Universidad de las Ciencias Informáticas

FACULTAD 1

Título: Sistema de Prevención de Intrusos para Nova Servidores 7

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

Autor: Michel Pedrera Suen

Tutoras:

Ing. Prof. Inst. Estela Odelsa Martín Coronel

Ing. Prof. Inst. Hanny Valdés Hernández

Septiembre de 2020

AGRADECIMIENTOS

Agradezco a mi familia por la insistencia permanente de convertirme en un profesional.

A cada uno de mis profesores, a todos, independientemente de que tan positivos o negativos hayan sido.

A mi tutora Hanny, que aunque la montaron tarde en este barco, puso todo el empeño que estuvo a su alcance para que saliéramos adelante, a pesar de mi empeño en trabajar solo. Esto no es un viaje que se hace estando solo.

A mi tutora Estela, quien no solo ha estado en momentos de tutoría, sino que me ha acompañado de corazón en otros proyectos personales, y que siempre es una buena compañía.

A Yasiel, que más allá de su deber siempre está disponible a ayudar, porque le gusta.

A Mayra que siempre ha estado tan dispuesta a prestar ayuda, y que de tanta ayuda ha sido sin saberlo.

A Serguey, por su dirección correcta, y por ser con nosotros uno de nosotros.

Agradezco a todos mis compañeros, los que vinieron, los que se fueron y los que permanecieron siempre. A Javier y a Manuel, que me han aguantado siempre con mucha paciencia y discreción. A Sueyaile y a Yadín por aguantar mis pesadeces. A Darelys por la compañía fiel, y también por aguantarme. A Natali, siempre presta y disponible. A Ludmila que de una forma u otra siempre me hace reír. A Sulema que aunque siempre compartamos momentos de “trabajo” pasamos buen tiempo. A Guilarte, Julio y Jose, porque la locura a veces es la mejor medicina para los malos momentos. A Beatriz que, en silencio, siempre está dispuesta a ayudar, y a Sergio, que no tan calladamente (y a regañadientes) siempre acaba ayudando también. Sin ellos no hubiese sobrevivido.

Agradezco a mis compañeros de Taller Literario. A Fran, por el buen humor eterno acompañado de sagacidad y locura; y a Jennifer, que con pocas palabras y un humor excelente siempre regala una buena cara. A Dayana, con su coraje y necesidad permanente de defender sus ideales. A Alexis, por sus palabras sabias pronunciadas en voz baja, pero siempre certeras. A Rubiel, también por sus palabras sabias, pero dichas con más jolgorio a veces. A Rosita, por haberme arrastrado y sacado del cascarón. Y a Martha, quien me ha impulsado y me ha enseñado a apreciar el talento –tanto el propio como el ajeno–, y que me ha dado confianza y me ha impulsado a hacer las cosas de forma correcta. Gracias también por el incansable talento que todos tiene para ofrecer.

Agradezco a mi Alarcos: Héctor, Rachel, Alain, Arianna, Osiel, Jose, Luis, Alexis, Carlos, Lisset, Yoslenys y en especial a César. Lo siento, para expresar mi agradecimiento necesitaría redactar un ensayo individual por cada uno. Gracias por aceptarme, por convertirme en uno de ustedes, por aguantar alguna que otra perreta, por sus consejos siempre útiles, por el amor que nos tenemos entre todos, unos más que otros. Gracias por su infinito talento y por aceptar también el mío que por escaso

no han hecho ver como menor, gracias por hacerme probar cosas a las que nunca me hubiera lanzado solo. Sé que nunca tendré momentos tan intensos como los que he vivido y viviré con Alarcos.

Agradezco especialmente a mi tío Andrés, quien ha confiado ciegamente en mí, esperando lo mejor de este futuro ingeniero y que sin su apoyo esta travesía se hubiera quedado a medias.

Agradezco especialmente a mi hermano Marco, que sin relacionarse con ninguno de mis procesos académicos me ha dado siempre su amor.

Agradezco especialmente a dos personas incomparables, a quienes si no hubiera conocido mi vida careciera de toda la alegría y los colores que hoy tiene. La palabra amor se queda corta para describir lo que profesamos. A mis dos pedazos de vida: Leo, el único que ha estado por tanto tiempo y el único que merece ser nombrado; y Popi, sin la cual prefiero no vivir porque no sé cómo hacerlo.

(…) Y a los negativos, a los que me hicieron probar el infierno diciendo que no podía o que no debía: su persistencia me ha hecho más fuerte (…). ¡Así que gracias!

