** HU_1 | **Nombre:** Registrar intentos de inicio de sesión |
| **Prioridad:** Alta | **Iteración Asignada: 1** |
| **Programador:** Mayra de la O Barrientos | **Tiempo Estimado:** 1 semana |
| **Riesgo en Desarrollo:** N/A | **Tiempo Real:** 1 semana |
| **Descripción:** El sistema registra internamente cada uno de los intentos de inicio de sesión que se realicen a la plataforma de clientes ligeros Nova-LTSP. | |
| **Observaciones:**<br>• El usuario debe acceder a la interfaz principal de la plataforma de clientes ligeros Nova-LTSP<br>• La url para acceder a la plataforma está compuesta por:<br>La dirección donde está publicado el servicio, el puerto 1000<br>Ejemplo: http://192.168.56.101:1000/ | |

*Tabla 6: Historia de usuario Listar intentos de inicio de sesión. Elaboración propia.*

| **Historia de usuario** | |
|---|---|
| **Número:** HU_2 | **Nombre:** Listar intentos de inicio de sesión |
| **Prioridad:** Alta | **Iteración Asignada:** 1 |
| **Programador:** Mayra de la O Barrientos | **Tiempo Estimado:** 24 horas |
| **Riesgo en Desarrollo:** N/A | **Tiempo Real:** 20 horas |

**Descripción:** El sistema permite mostrar el listado de todos los intentos de inicio de sesión realizados a la plataforma. Por cada intento de inicio de sesión a la plataforma de clientes ligeros Nova-LTSP, se muestra la dirección IP, el nombre de la PC, el usuario, así como la hora y la fecha en que se realizó. Para ello el administrador debe acceder al listado de los intentos de inicio de sesión realizados.

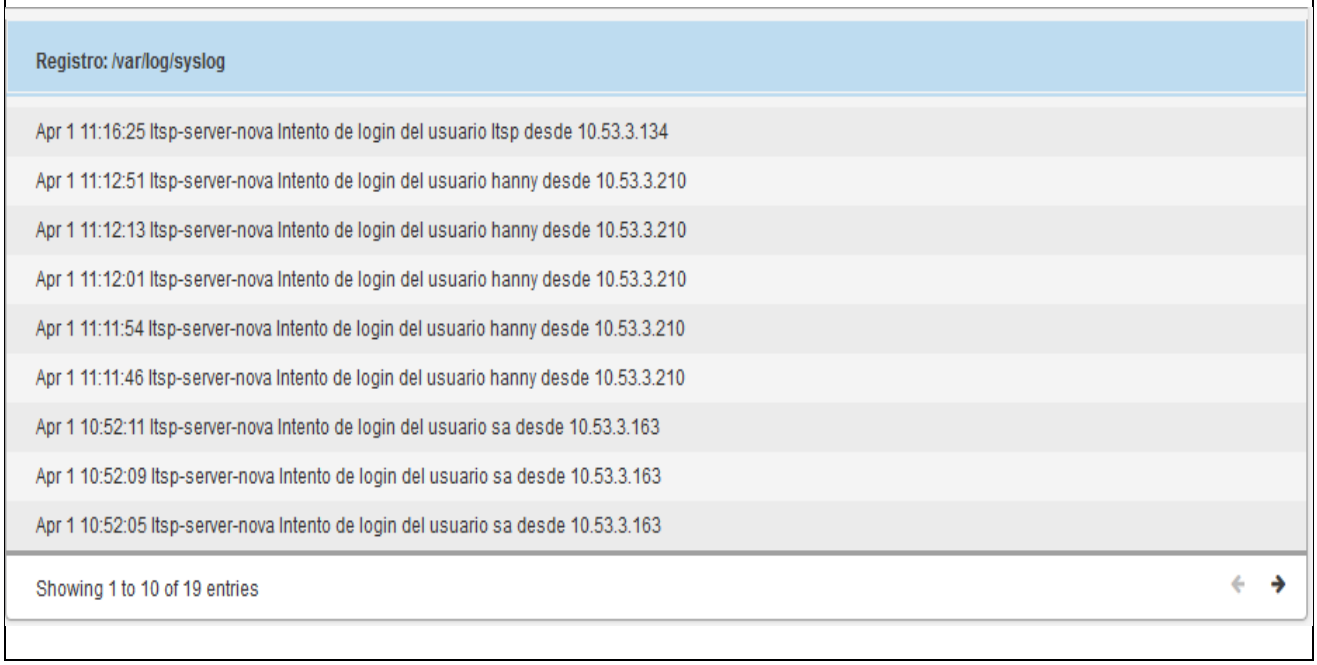
**Observaciones:**

- El usuario debe acceder a la interfaz principal de la plataforma de clientes ligeros Nova-LTSP
- La URL para acceder a la plataforma está compuesta por:

  La dirección donde está publicado el servicio, el puerto 1000

  Ejemplo: http://192.168.56.101:1000/

**Prototipo:**




### 2.2.5 Validación de requisitos de software

La validación de los requerimientos analiza la especificación a fin de garantizar que todos ellos han sido enunciados sin ambigüedades; que se detectaron y corrigieron las inconsistencias, las omisiones y los

errores, y que los productos del trabajo se presentan conforme a los estándares establecidos para el producto. A continuación, se describen las técnicas de validación de requisitos utilizadas en la propuesta de solución:

**Revisión Técnica Formal**

Una Revisión Técnica Formal (RTF) es una actividad del control de calidad del software realizada por ingenieros de software. Los objetivos de una RTF son: descubrir los errores en funcionamiento, lógica o implementación de cualquier representación del software; verificar el cumplimiento de sus requisitos; garantizar su representación de acuerdo a los estándares predefinidos; obtener uniformidad en su desarrollo y hacer proyectos más manejables (PRESMAN, 2002).

La RTF realizada a los diferentes productos de trabajo obtenidos en esta disciplina fue desarrollada por la analista y administradora de la calidad del proyecto de Nova. La misma permitió identificar las deficiencias existentes en la descripción de los requisitos de software.

**Diseño de casos de pruebas**

Los diseños de casos de pruebas permiten crear un conjunto de entradas y salidas esperadas que sean efectivas para descubrir defectos en los programas y muestren que el sistema satisface sus requerimientos. Para su diseño, se selecciona una característica del sistema o componente que se está probando, un conjunto de entradas que ejecutan dicha característica, se documentan las salidas esperadas o rasgos de salida y donde sea posible se diseña una prueba automatizada que demuestre que las salidas reales y las esperadas son las mismas.

## 2.3 Análisis y diseño

La disciplina del análisis tiene como propósito conseguir una comprensión más precisa de los requisitos, refinarlos y estructurarlos; el diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución y que prepara para la implementación y prueba del sistema (PRESSMAN, 2002).

### 2.3.1 Diseño arquitectónico

La arquitectura que se utilizará para el desarrollo del módulo es la empleada por Django, el cual sigue una arquitectura Modelo-Vista-Controlador, solo que hace una adaptación de esta a Modelo-Vista-Plantilla (a

partir de ahora MTV por sus siglas en inglés, *Model Template View*). Por tanto, el módulo propuesto hereda una arquitectura MTV como se aprecia en la Figura 2, debido a que se desarrollará sobre el marco de trabajo Django.

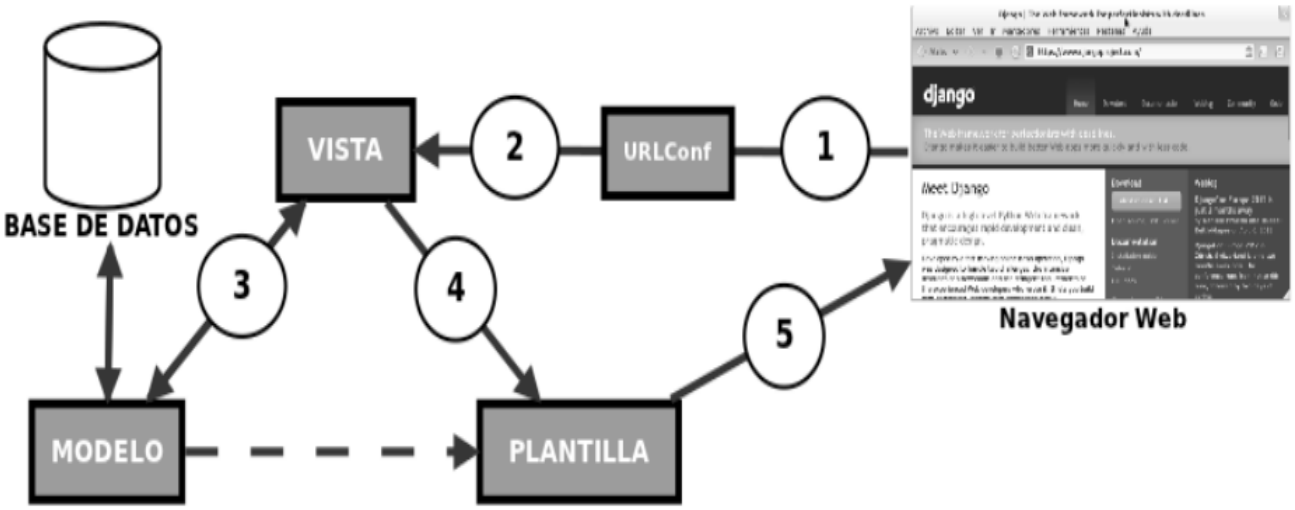



*Figura 2: Funcionamiento del Modelo Vista Plantilla de Django (Montero, 2012)*

**Modelo:** contiene toda la información sobre los datos. Cada una de las entidades de la base de datos se encuentra en el modelo en forma de clases de Python, y sus atributos se almacenan en variables con ciertos parámetros. También estos archivos poseen métodos, lo que permite indicar y controlar el comportamiento de los datos.