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____
Michel Pedrera Suen

Autor

_____ _____

Ing. Estela Odelsa Martín Coronel Ing. Hanny Valdés Hernández

Tutora Tutora

DATOS DE CONTACTO

Datos del Autor

Nombre y apellidos: Michel Pedrera Suen

Correo electrónico: mpedrera@estudiantes.uci.cu

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

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

Datos de las Tutoras

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, La Habana, Cuba, Código Postal 19370.

Nombre y apellidos: Estela Odelsa Martín Coronel

Correo electrónico: eomartin@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, La Habana, Cuba, Código Postal 19370.

RESUMEN

La Distribución Cubana de GNU/Linux Nova Servidores es un sistema que permite el trabajo con servidores. Actualmente, esta carece de mecanismos de prevención de intrusos que intenten acceder maliciosamente al sistema, lo que trae como consecuencia vulnerabilidad de la información almacenada. El objetivo de este trabajo consiste en desarrollar un sistema de prevención de intrusos para Nova Servidores 7, que será administrado de forma remota desde una computadora con la Distribución Cubana de GNU/Linux Nova, y que garantice la implementación de los mecanismos mencionados. La propuesta de solución es guiada por la metodología de desarrollo de software Proceso Unificado Ágil en su variación para la Universidad de las Ciencias Informáticas en el escenario de historias de usuario. Además, se realiza un análisis de las diferentes herramientas, tecnologías, y lenguajes a utilizar para el modelado y desarrollo. Se realizaron pruebas funcionales, de unidad, integración, aceptación y de seguridad para comprobar el correcto funcionamiento y calidad del sistema para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos maliciosos.

Palabras clave: Nova Servidores 7, prevención de intrusos, seguridad, sistema.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

En la actualidad internacional, imaginarse el mundo sin la interacción con las facilidades que ofrece la Internet se torna poco usual. La rápida asimilación de las nuevas tecnologías referidas a ella es cada vez más necesaria y frecuente. El funcionamiento de estas se refieren principalmente al empleo de interconexiones varias. Debido a una constante evolución en las tecnologías asociadas a la interconexión, las Tecnologías de la Información y la Comunicación (TIC) han sido convertidas en un elemento estratégico para el crecimiento y transformación de las organizaciones a nivel mundial. El aporte de estas al perfeccionamiento de los productos de software que hacen posible la informatización de los procesos en las diferentes esferas de la sociedad es innegable. Uno de los temas principales acerca del uso de las TIC a nivel mundial es la seguridad.

La seguridad de la información es imprescindible para el desarrollo y manutención de cualquier sistema que, precisamente, gestione o maneje datos. Existen cientos de sitios en Internet que ofrecen información, herramientas y métodos para vulnerar sistemas informáticos. Es importante atender al concepto de informática como 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 (*BACA*, 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 informática se clasifican en diferentes tipos: preventivos, consisten en prevenir la ocurrencia de 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 adecuado, y los disuasivos, se encargan de desalentar a los perpetradores de que cometan su ataque para minimizar los daños que puedan tener los bienes (*POSTIGO,2020*).

Entre los mecanismos preventivos están los sistemas de prevención de intrusos (IPS), también conocidos como sistemas de detección y prevención de intrusos (IDPS), los cuales son dispositivos de seguridad de red que monitorean las actividades de la red o del sistema en busca de actividad maliciosa. Las funciones principales de los sistemas de prevención de intrusos son identificar actividades maliciosas, registrar información sobre esta actividad, informarla e intentar bloquearla o detenerla.(SCARFONE, 2007) El reconocido autor Robert Newman en Computer Security: Protecting Digital Resources dijo que “los sistemas de prevención de intrusos se consideran extensiones de los sistemas de detección de intrusos(...)” porque ambos monitorean el tráfico de la red y/o las actividades del sistema en busca de actividad maliciosa. Las principales diferencias son, a diferencia de los

sistemas de detección de intrusos, los sistemas de prevención de intrusos se colocan en línea y pueden prevenir o bloquear activamente los intrusos que se detectan. (NEWMAN, 2010)

Debido a que los IPS pueden tomar medidas como enviar una alarma, descartar paquetes maliciosos detectados, restablecer una conexión o bloquear el tráfico desde la dirección IP infractora (BOYLES, 2016) son usados para la seguridad en muchas entidades del mundo entre las que se encuentra el Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas en Cuba (creada por el Comandante en Jefe Fidel Castro Ruz en su afán de lograr una soberanía tecnológica y seguridad para la misma así como la informatización de todas las esferas de la sociedad). CESOL tiene entre sus objetivos el desarrollo de la distribución cubana de GNU/Linux Nova la cual usa en parte los IPS para la mantención de la seguridad por los motivos antes mencionados.

Esta distribución está basada en la amigabilidad lo cual se garantiza con la interacción intuitiva que persigue minimizar el cambio brusco de las personas familiarizadas con sistemas *Microsoft Windows*. Esta distribución posee además una variante de sistema operativo para la configuración y manejo de servidores cuyo nombre es precisamente Nova Servidores, del cual, su última versión es Nova Servidores 7. En dicho sistema se intentó alcanzar la mejor calidad de prestación de servicios, además de mantener la seguridad en todos los procesos que pueda realizar la computadora como servidor. Es necesario entender que debe haber facilidades tanto para el manejo de sistemas operativos de computadoras de uso personal como de ordenadores destinados a realizar funciones de servidores con Nova Servidores. Para ello es imprescindible además la implementación de sistemas de seguridad que resguarden la integridad, no solo de dichos equipos si no de la información que en ellos se maneja de forma independiente o en su interacción.

A pesar de que otras distribuciones trabajen su seguridad con IPS, Nova Servidores 7 carece de un mecanismo que prevenga los posibles intrusos lo cual provoca vulnerabilidad en los servicios y protocolos con los que se trabaja en el servidor mientras no se instale y configure manualmente un IPS externo al sistema.

Determinada la mencionada **situación problemática** se identifica que el **problema de investigación** a resolver es: ¿Cómo garantizar la prevención de intrusos en Nova servidores 7?

Partiendo del problema planteado se identifica como **objeto de estudio** de la investigación: mecanismos de prevención de intrusos, se define además como **campo de acción**: herramienta de prevención de intrusos.

Para darle solución al problema anteriormente planteado se define como **objetivo general** Desarrollar una herramienta que permita la prevención de intrusos en Nova Servidores 7.

Para poder cumplir dicho **objetivo general** es preciso definir **objetivos específicos**:

1. Elaborar el marco teórico de la investigación sobre aplicaciones que garanticen la prevención de intrusos.

2. Diseñar una herramienta que garantice la prevención de intrusos para Nova Servidores 7.

3. Implementar una herramienta que garantice la prevención de intrusos para Nova Servidores 7.

4. Evaluar la herramienta que garantice la prevención de intrusos para Nova Servidores 7.

Para facilitar la investigación se establecen las siguientes **preguntas científicas**

¿Qué sistemas de prevención de intrusos son más usados en la actualidad?

¿Qué sistema de prevención de intrusos es más adecuado para Nova Servidores 7?

¿El desarrollo de una herramienta informática permitirá la correcta prevención de intrusos en Nova Servidores 7?

¿Qué herramientas, tecnología y metodología son necesarias para desarrollar una herramienta que permita la prevención de intrusos en Nova Servidores 7?

¿Qué elementos se deben considerar para analizar y diseñar la herramienta que permita la prevención de intrusos en Nova Servidores 7?

¿Qué resultados serán obtenidos al evaluar la herramienta que permita la prevención de intrusos en Nova Servidores 7?

Para el desarrollo de la investigación se utilizan los siguientes **métodos científicos**:

Métodos empíricos

Entrevista: usada para la obtención de información referente a los sistemas de prevención de intrusos y definir a que servicios de Nova Servidores 7 es necesario aplicarles un IPS. De ella fue posible identificar los principales problemas existentes, además de recoger opciones y recomendaciones para la solución.

Observación: utilizada para hacer un análisis y evaluación del comportamiento de los mecanismos de seguridad, con el objetivo de identificar elementos que aporten a la investigación.

Métodos teóricos

Histórico-lógico: El uso de este método permite conocer la evolución que han tenido los procesos de los mecanismos de seguridad en Nova Servidores, así como las causas que dieron paso a su surgimiento, las características comunes que se han mantenido con el transcurso del tiempo y las nuevas que han surgido.

- **Analítico-sintético:** Empleado 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.

La investigación consta de introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

Capítulo 1: Fundamentación teórica de la herramienta informática para la prevención de intrusos en Nova Servidores 7.

En este prevalece el estudio de los elementos relacionados con los diferentes sistemas de prevención de intrusos, realizando un análisis de ellos en cuanto a características, funciones, ventajas y desventajas, y así definir cuál será estudiado y empleado en la investigación. Se ofrecen definiciones asociadas a los mismos así como la fundamentación de la selección de la metodología de desarrollo de software, herramientas y lenguaje de programación a utilizar.