**Vista:** es la capa de la lógica de negocios, contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Esta capa sirve de "puente" entre el modelo y la plantilla, se presenta en forma de funciones de Python y su función principal es determinar qué datos serán visualizados en las plantillas.

**Plantilla:** recibe los datos de la vista y luego los organiza para la presentación al navegador web Básicamente es una página HTML con algunas etiquetas extras que son propias del Django.

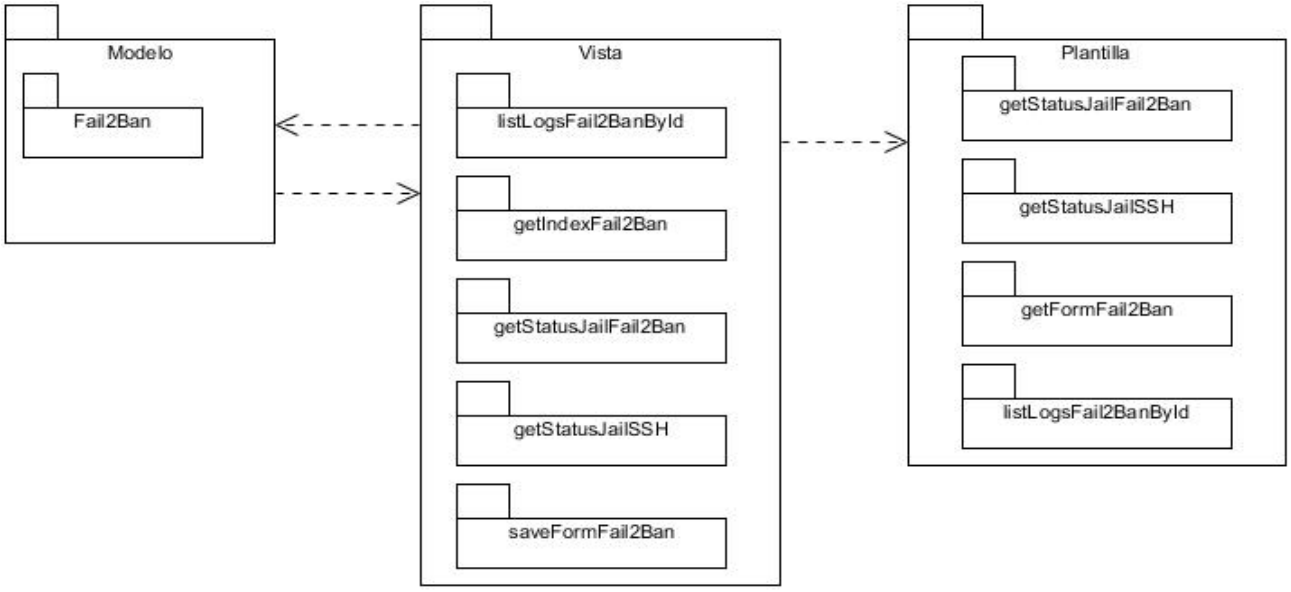



*Figura 3: Diseño arquitectónico de la propuesta de solución. Elaboración propia.*

## 2.4 Modelo de diseño

El modelo de diseño es planteado como un modelo de objetos que describe la realización física de los casos de usos, centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar, constituyendo una entrada principal en la actividad de implementación (LÓPEZ, 2015).

### 2.4.1 Diagrama de clases

Los diagramas de clases (DC) son un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos; las cuales pueden ser asociativas, de herencia, de uso y de contenido. Expresan la estructura u organización del software en términos de clases. Además, constituyen el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer, como para mostrar cómo puede ser construido.

Para el diseño de la propuesta de solución fueron generados un total de 9 diagramas de clases, a continuación, se muestran 2 de estos correspondientes al RF_2 y RF_3 respectivamente, detallados en el sub-epígrafe 2.2.3, en las Figura 4 y Figura 5.

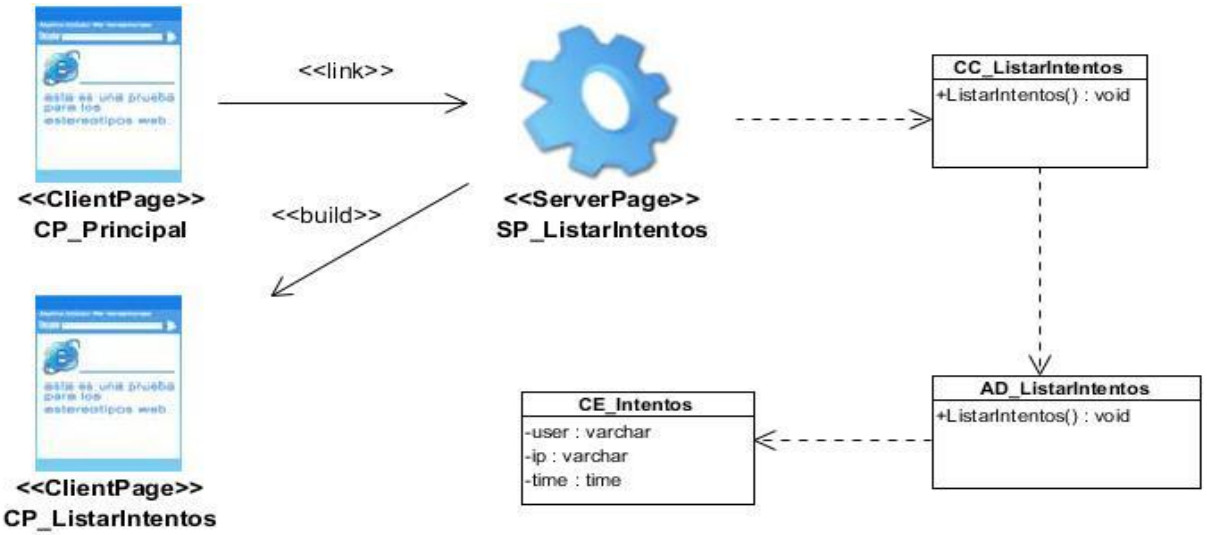



*Figura 4: Diagrama del diseño de la HU Listar  intentos de inicio de sesión. Elaboración propia.*

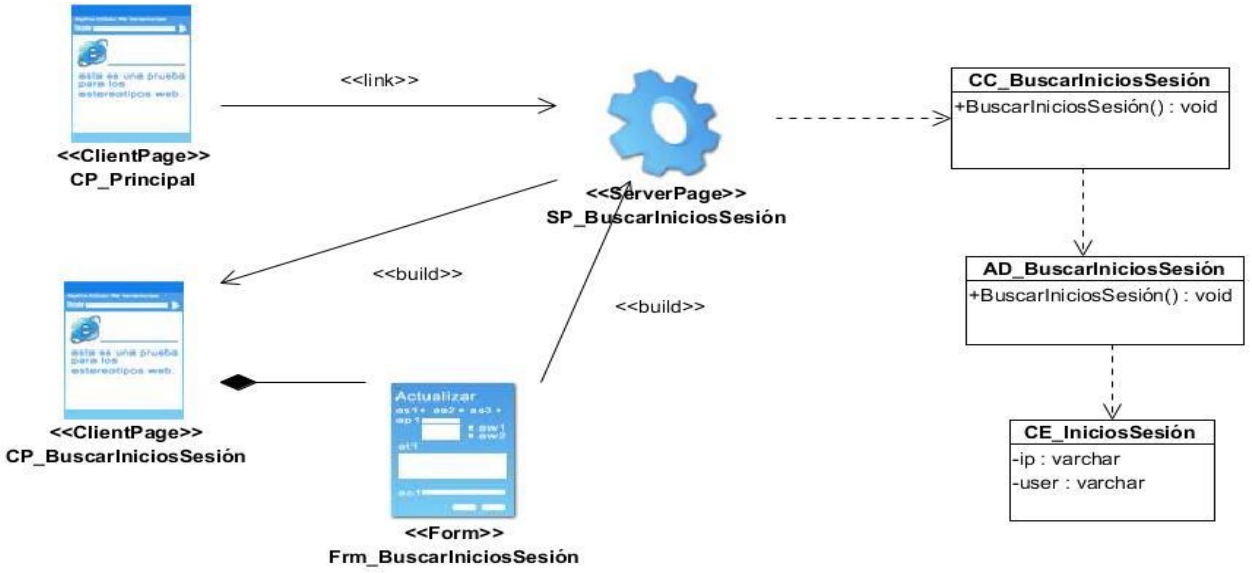



*Figura 5: Diagrama del diseño de la HU Buscar intentos de inicio de sesión. Elaboración propia.*

### 2.4.2 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Contribuyen a reutilizar diseño gráfico identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran

cantidad de situaciones (PRESMAN, 2002). Para la implementación de la propuesta de solución fueron aplicados los siguientes patrones de diseño:

**Patrones GRASP**

General Responsibility Assignment Patterns (GRASP), son una serie de patrones que describen los principios fundamentales de la asignación de responsabilidades a objetos y son considerados una serie de buenas prácticas en el diseño de software (LARMAN, 1999).

**Experto**

Experto es un patrón que suele utilizarse en el diseño orientado a objetos. Este patrón sugiere asignar responsabilidad al objeto que posea la información necesaria para desempeñarla. Con la utilización se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener.

Este patrón se evidencia en la clase fail2Ban que es la que tiene acceso a todas las entidades necesarias y por ello a la información, por lo cual se le asigna la responsabilidad de modelar todos los atributos de configuración como se muestra en la Figura 6

```
class Fai2Ban(models.Model):
```

*Figura 6: Clase experta en información. Elaboración propia.*

**Creador**

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Brinda un soporte a un bajo acoplamiento, lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

```
def getConfigure(request):
    fail2ban=Fail2Ban()
    data = fail2ban.getConfigure()

    return render(request, 'modules/system/fail2ban/edit.html', {
        "data": data
    })
```

*Figura 7: Código fuente de configuración de fail2Ban. Elaboración propia.*

**Patrones GOF**

Gang-of-Four (GoF), también conocidos como la 'pandilla de los cuatro' son patrones de diseño utilizados en situaciones muy frecuentes debido a que se basan en la experiencia acumulada al resolver problemas reiterativos. Además, favorecen la reutilización del código (LARMAN,1999) A continuación, se describe el patrón GoF utilizado en la solución propuesta:

**Singleton**

El patrón de diseño Singleton recibe su nombre debido a que solo se puede tener una única instancia para toda la aplicación de una determinada clase. La intención de este patrón es garantizar que solamente pueda existir una única instancia de una determinada clase y que exista una referencia global en toda la aplicación, se muestra un ejemplo de la aplicación de este patrón en la Figura 8

```
if not cont:
    con = psycopg2.connect(database='novaltsp', user='postgres', passwork='postgres', host='10.0.0.1')
cur = cont.cursor()
cur.execute("SELECT * FROM main_cliente")
rows = cur.fetchall()
```

*Figura 8: Patrón Singleton de la propuesta de solución. Elaboración propia.*

## 2.5 Conclusiones del capítulo

Se lograron extraer mediante la técnica de entrevista 9 requisitos funcionales y 6 no funcionales lo que permitió tener claridad en las características de la propuesta de solución. También se realizaron las descripciones de las historias de usuario, el modelo conceptual, y el diagrama de despliegue como lo propone la metodología propuesta, lo que permitió tener una guía de la situación real de la propuesta de solución. El uso de una arquitectura modelo-vista-plantilla y el empleo de patrones de diseño, garantizan obtener una solución de software con poca dependencia entre clases, flexible al mantenimiento y a la aceptación de cambios.

# Capítulo 3. Implementación y pruebas al módulo de bloqueo a conexiones remotas que intentan accesos por fuerza bruta en Nova-LTSP

En este capítulo se muestran los diferentes artefactos que se utilizan para la implementación y pruebas del sistema, así como los estándares de codificación que debe seguir el equipo de desarrollo para un mejor entendimiento y organización del código. De acuerdo a la metodología que se utiliza, se especifican, de los tipos de pruebas que esta plantea, los que serán empleados para darle validez a los requisitos funcionales y garantizar el óptimo funcionamiento de la aplicación.

## 3.1 Modelo de implementación

Según Pressman el modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se encuentran datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

### 3.1.1 Diagrama de componentes

Los diagramas de componentes son utilizados para estructurar el modelo de la implementación. Permiten modelar una vista estática del sistema, muestran la organización y las dependencias lógicas entre un conjunto de componentes del *software*, que pueden ser librerías, binarios, ejecutables y códigos fuentes.

La Figura 9 muestra el diagrama de componentes correspondiente al módulo de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP.

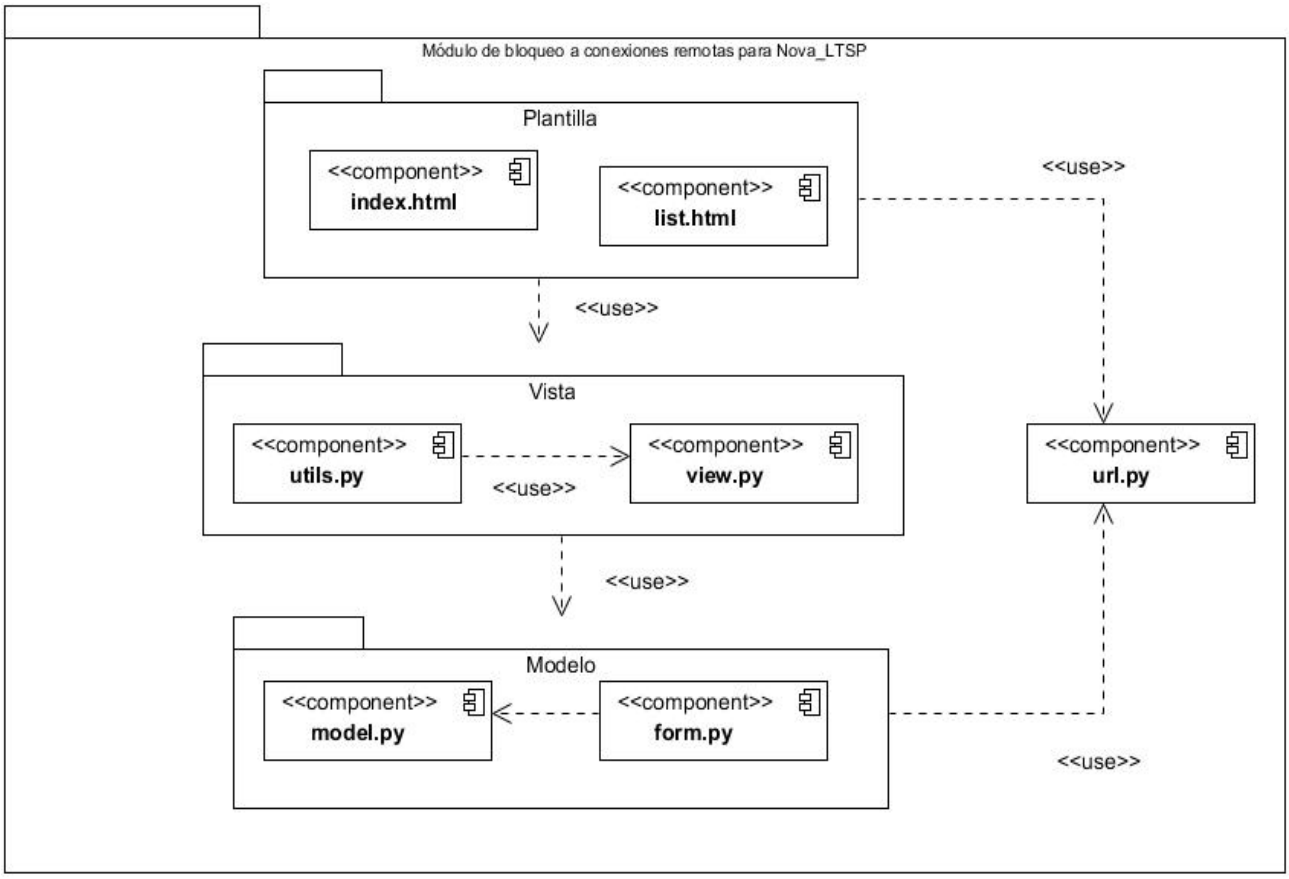



*Figura 9: Diagrama de componentes del módulo de bloqueo a conexiones remotas para Nova-LTSP. Elaboración propia.*

**Descripción de los componentes del sistema**

**Plantilla:** paquete que agrupa a las plantillas encargadas de presentar la información al usuario.

**Vista:** paquete que agrupa a todos los componentes que interactúan con el paquete de clases Modelo; estos componentes permiten trabajar con algunas utilidades sobre los formularios y la renderización de la información en las plantillas apropiadas.

**Modelo:** paquete que agrupa las clases que representan el dominio de entidades de la base de datos y que permiten la interacción directa con el paquete Vista.

**URL:** archivo que contiene todas las URL.

## 3.2 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código el cual refleja un estilo armonioso, como si un único programador hubiese escrito todo el código de una sola vez. Cuando el proyecto de software incorpora código fuente previo, o cuando realiza el mantenimiento de un sistema de software el estándar de codificación debería establecer cómo operar con la base de código existente. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

Para la codificación de la propuesta de solución se utiliza el estándar de codificación para Python basado en la guía de estilo del código Python por Guido Van Rossum y Barry Warsaw. En este documento se listan convenciones utilizadas en el código Python comprendido en la librería estándar de la distribución principal de Python

**Indentación**

- Usa cuatro espacios por cada nivel de codificación.
- Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).
- No se mezclarán tabuladores y espacios en la codificación. El método de indentación más popular en Python es con espacios. El segundo más popular con tabulaciones, sin mezclar unos con otros. Cualquier código indentado con una mezcla de espacios y tabulaciones debe ser convertido a espacios exclusivamente.

**Máxima longitud de las líneas**

- Todas las líneas deben estar limitadas a un máximo de 79 caracteres.
- Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas.
- En cualquier circunstancia se puede utilizar el caracter “\” para cortar líneas largas.

**Líneas en blanco**

- Utilizar la codificación UTF-8.
- Se pueden incluir caracteres que no correspondan a esta codificación utilizando “\x”, “\u”, “\U” para cadenas (strings).

**Importación**

- Las importaciones deben estar en líneas separadas.
- Las importaciones siempre se colocan al comienzo del archivo, simplemente luego de cualquier comentario o documentación del módulo, y antes de globales y constantes.
- Deben quedar agrupadas de la siguiente forma:
    1. Importaciones de la librería estándar.
    2. Importaciones terceras relacionadas.
    3. Importaciones locales de la aplicación/librerías.
- Cada grupo de importaciones debe de estar separado por una línea en blanco.