Capítulo 2: Análisis y diseño de la herramienta informática para la prevención de intrusos en Nova Servidores 7.

Aquí se realiza el análisis y diseño de la herramienta informática que posibilite la prevención de intrusos en Nova Servidores 7. Se especifican los requisitos funcionales y no funcionales.

Capítulo 3: Implementación y prueba de la herramienta informática para la prevención de intrusos en Nova Servidores 7.

En este capítulo se da una descripción acerca de cómo será la implementación de la aplicación, a la propuesta de solución realizada en el capítulo anterior se le realiza pruebas, se expone el proceso de las mismas para la evaluación de la herramienta informática para la prevención de intrusos en Nova Servidores 7.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA PREVENCIÓN DE INTRUSOS

1.1 INTRODUCCIÓN

Este capítulo abarca el estudio de diferentes herramientas de prevención de intrusos ampliamente usadas en el mundo, dónde se analiza cada una teniendo en cuenta sus características y funcionalidades para poder seleccionar la más adecuada para la distribución Nova Servidores 7. A partir de ello, se analizan y definen las herramientas, tecnologías y metodología a utilizar para el desarrollo de una propuesta de solución.

1.2 CONCEPTOS GENERALES ASOCIADOS A LA PREVENCIÓN DE INTRUSOS.

Atendiendo a que el lector pueda alcanzar una mejor comprensión de los temas que serán abordados en este capítulo, relacionados con el objeto de estudio de la investigación, a continuación se enuncian los conceptos fundamentales relativos al problema.

1.2.1 SISTEMA DE PREVENCIÓN DE INSTRUSOS (IPS)

Es una tecnología de seguridad de red / prevención de amenazas que examina los flujos de tráfico de red para detectar y prevenir vulnerabilidades. Los *exploits*[1] de vulnerabilidad generalmente se presentan en forma de entradas maliciosas a una aplicación o servicio objetivo que los atacantes usan para interrumpir y obtener el control de una aplicación o máquina. Después de una explotación exitosa, el atacante puede deshabilitar la aplicación de destino (lo que resulta en un estado de denegación de servicio), o puede acceder potencialmente a todos los derechos y permisos disponibles para la aplicación comprometida.

Como componente de seguridad en línea, el IPS debe funcionar eficientemente para evitar degradar el rendimiento de la red. También debe funcionar rápido porque las vulnerabilidades pueden ocurrir casi en tiempo real. El IPS también debe detectar y responder con precisión, para eliminar amenazas y falsos positivos (paquetes legítimos mal interpretados como amenazas). La detección de anomalías estadísticas toma muestras de tráfico de red al azar y las compara con un nivel de rendimiento de línea de base precalculado. Cuando la muestra de la actividad del tráfico de la red está fuera de los parámetros del rendimiento de la línea base, el IPS toma medidas para manejar la situación. (*MARTELINI Y MALIZA, 2017)*

[1] Exploit: es un fragmento de software, fragmento de datos o secuencia de comandos y/o acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo.

IPS se construyó y lanzó originalmente como un dispositivo independiente a mediados de la década de 2000. Sin embargo, esto fue en el advenimiento de las implementaciones de hoy, que ahora se integran comúnmente en las soluciones de gestión de amenazas unificadas (UTM) (para pequeñas y medianas empresas) y cortafuegos de próxima generación (a nivel empresarial). (*MARTELINI Y MALIZIA, 2017*)

1.2.2 PREVENCIÓN

La prevención de riesgos de ciberseguridad está a la orden del día. Los sistemas digitales de nuestras empresas se exponen casi de forma diaria a un gran número de amenazas que pueden hacer que nuestro sistema se vea dañado o desaparezca por completo con todo lo que almacene en su interior. Tomar medidas al respecto para evitar los posibles ataques es algo fundamental, dado que ningún negocio se quiere encontrar ante una situación sin vuelta atrás en la cual se pierda información valiosa. (RODRÍGUEZ CANFRANC, 2019)

1.2.3 SEGURIDAD

La Organización Internacional de Normalización (ISO por sus siglas en inglés) hace referencia a que la seguridad consiste en minimizar la vulnerabilidad de bienes y recursos. La seguridad en un sistema se basa en los mecanismos de protección que ese sistema proporcione. Estos mecanismos deben permitir el control sobre qué usuarios tienen acceso a los recursos del sistema y qué tipo de operaciones pueden realizar sobre esos recursos. (*POSTIGO*, 2020)