**Espacios en blanco en expresiones y sentencias**

- Evitar usar espacios en blanco en las siguientes situaciones:
    1. Inmediatamente dentro de paréntesis, corchetes o llaves.
    2. Inmediatamente antes de una coma, un punto y coma o dos puntos.
    3. Inmediatamente antes de un corchete que empieza una indexación.
    4. Más de un espacio alrededor de un operador de asignación u otro para alinearlo con otro.
- Deben rodearse de espacios en blanco los siguientes operadores:
    1. Asignación (“=”).
    2. Asignación de aumentación (“+=”, “-=”, “*=”, “/=”).
    3. Comparación (“==”, “<”, “>”, “>=”, “<=”, ¡ “! =”, “<>”, “in”, “not in”, “is not”).
    4. Expresiones lógicas.

- Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad.
- No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.

**Comentarios**

- Los comentarios deben ser oraciones completas.
- Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula.
- Si un comentario es corto el punto final puede omitirse.
- Los comentarios de una línea para aclaraciones del código aparecerán seguidos de los caracteres "//" en caso de código JavaScript mientras que en Python por el caracter "#" y deben ubicarse en la misma línea que se desea comentar.
- Los comentarios de varias líneas para organización del código aparecerán dentro de los caracteres "/** … **/" en caso de código *JavaScript*, mientras que en *Python* se hará con los caracteres "'…'".

**Convecciones de nombramiento**

- Nunca se deben utilizar como simples caracteres para nombres de variables los caracteres ele minúscula "l", o mayúscula "O", ele mayúscula "L" ya que en algunas fuentes son indistinguibles de los números uno (1) y cero (0).
- Los módulos deben tener un nombre corto y en minúscula.
- Los nombres de clases deben utilizar la convención "CapWords" (palabras que comienzan con mayúscula).
- Los nombres de las excepciones deben estar escrito también en la convención "CapWords" utilizando el sufijo "Error".
- Los nombres de las funciones deben estar escritos en minúscula separando las palabras con un guion bajo (_).
- Las constantes deben quedar escritas con letras mayúsculas separando las palabras con un guion bajo (_).

La Figura 10 representa un ejemplo donde se muestra la aplicación de algunos estándares de codificación.


*Figura 10: Aplicación de los estándares de codificación. Elaboración propia*.

Luego de realizar el modelado de implementación y definido los estándares de codificación utilizados para comprender todos los aspectos a tener en cuenta en la generación de código se define la estrategia para lograr una mejor organización y se realizan las pruebas de software.

## 3.3 Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia, el desempeño de un software, involucra las operaciones del sistema bajo condiciones controladas y evalúa los resultados. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad (PRESSMAN, 2002).

Una vez que se implementa el código del sistema es necesario probarlo para encontrar y enmendar la mayor cantidad de errores antes de entregarlo al cliente. El objetivo de este proceso es diseñar casos de pruebas que tengan una alta probabilidad de encontrar errores. Para alcanzar este objetivo existen técnicas de pruebas de software, que contienen directrices sistemáticas para comprobar la lógica interna y de interfaces de los componentes del sistema, así como los dominios de entrada y salida para identificar errores funcionales y de desempeño. A continuación, se describen los tipos de pruebas de software aplicadas, así como los métodos y técnicas empleadas para la evaluación del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP. En el caso de las pruebas de aceptación, se llevan a cabo para verificar que el software está listo y que puede ser utilizado por usuarios finales, para ejecutar las tareas y funciones para las que fue construido.

## 3.4 Aplicación de las pruebas de software

En el epígrafe se describen las pruebas de software realizadas en las disciplinas de Pruebas internas y Pruebas de aceptación propuestas en la metodología de desarrollo de software AUP-UCI.

### 3.4.1 Pruebas unitarias

Las pruebas unitarias se centran en probar cada componente de código de un software de forma individual para asegurar que funcione de manera apropiada como unidad. Emplean técnicas de prueba que recorren caminos específicos en la estructura de control de los componentes (pruebas estructurales) (RODRÍGUEZ, 2014). El método de prueba utilizado para la realización de esta prueba es Caja blanca y la técnica de prueba contenida en este método que se empleó fue la Técnica del camino básico.

**Método de prueba: Caja blanca**

El método de Caja blanca se enfoca en probar el sistema teniendo en cuenta la estructura interna del mismo. Verifica la correcta implementación de las unidades internas, las estructuras y sus relaciones y hacen énfasis en la reducción de errores internos

**Técnica de prueba: Camino básico**

La técnica del Camino básico permite obtener una medida de la complejidad lógica de la codificación de software y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución independiente en un componente o programa. Un camino o ruta es una vía por la cual procede la ejecución a través de una función desde su inicio hasta el fin (PRESSMAN, 2002).

```
def pullData(key,data):
    arr=data.split('\n')//1
    for item in arr://2
        if key in item: //3
            re=item.split(":")//4
            return re[1].strip()//5
```

*Figura 11: Código de implementación para el método del camino básico. Elaboración propia*

A continuación, se describen los pasos que se realizaron para desarrollar la técnica del camino básico:

1. **Confeccionar el grafo de flujo**: usando el código de la figura 11 se realizó la representación del grafo de flujo, el cual está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo
- Regiones: son las áreas delimitadas por aristas y nodos

A continuación, la figura 12 muestra el grafo de flujo obtenido:


*Figura 12: Grafo resultante del método caja blanca camino básico. Elaboración propia*

1. **Calcular la complejidad ciclomática:** proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, la complejidad ciclomática se calcula de tres formas distintas. En la solución propuesta se aplican las tres variantes con el propósito de triangular los resultados obtenidos, validando que la cantidad de caminos a definir es la correcta:

**Variante 1:**

V (G)= A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo

V (G) = A-N+2 = 4-5+2=1

**Variante 2:**

Teniendo en cuenta el número de regiones que presenta el grafo de flujo obtenido, se definen un total de 1 región.

**Variante 3:**

V(G) = Nodos de predicado + 1, donde los nodos predicados son los que tienen como salida más de una arista.

V(G) = Nodos de predicado + 1

V(G) = 0 + 1 = 1

Las tres variantes aplicadas brindan como resultado que el valor de la complejidad ciclomática del método pullData(key, data) es igual a 1.

1. **Determinar un conjunto básico de caminos linealmente independientes:** una vez aplicadas las tres variantes para calcular la complejidad ciclomática del método pullData (key, data), se obtiene

que el número de caminos linealmente independientes de la estructura de control del método es 1, definiéndose el siguiente camino:

**Camino básico #1**: 1 -2-3-4-5

2. **Determinar un conjunto básico de caminos linealmente independientes:** Obtención de casos de prueba: cada camino independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. En este caso se obtuvo 1 camino básico, por tanto, se hace necesario la confección de igual número de CP, para aplicar las pruebas a este método. A continuación, en la tabla 12 se muestra el CP diseñado para el camino básico obtenido.

*Tabla 6: Resultados de las pruebas por casos de prueba. Elaboración propia*

| Diseño de caso de prueba | |
|---|---|
| **Descripción** | Lista la cantidad de intentos de inicios de sesión que se hayan realizado a la plataforma de clientes ligeros Nova-LTSP. |
| **Condición de ejecución** | Una vez el usuario acceda a la plataforma e intente registrase, la herramienta guarda el nombre y la dirección IP de la pc de donde se accede |
| **Entrada** | Nombre de usuario y contraseña |
| **Resultado esperado** | Listar cada uno de los intentos de inicios de sesión realizados a la plataforma. |

### 3.4.2 Pruebas funcionales

Las pruebas funcionales son aquellas que se llevan a cabo sobre la interfaz del software sin prestar atención al código, por lo que los casos de prueba son creados con el objetivo de demostrar que la entrada es aceptada de forma adecuada y que se produce una salida correcta. El diseño de esta prueba se realiza con la intención de detectar funciones incorrectas o ausentes, errores en accesos a bases de datos externas, errores de interfaz, errores de rendimiento, y errores de inicialización y de terminación(PRESSMAN, 2002).

El método de prueba utilizado para la realización de esta prueba es Caja negra y la técnica de prueba contenida en este método que se empleó fue la de Partición de equivalencia.

**Método de prueba: Caja negra**

Se enfoca en probar el módulo sin tomar en cuenta la estructura interna del mismo, su objetivo es validar que las salidas sean las esperadas. Se centra en encontrar las circunstancias en las que el sistema no se comporta conforme a las especificaciones establecidas.

**Técnica de prueba: Partición de equivalencia**

La técnica divide el dominio de entrada de un programa en clases de datos, a partir de las cuales pueden derivarse casos de prueba. Además, descubre clases de errores, que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general. Mediante su empleo se puede reducir al máximo el total de casos de prueba que deben desarrollarse

**Diseños de casos de pruebas**

Es una parte de las pruebas de componentes y sistemas en las se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. Su objetivo es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para su diseño, se selecciona una característica del sistema o componente que se está probando, un conjunto de entradas que ejecutan dicha característica, se documentan las salidas esperadas o rasgos de salida y donde sea posible se diseña una prueba automatizada que demuestre que las salidas reales y las esperadas son las mismas. A continuación, se presenta el diseño de caso de prueba para el RF 1. Registrar intentos de inicios de sesión, Tabla 7.