1.2.4 SEGURIDAD INFORMÁTICA

Se denomina Seguridad Informática al conjunto de métodos y herramientas destinados a proteger los bienes informáticos de una institución. La seguridad en la información tiene el objetivo de garantizar (PFLEEGER, 2015):

- Confidencialidad: la información o los activos informáticos son accedidos sólo por las personas autorizadas para hacerlo. La información confidencial debe estar disponible únicamente para un grupo de individuos pre-establecido. La transmisión y uso de información no autorizada debe restringirse. Por ejemplo, la confidencialidad de información garantiza que la información persona lo financiera no esté al alcance de individuos no autorizados con propósitos malintencionados tales como: robo de identidad o fraude de crédito.
- Integridad: los activos o la información sólo pueden ser modificados por las personas autorizadas y de la forma autorizada. La información no se debe alterar de forma tal que la

reproduzcan incompleta o incorrecta. Se debe restringir a los usuarios no autorizados de la capacidad de modificar o destruir información confidencial.

• Disponibilidad: los activos informáticos son accedidos por las personas autorizadas en el momento requerido. La información debe estar accesible a usuarios autorizados, en cualquier momento, es decir, cuando se necesite. La disponibilidad garantiza que la información pueda obtenerse con una frecuencia y puntualidad acordadas. Suele medirse en términos de porcentajes y se acepta de manera formal en los Acuerdos de Nivel de Servicio (SLA por sus siglas en inglés) usados por los proveedores de servicios de red y los clientes corporativos.

1.3 ESTUDIO DE HERRAMIENTAS DE PREVENCIÓN DE INTRUSOS EXISTENTES

El IPS a menudo se ubica directamente detrás del cortafuego y proporciona una capa complementaria de análisis que selecciona negativamente el contenido peligroso. A diferencia de su predecesor, el Sistema de detección de intrusos (IDS), que es un sistema pasivo que escanea el tráfico e informa sobre amenazas, el IPS se coloca en línea (en la ruta de comunicación directa entre el origen y el destino), analizando activamente y tomando acciones automatizadas en todos flujos de tráfico que ingresan a la red. (*MARTELINI Y MALIZIA, 2017*)

A continuación se hace una descripción de algunos IPS utilizados, dejando plasmadas algunas de las características que tienen en común, rasgos que la diferencian y principales funcionalidades.

1.3.1 SPLUNK

Splunk Enterprise Security (ES) es el centro neurálgico del ecosistema de seguridad, brindando a los equipos la información para detectar y responder rápidamente a ataques internos y externos, simplificar la gestión de amenazas y minimizar el riesgo. ES ayuda a los equipos a obtener visibilidad de toda la organización e inteligencia de seguridad para monitoreo continuo, respuesta a incidentes, operaciones de Centro de Operaciones de Seguridad (SOC) y brinda a los ejecutivos una ventana al riesgo comercial. (PAGE, DELGADO Y DIAKUN, 2017)

• Monitoreo continuo: visualice claramente la postura de seguridad con tableros, indicadores clave de seguridad, umbrales estáticos y dinámicos, y tendencias

• Prioridad y actuación: optimice, centralice y automatice los flujos de trabajo de respuesta a incidentes con alertas, registros centralizados e informes y correlaciones predefinidos

• Realizar investigaciones rápidas: utilice búsquedas y correlaciones ad-hoc [2]para detectar actividades maliciosas.

• Manejar investigaciones de varios pasos: rastrear actividades asociadas con sistemas comprometidos y aplicar la metodología de matar la cadena para ver el ciclo de vida del ataque

Splunk ES ofrece a las organizaciones la capacidad de (MARLETTE, 2016):

• Mejorar las operaciones de seguridad con tiempos de respuesta más rápidos.

• Mejorar la postura de seguridad al obtener visibilidad de extremo a extremo en todos los datos de la máquina

• Aumentar las capacidades de detección e investigación utilizando análisis avanzados.

• Tomar decisiones mejor informadas aprovechando la inteligencia de amenazas

Además este presenta revisión y clasificación de incidentes, está construido sobre una plataforma de *Big Data*[3] para inteligencia de seguridad con búsquedas ad hoc que permiten a los equipos de seguridad comprender rápidamente qué ataques están ocurriendo en su entorno para determinar el mejor curso de acción.

1.3.2 OPEN WIPS-NG

OpenWIPS-ng es un IPS inalámbrico de código abierto y modular. Se compone de tres partes:

• Sensor (es): dispositivos "tontos" que capturan el tráfico inalámbrico y lo envían al servidor para su análisis. También responde a los ataques.