*Tabla 7: Caso de prueba Funcional del RF 1 Registrar intentos de inicios de sesión. Elaboración propia*

| Escenario | Descripción | Variable 1 | Variable 2 | Variable 3 | Variable 4 | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|---|---|
| EC 1.1 Registrar intentos de | El sistema permite registrar todos los | V | V | V | V | El usuario accede sin problemas a la | 1. El usuario accede a la plataforma |
| | | Usuario | Igual | Mayra | Usuario Mayra | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| inicios de sesión. | intentos de inicio de sesión que se realicen a la plataforma de clientes ligeros Nova-LTSP, en caso de que no exista ningún dato incorrecto. | | | | | plataforma de clientes ligeros | de clientes ligeros<br>2. El usuario rellena los campos de usuario y contraseña<br>3. El usuario da clic en entrar. |
| EC 1.2 Error cuando introduce datos incorrectos | El sistema permite verificar si existen datos incorrectos y permite que se registre el usuario. | V | I | V | I | Muestra un error si los datos son incorrectos | 1. El usuario accede a la plataforma de clientes ligeros<br>2. El usuario no rellena los campos de usuario y contraseña<br>3. El usuario da clic en entrar. |
| EC 1.3 Error cuando introduce campos vacíos | El sistema permite verificar si existen datos vacíos y permite que | V | I | V | I | Muestra el mensaje *"Existen campos obligatorios vacíos "* | 1. El usuario accede a la plataforma de clientes ligeros |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | se registre el usuario | | | | | | 2. El usuario no rellena los campos de usuario y contraseña<br>El usuario da clic en entrar. |

Con el objetivo de comprobar que las funcionalidades del sistema fueron implementadas correctamente y responden a las necesidades del cliente, aplicando los diseños de casos de prueba antes descritos, se realizaron pruebas funcionales. Las pruebas se realizaron en dos iteraciones. En la primera se detectaron un total de 20 No Conformidades (NC), clasificadas en 6 de ortografía, 12 de redacción y 2 de validación, al finalizar la iteración todas las NC quedaron resueltas. En la segunda iteración los resultados fueron satisfactorios, obteniéndose cero NC. Figura 13 ilustra los resultados de aplicar el método de caja negra, teniendo en cuenta los tipos de NC identificadas (ortografía, redacción y validación).

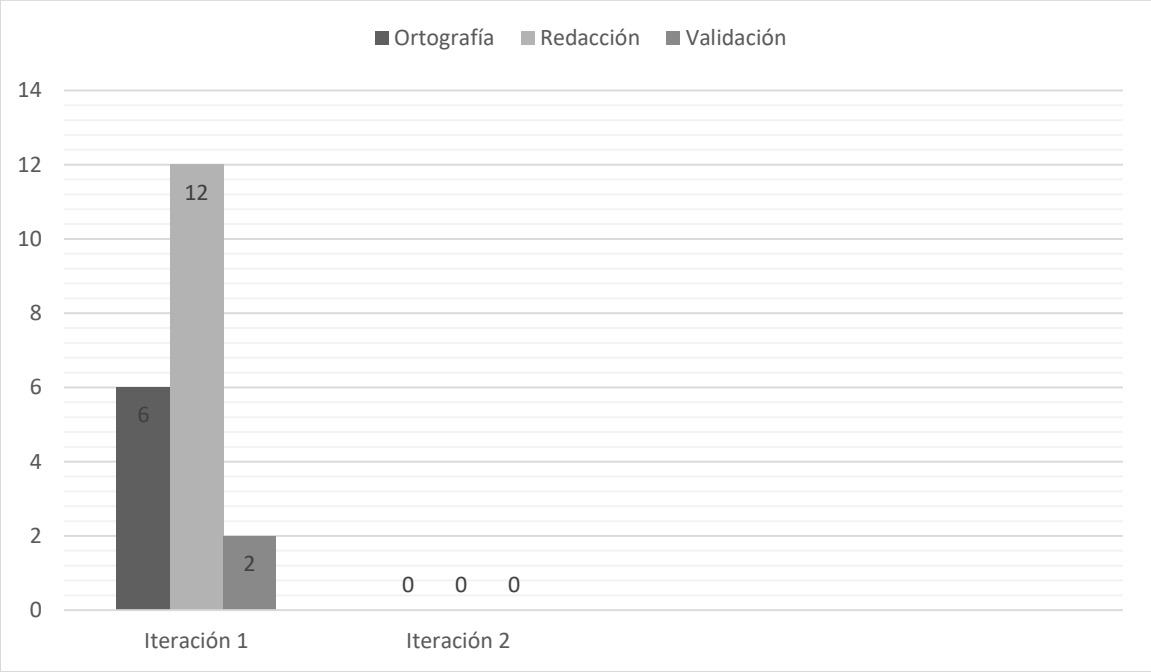


*Figura 13: No Conformidades detectadas al aplicar el método de caja negra. Elaboración propia.*

### 3.4.3 Pruebas de integración

La prueba de integración es una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera independiente y construir una estructura de programa que determine el diseño(PRESSMAN, 2002). Es una forma de verificar la correcta interrelación de los distintos componentes del sistema, en el caso de la solución desarrollada es la verificación de una correcta interoperabilidad entre el módulo desarrollado y Nova-LTSP.

Partiendo que el módulo debe integrarse a una plataforma base, se emplea una estrategia de integración ascendente, donde los componentes se integran de abajo hacia arriba. En esta prueba de integración ascendente se realiza la prueba de regresión que permite ejecutar nuevamente el mismo subconjunto de pruebas que ya se ha aplicado para asegurar que los cambios no han propagado efectos colaterales indeseables.

**Resultados de la prueba de integración**

Las pruebas de integración realizadas permitieron identificar una URL duplicada en diferentes módulos de la plataforma Nova-LTSP. Luego de solucionar esta no conformidad, el módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP se integró correctamente con dicha plataforma

### 3.4.4 Pruebas de seguridad

Las pruebas de seguridad garantizan que los usuarios estén restringidos a funciones específicas o que su acceso esté limitado únicamente a los datos que están autorizados a acceder. Sólo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funcionalidades disponibles. El objetivo fundamental de este tipo de pruebas es comprobar los niveles de seguridad lógica del sistema.

**Resultados de las pruebas de seguridad**

Para evaluar la seguridad del módulo se realizó un caso de prueba, donde se escogieron 6 PC cada una con diferente IP, 3 de estas PC conocían la clave de acceso a la plataforma de clientes ligeros Nova-LTSP y las otras 3 no. Se definió 5 intentos como máximo, 60 min como tiempo de bloqueo y 1 min para el tiempo entre los intentos de inicios de sesión. Las 3 PC que conocían la clave de acceso pudieron acceder sin

problemas a la plataforma y las otras 3 después de 5 intentos fallidos fue bloqueado el IP de cada una por lo que se escribieron reglas Iptables en el firewall para cada una.

### 3.4.5 Pruebas de aceptación

Esta es la etapa final en el proceso de pruebas, antes de que el sistema sea aceptado para uso operacional. El sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados. Las pruebas de aceptación revelan los errores y las omisiones en la definición de requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba (SOMMERVILLE, 2007).

*Tabla 8: Caso de prueba de aceptación para la historia de usuario "Listar intentos de inicios de sesión". Elaboración propia.*

| **Caso de prueba de aceptación** |
|---|
| **Nombre de la historia de usuario:** Listar intentos de inicio de sesión |
| **Nombre de la persona que realiza la prueba:** Yasiel Pérez Villazón |
| **Descripción de la prueba:** El sistema permite listar todos los intentos de inicios de sesión realizados a la plataforma de clientes ligeros Nova-LTSP |
| **Condiciones de ejecución:** El usuario debe autenticarse en la plataforma de clientes ligeros Nova-LTSP |
| **Entrada/Pasos de ejecución:**<br>1. Acceder a la plataforma Nova-LTSP<br>2. Autenticarse en la plataforma Nova-LTSP |
| **Resultado esperado:** El sistema guarda en una lista al usuario autenticado, donde muestra la hora, fecha y dirección IP |
| **Evaluación de la prueba:** Satisfactoria |

Al terminar las pruebas de aceptación se obtuvo un total de 6 No Conformidades en la primera iteración, de las cuales 4 fueron resueltas y 2 no procedían. En la segunda iteración se obtuvieron 5 no conformidades donde 4 fueron resueltas y 1 no procedía; y en la tercera iteración se encontraron 3 no conformidades y todas fueron resueltas. Durante las 3 iteraciones no quedó ninguna no conformidad por resolver. Los resultados correspondientes a estas pruebas se muestran en la Figura 14.

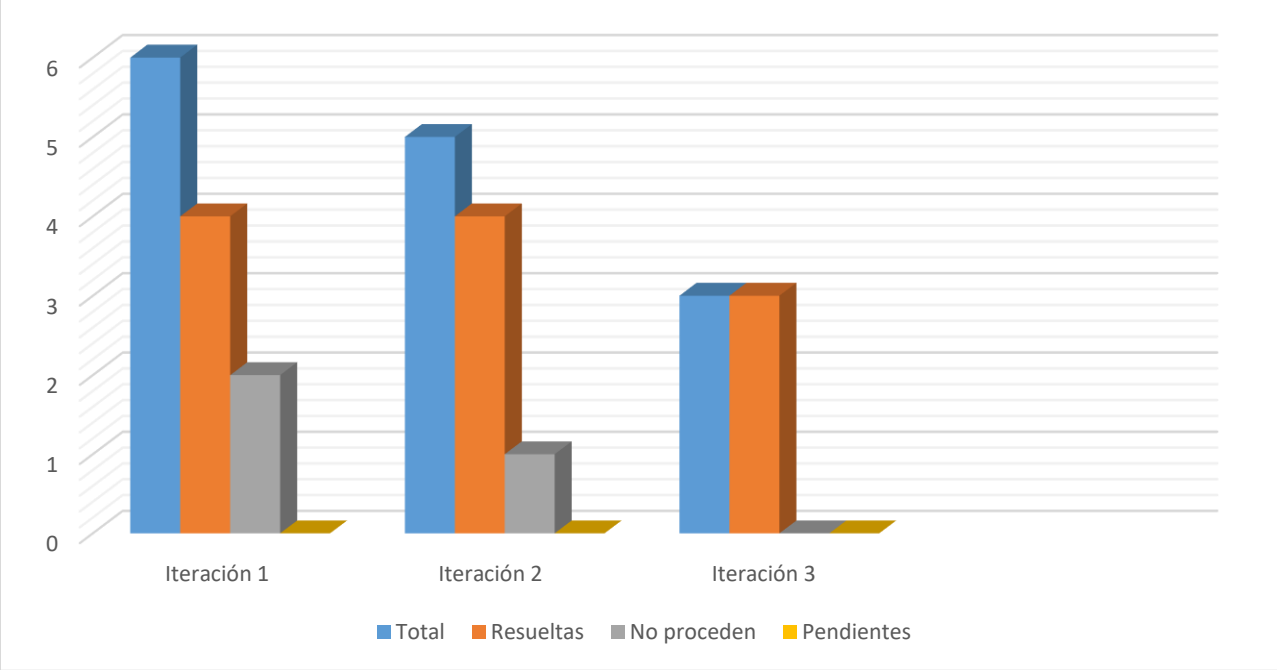



*Figura 14: Resultados de las pruebas de aceptación. Elaboración propia.*

### 3.7 Evaluación del objetivo general de la investigación

La técnica de *IADOV* se compone de cinco preguntas claves: tres cerradas y dos abiertas, es utilizada para determinar el nivel de satisfacción individual y grupal de los usuarios a partir de una encuesta elaborada según las exigencias pertinentes. La aplicación de esta técnica constituye una vía indirecta para el estudio de satisfacción, ya que los criterios que se utilizan se fundamentan en las relaciones que se establecen entre las tres preguntas cerradas, que se intercalan dentro de un cuestionario que se muestra en el Anexo 4 y cuya relación el encuestado desconoce.

La encuesta elaborada para evaluar el índice de satisfacción de los usuarios potenciales de la propuesta de solución fue aplicada a 7 especialistas del proyecto Nova. Las preguntas 2,3 y 4 de la encuesta se relacionan a través de lo que se denomina el "Cuadro Lógico de IADOV" que se muestra en la siguiente tabla.

*Tabla 9: Cuadro lógico de IADOV. Elaboración propia.*

| 1. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la solución desarrollada. | 2. ¿Considera usted correcta la forma en que se realiza el bloqueo de conexiones remotas a la plataforma Nova-LTSP? | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **No** | | | **No sé** | | | **Sí** | | |
| | 1. ¿Considera usted factible la implementación de un módulo que permita el bloqueo de las conexiones remotas que intentan accesos por fuerza bruta a la plataforma Nova-LTSP? | | | | | | | | |
| | **Sí** | **No sé** | **No** | **Sí** | **No sé** | **No** | **Sí** | **No sé** | **No** |
| Me gusta mucho | 1 | 2 | 6 | 2 | 6 | 6 | 6 | 6 | 6 |
| Me gusta más de lo que me disgusta | 2 | 2 | 3 | 2 | 3 | 3 | 6 | 3 | 3 |
| Me da lo mismo | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Me disgusta más de lo que me gusta | 6 | 3 | 6 | 3 | 4 | 4 | 3 | 3 | 4 |
| No me gusta nada | 6 | 6 | 6 | 6 | 4 | 4 | 6 | 4 | 5 |
| No sé qué decir | 2 | 3 | 6 | 3 | 3 | 3 | 6 | 3 | 4 |

La forma de utilizar la tabla es la siguiente, cada encuestado recibe una evaluación individual en dependencia de las respuestas que dé a las preguntas cerradas. Para facilitar el procesamiento posterior, en el diseño de la encuesta se debe tener en cuenta que a estas preguntas solo se responda de la forma prevista en el cuadro lógico de IADOV. Las respuestas a las preguntas 2 y 3 pueden ser Sí, No, No sé, y a las preguntas 4, "Me gusta mucho", "Me gusta más de lo que me disgusta", "Me da lo mismo", "Me disgusta más de lo que me gusta", "No me gusta nada", o "No sé qué decir".

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y – 1. El número resultante de la interrelación de las tres preguntas que indica la posición de cada encuestado en la siguiente escala de satisfacción:

1. Clara satisfacción +1

2. Más satisfecho que insatisfecho 0.5

3. No definido y contradictorio 0

4. Más insatisfecho que satisfecho -0.5

5. Clara insatisfacción -1

El índice de satisfacción grupal (ISG) se expresa en una escala numérica que va desde 1 (máxima satisfacción), hasta -1 (máxima insatisfacción). El ISG se calcula mediante la siguiente fórmula:

$$ISG = \frac{A(+1) + B(+0.5) + C(0) + D(-0.5) + E(-1)}{N}$$

En esta fórmula A, B, C, D, E, representan la cantidad de encuestados colocados respectivamente en las posiciones de satisfacción 1; 2; 3 u 6; 4; 5 y donde N representa la cantidad total de encuestados

**Resultados obtenidos**

Los resultados obtenidos de la aplicación de la encuesta se presentan en la Tabla 10.

*Tabla 10: Resultados obtenidos de los encuestados. Elaboración propia.*

| Categorías grupales de satisfacción | N = 7 | Escala |
|---|---|---|
| Clara satisfacción | 4 | A |
| Más satisfecho que insatisfecho | 2 | B |
| No definido | 0 | C |
| Más insatisfecho que satisfecho | 0 | D |
| Clara insatisfacción | 0 | E |
| Contradictorio | 1 | C |

2. Cálculo del ISG

ISG=A (+1) +B (+0.5) + C (0) /N

ISG= (4(+1) +2(+0.5)) +1(0) /7=0.71

3. Interpretación del resultado del ISG

El proceso de evaluación del objetivo de la investigación mediante la técnica de IADOV confirmó su factibilidad de uso, expresando cuantitativamente en el alto ISG (0.7) y cualitativamente en los criterios emitidos en el centro de desarrollo CESOL, lo que refleja la aceptación de la propuesta, como se muestra en el Anexo 3 y el reconocimiento a su utilidad.

### 3.8 Interfaz principal

Una vez finalizado el desarrollo del software es posible visualizar la pantalla principal del módulo de bloqueo a conexiones remotas que intentan accesos por fuerza bruta en Nova-LTSP, donde se observa el resultado obtenido durante la implementación de las historias de usuario descritas en el capítulo anterior.

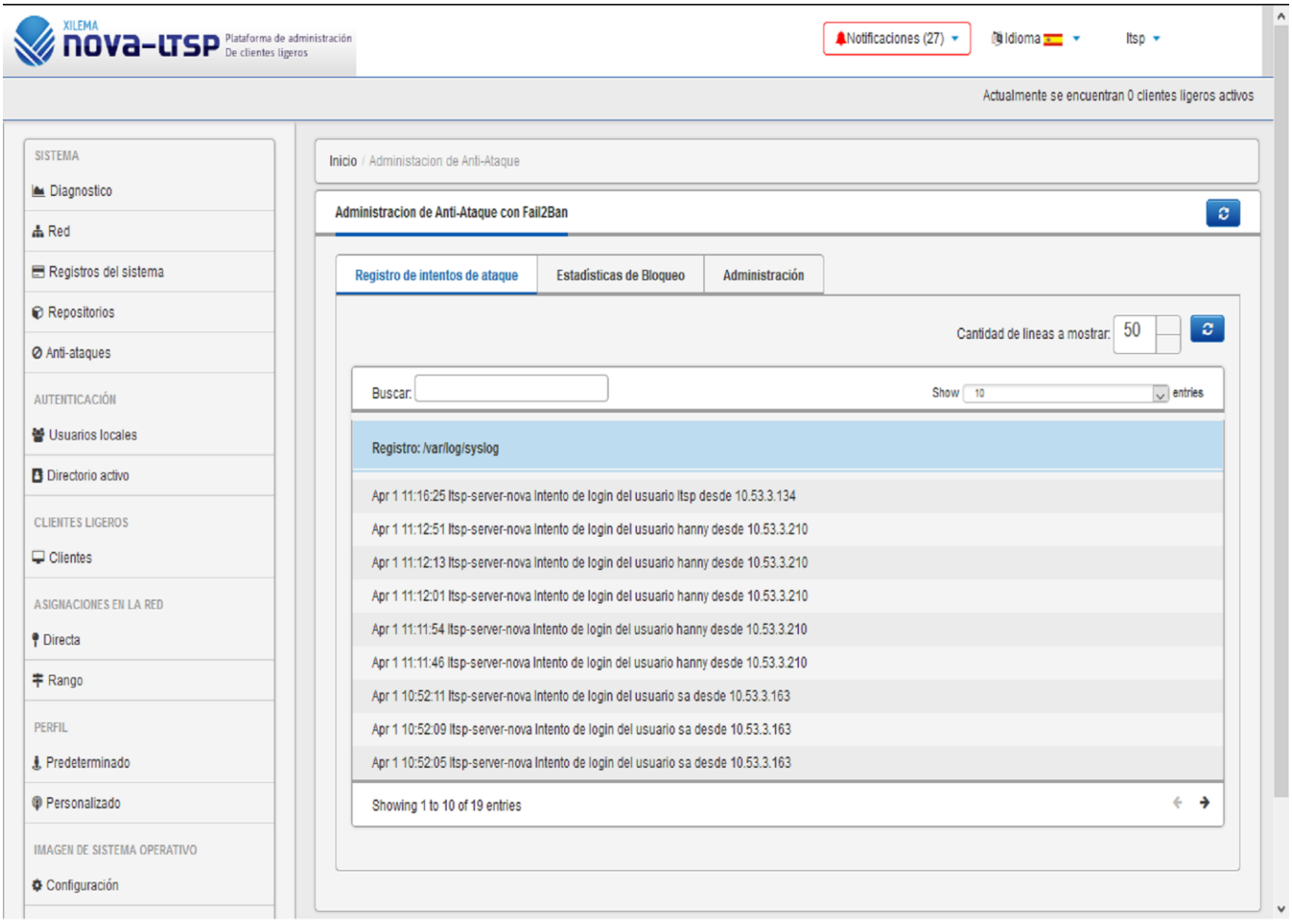



*Figura 15 Interfaz principal del módulo de bloqueo de conexiones remotas a Nova-LTSP. Elaboración propia.*

## 3.9 Conclusiones del capítulo

- En este capítulo se abordaron los elementos de la implementación del módulo de bloqueo a conexiones remotas que intentan accesos por fuerza bruta, así como las pruebas realizadas al mismo y los resultados obtenidos, por lo que se puede concluir:
- La elaboración del diagrama de componentes permitió una mejor comprensión de la estructura de los componentes de la propuesta de solución.
- La implementación del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP facilitó la obtención de una aplicación funcional lista para su uso.
- La aplicación de las pruebas unitarias, integración, funcionales, seguridad y aceptación permitieron detectar errores que afectaban el funcionamiento del módulo, lo que posibilitó corregirlos a tiempo para que el mismo cumpliese con los requisitos funcionales definidos en la etapa de análisis.

## Conclusiones

De manera general se puede concluir sobre la presente investigación:

- El análisis de los referentes teóricos y de las herramientas informáticas que implementan mecanismos de bloqueo a conexiones remotas estudiadas evidenció la necesidad de desarrollar un módulo que permitiera el bloqueo de conexiones remotas a la plataforma de clientes ligeros Nova-LTSP.
- La selección de herramientas, lenguajes y tecnologías permitió la implementación del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP.
- Las pruebas diseñadas y ejecutadas permitieron detectar las deficiencias presentes en el módulo desarrollado, corrigiéndose las mismas logrando un producto más seguro y funcional.
- La aplicación de la técnica de *IADOV* propició la evaluación satisfactoria del módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP.

## Recomendaciones

Una vez concluido el trabajo de diploma se recomienda lo siguiente:

- Implementar un mecanismo para  personalizar las reglas de detección de intrusos.

## Referencias Bibliográficas

**[1] ACERO,** Fernando. DenyHost, una magnífica herramienta de corrección de un error. 2008.

**[2] CABEZA**, Yanet. Módulo de JAVA para la herramienta Auditoría de Código Fuente. Tesis para optar por el título de Ingeniero en Ciencias Infotmáticas, Universidad de las Ciencias Informáticas, La Habana, 2015.

**[3] CRESPO**, Adrian. Redeszone. [En línea]. 2017. [Consultado el: 15 de octubre de 2018]. Disponible en: [https://www.redeszone.net/2017/10/20/ataques-fuerza-bruta-debo-saber-puedo-protegerme/].

**[4] COHN,** Mike. Agile Estimating and Planing. 2005. ISBN 0-13-147941-5. p. 179-350.

**[5] DHAWAN**, Chander. Enabiling Remote Access. 2002. ISBN 0-17-151831-1.

**[6] DÍAZ**, Lexys Manuel. Módulo para el monitoreo en tiempo real de clientes ligeros desde Nova-LTSP. Tesis para optar por el título de ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2017.

**[7] EUGENIN,** Daniel. Fail2Ban para protección contra intentos de conexión. [En línea]. 2015. [Consultado el: 26 de marzo de 2019]. Disponible en [https://soporte.itlinux.cl/hc/es/articles/200121798-Fail2Ban-para-protecciónn-contra-intentos-de-conexión].

**[8] GAUCHAT,** Juan Diego. El gran libro de HTML5, CSS3 y Javascript. 2012. Barcelona:Marcombo. ISBN 978-84-267-1782-5.

**[9] GÓMEZ** Dayli, **PÉREZ** Yasiel, **PÉREZ** Yadiel. Plataforma de Administración de clientes ligeros. VIII Taller Internacional de Tecnologías de Software Libre y Código Abierto. Universidad de las Ciencias Informáticas, La Lisa, La Habana, Cuba. 2018.

**[10] GÓMEZ**, Ramón. Programación Avanzada en SHELL. [En línea]. 2015. [Consultado el: 9 de diciembre de 2108]. Dsiponible en [http://www.informatica.us.es/~ramon/articulos/Programacion-BASH].

**[11] GUEVARA,** Alexander. DevCode. [En línea]. 2016. [Consultado el: 10 de febrero de 2019]. Disponible en: [https://devcode.la/blog/frontend-y-backend/].

**[12] HERNÁNDEZ**, Enrique. El Lenguaje Unificado de Modelado (UML). [En línea]. 2010. [Consultado el: 6 de diciembre de 2018]. Disponible en: [http://www.disca.upv.es/].

**[13] INFANTE,** Montero. Curso Django para perfeccionistas con deadlines. [En línea]. 2012. [Consultado el: 22 de novimebre de 2018]. Disponible en [http://www.academia.edu/9510572/maestrosdelweb_curso_django]

**[14] JIMÉNEZ** Jessy, **VIVAR** Cristina, **ERAZO** Ronal. Grupo #4. [En línea]. 2015. [Consultado el: 14 de noviembre de 2018]. Disponible en: [https://iptables.webnode.es/definicion-de-iptables/].

**[15] KANTER**, Joel P. Understanting Thin-client/server Computing. 1998. New York, Estados Unidos.

**[16] LARMAN,** Craig. UML y patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Segunda edición. 1999. México. Prentice Hall. ISBN 970-17-0261-1.p.164-238.

**[17] MANCOMÚN.** OSSEC: Sistema de detección de intrusos.[En línea]. 2017. [Consultado el: 24 de marzo de 2019]. Disponible en: [https://www.mancomun.gal/es/artigo-tic/ossec-analise-e-monitorizacion-de-registros-sistemas].

**[18] MOLINA** Marco Antonio, **GONZÁLEZ** Victor Enrique. Sistema de prevención de intrusos basado en OpenBSD y Snort. México, 2008.

**[19] NAVAJA,** Antonio. Guía completa de CSS3. [En línea]. 2012. [Consultado el: 20 de noviembre de 2018]. Disponible en: [https://openlibra.com/es/book/guia-completa-de-css3].

**[20] OROVENGUA** Javier, LinuxParty. [En Línea]. 2013. [Consultado el: 20 de noviembre de 2018]. Disponible en: [https://www.linux-party.com/57-seguridad/8971-instalar-denyhosts-para-bloquear-ataques-al-servidor-ssh-ataques-de-fuerza-bruta/].

**[21] POSTGRESSQL**. The PostgreSQL Global Development Group. [En línea]. 2016. [Consultado el: 20 de marzo de 2019] Disponible en [http://www.postgresql.org/about/].

**[22] PRESSMAN**, Roger. Ingeniería de Sofware: Un enfoque Práctico. Quinta Edición. New York, Estados Unidos: McGraw-Hill, 2002, p.207-384.

**[23] QUINTANA** Yoandri, **CAMEJO** Lianet, **DÍAZ** Abel. Diseño de la base de datos para sistemas de digitalización y gestión de medidas. [En línea:]. 2011. [Consultado el: 6 de diciembre de 2018]. Disponible en: [http://laboratorios.fi.uba.ar/lie/Revista/Articulos/].

**[24] RAMOS** Galo, **PÁEZ** Jaime. Análisis del método para calificación de software QSOS para la selección de software aplicable a procesos educativos. [En línea]. 2011. [Consultado el: 19 de marzo de 2019]. Disponible en: [http://scielo.sld.cu/].

**[25] RICHARDS** David. Linux Thin Cliente Networks Desing and Deployment. 2007.

**[26] RODRÍGUEZ**, Tamara. Metodología de desarrollo para la actividad productiva en la UCI. La Lisa, La Habana, Cuba, 2014.

**[27] SOMMERVILLE**, Ian. Software engineering, eigth edition. Harlow: Pearson Education Limited, 2007, 36p.

**[28] TECNOLOGÍAS E INFORMACIÓN**. Modelo de datos: Modelo Conceptual, Físico y Lógico. [En línea]. 2018. [Consultado el: 15 de marzo de 2019]. Disponible en: [https: // www.tecnologías-información.com/modelos-datos.html].

**[29] TARAZONA**, César. Amenazas informáticas y seguridad de la información. 2016.

**[30] WORDPRESS,** Ingeniería de Software. [En línea]. 2012. [Consultado el: 14 de enero de 2019]. Disponible en: [https://arlethparedes.com/2012/08/27/patrones-de-arquitectura-vs-patrones-de-diseño/].

## Anexos

### Anexo 1: Entrevista realizada al especialista encargado del despliegue de la herramienta Nova-LTSP.

**Objetivo:** Conocer, analizar y determinar las vulnerabilidades que presenta la plataforma de clientes ligeros Nova-LTSP respecto a la seguridad de los datos que almacena.

1. ¿Qué servicios posee la plataforma de clientes ligeros Nova-LTSP?
2. ¿Considera que esta herramienta es propensa a ataques?
3. ¿Se ha detectado algún ataque? ___Sí ___No
4. ¿Sería aplicable utilizar algún mecanismo de detección/prevención de intrusos?

### Anexo 2: Descripción de las historias de usuario

*Tabla 11: Historia de usuario Buscar intentos de inicio de sesión. Elaboración propia.*

| **Historia de usuario** | |
|---|---|
| **Número:** HU_ 3 | **Nombre:** Buscar intentos de inicio de sesión |
| **Prioridad:** Alta | **Iteración Asignada: 1** |
| **Programador:** Mayra de la O Barrientos | **Tiempo Estimado:** 20 horas |
| **Riesgo en Desarrollo:** N/A | **Tiempo Real:** 17 horas |

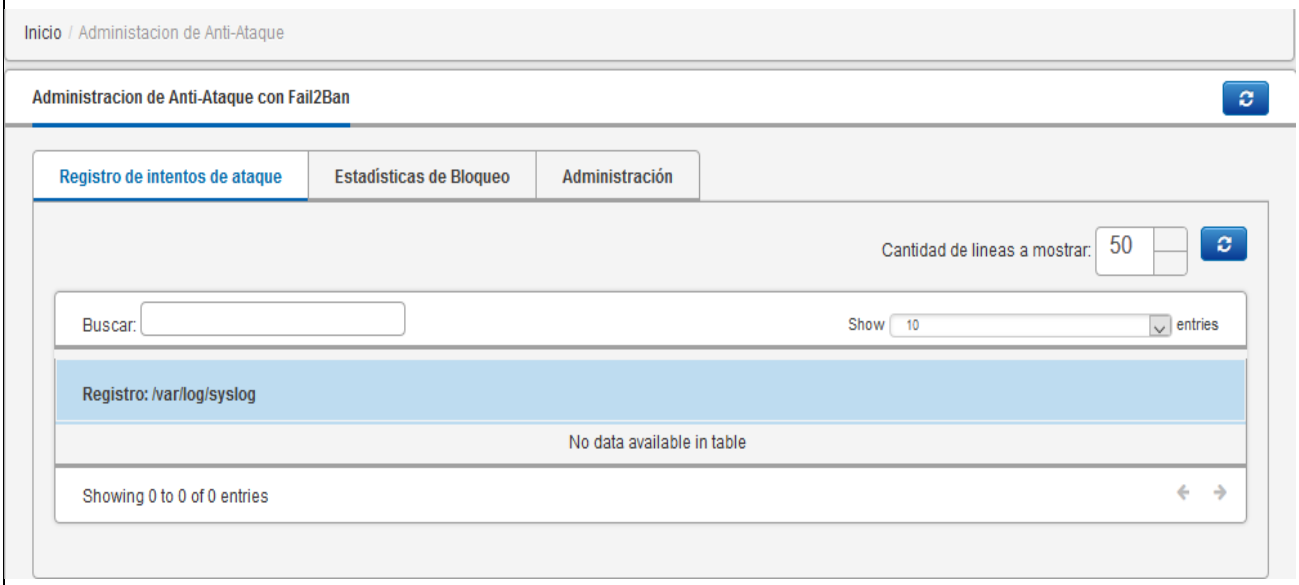
| |
|---|
| **Descripción:**<br><br>El sistema permite buscar un intento de inicio de sesión determinado. Para ello el administrador debe acceder al listado de registro de intentos de inicio de sesión e introducir en el campo de texto de búsqueda el dato deseado. |
| **Observaciones:**<br><br>En caso de que no exista el dato introducido por el administrador la lista se muestra vacía. |
| **Prototipo:**<br><br>  |

*Tabla 122: Historia de usuario Mostrar estadísticas de los intentos de inicio de sesión de Nova-LTSP. Elaboración propia.*

| **Historia de usuario** | |
|---|---|
| **Número:** HU_4 | **Nombre:** Mostrar estadísticas de los intentos de inicio de sesión de Nova-LTSP |
| **Prioridad:** Alta | **Iteración Asignada: 1** |
| **Programador:** Mayra de la O Barrientos | **Tiempo Estimado:** 3 semanas |

| **Riesgo en Desarrollo:** N/A | **Tiempo Real:** 2 semana |
|---|---|

**Descripción:**

El sistema permite mostrar las estadísticas de bloqueo, donde se muestran los ataques a la plataforma Nova-LTSP y los ataques mediante SSH.

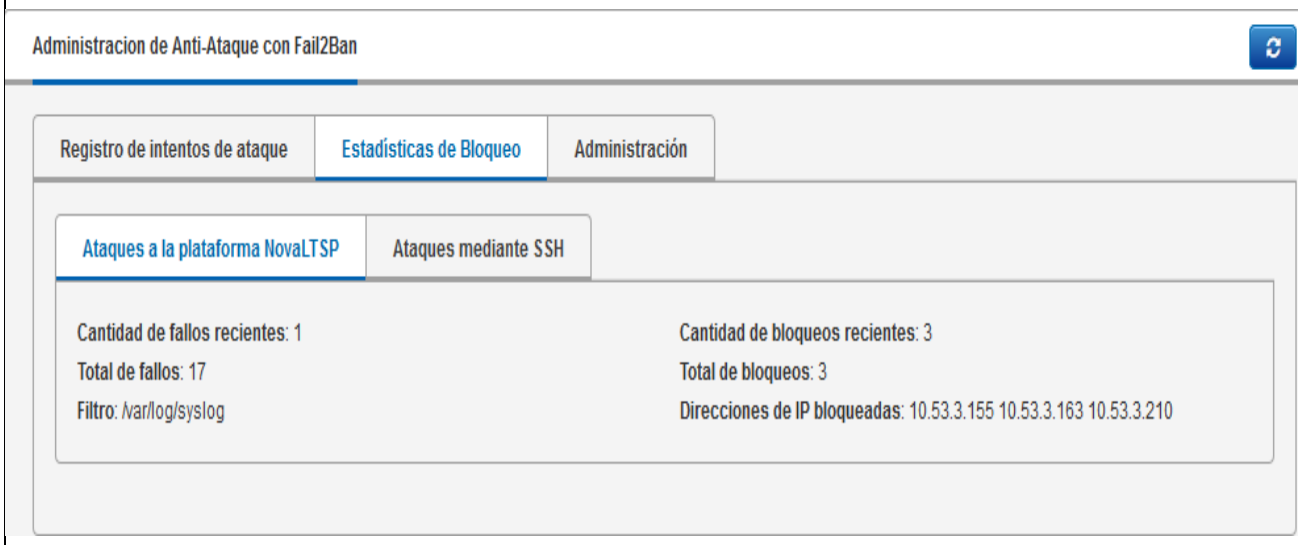
**Observaciones:**

- El usuario debe acceder a la interfaz principal de la plataforma de clientes ligeros Nova-LTSP
- La URL para acceder a la plataforma está compuesta por:

  La dirección donde está publicado el servicio, el puerto 1000

  Ejemplo: http://192.168.56.101:1000/

**Prototipo:**



## Anexo 3 Acta de aceptación

Acta de aceptación de productos de trabajo

Universidad de las Ciencias Informáticas

ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO

En cumplimiento del **Convenio de colaboración** establecido entre el **Centro de Software Libre (CESOL)** y la estudiante **Mayra de la O Barrientos** de la Facultad 1 de la Universidad de las Ciencias Informáticas y en función de la ejecución del proyecto: **Módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP**, se hace entrega del producto que se relaciona a continuación:

- Módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova-LTSP

La parte Cliente, luego de haber revisado el producto de trabajo relacionado anteriormente procede a firmar la aceptación de los mismos en total conformidad.

| *Entrega* | Recibe |
|---|---|
| ***Nombre y Apellidos:*** | **Nombre y Apellidos:** |
| *Mayra de la O Barrientos* | Hanny Valdés Hernández |
| ***Cargo:*** Estudiante Facultad 1 | **Cargo:** Jefe de Departamento |
| **Firma:** | **Firma:** |

Fecha: 25 de marzo de 2019

## Anexo 4: Encuesta realizada a especialistas del Centro de Software Libre (CESOL)

**Objetivo:** Evaluar la propuesta de solución desarrollada

Especialista, le invito a responder el siguiente cuestionario con el fin de conseguir su colaboración en la presente investigación, solicito que exprese en sus respuestas criterios verídicos que guíen al autor de la investigación:

1. ¿Considera importante la implementación de un sistema que permita el bloqueo de conexiones remotas que intentan accesos por fuerza bruta a la plataforma Nova-LTSP?

   ___Sí ___No ___ No sé

2. ¿Considera usted correcta la forma en que se realiza el bloqueo de conexiones remotas a la plataforma Nova-LTSP?
   ___Sí ___No ___ No sé

3. ¿Considera usted factible la implementación de un módulo que permita el bloqueo de las conexiones remotas que intentan accesos por fuerza bruta a la plataforma Nova-LTSP?
   ___Sí ___No ___ No sé

4. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la solución desarrollada.
   ___me gusta mucho ___me disgusta más de lo que me gusta
   ___me gusta más de lo que me disgusta ___no me gusta nada
   ___me da lo mismo ___no sé decir

5. ¿Qué opina usted acerca de los beneficios que traería para la universidad disponer del sistema propuesto para el bloqueo de conexiones remotas que intentan accesos por fuerza bruta en la plataforma Nova-LTSP?