• Servidor: agrega los datos de todos los sensores, los analiza y responde a los ataques. También registra y alerta en caso de un ataque.

• Interfaz: la interfaz gráfica de usuarios administra el servidor y muestra información sobre las amenazas en sus redes inalámbricas.

Para su uso es necesaria una tarjeta inalámbrica que admite el modo de monitor.

La idea de su creación viene a partir de conseguir un IPS inalámbrico lo más barato posible. Este es también de código abierto.

Se puede decir que son ventajas que no sea necesario volver a compilar el servidor, además si el cliente tiene un código propietario, no es necesario distribuirlo.

Tipos de complementos que posee

---

[2] **Ad Hoc** Que es apropiado, adecuado o especialmente dispuesto para un determinado fin o que está hecho especialmente para un fin determinado o pensado para una situación concreta.

[3] ***Big Data*** conjuntos de datos o combinaciones de conjuntos de datos cuyo tamaño (volumen), complejidad (variabilidad) y velocidad de crecimiento (velocidad) dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas convencionales

• Análisis de trama (anomalía / detección de ataque / respuesta)

•*Loging* [4]

•Alerta

• Conexión de base de datos

• Contenedor de scripts [5]

Para la prevención de intrusos, cuando se es atacado Open WIPS-NG, en la mayoría de los casos, retira la autenticación a los atacantes y avisa al administrador. Si se ataca a usuarios legítimos los banea[6] a ambos desde la red hasta que se resuelva el problema. Puede evitar que los usuarios se conecten a otras redes si se desea. (BOUVETTE, 2011)

1.3.3 FAIL2BAN

Fail2Ban puede realizar múltiples acciones siempre que se detecte una dirección IP abusiva: actualizar las reglas de cortafuegos y de las iptables[7], rechazar la dirección IP de un abusador, o cualquier acción definida por el usuario que pueda ser realizada por un script.

Fail2Ban funciona monitoreando archivos de registro para las entradas seleccionadas y ejecutando scripts basados en ellas. (VAN IMPE, 2015) Con mayor frecuencia, esto se usa para bloquear direcciones IP seleccionadas que pueden pertenecer a hosts[8] que intentan violar la seguridad del sistema. Puede prohibir cualquier dirección IP que haga demasiados intentos de inicio de sesión o realice cualquier otra acción no deseada dentro de un marco de tiempo definido por el administrador. Incluye soporte para IPv4 e IPv6. (ALEKSANDERSEN, 2016) Opcionalmente, se pueden configurar prohibiciones más largas para los abusadores "reincidentes" que siguen regresando. (BLEDSOE, 2016) Fail2Ban generalmente está configurado para desbancar un host bloqueado dentro de un cierto período, a fin de no "bloquear" ninguna conexión genuina que pueda haber sido mal configurada temporalmente. Sin embargo, un tiempo de inactividad de varios minutos suele ser suficiente para

---

4 **Login** En el ámbito de seguridad informática, ***log in*** o ***log on*** (en español **ingresar** o **entrar**) es el proceso mediante el cual se controla el acceso individual a un sistema informático mediante la identificación del usuario utilizando credenciales provistas por el usuario

5 **Script** Documento que contiene instrucciones, escritas en códigos de programación

6 **Banear** Prohibir acceso de un usuario.

7 **iptables** es un programa de utilidad de espacio de usuario que permite al administrador del sistema configurar las tablas proporcionadas por el *firewall*.

8 **Host** (Anfitrión en español) Dispositivo que conectados a una red, proveen y utilizan servicios de ellas

evitar que una conexión de red se vea inundada por conexiones maliciosas, así como para reducir la probabilidad de un ataque de diccionario exitoso.

La configuración estándar se envía con filtros para Apache, SSH[9], vsftpd[10], qmail, Postfix y Courier Mail Server. (WALLEN, 2016) Los filtros están definidos por expresiones regulares de Python, que pueden ser convenientemente personalizadas por un administrador familiarizado con las expresiones regulares. Una combinación de un filtro y una acción se conoce como "cárcel" y es lo que hace que un host malintencionado no pueda acceder a servicios de red específicos. (BLEDSOE, 2016) Además de los ejemplos que se distribuyen con el software, se puede crear una "cárcel" para cualquier proceso orientado a la red que cree un archivo de registro de acceso.

1.3.4 DENYHOSTS

DenyHosts es un script destinado a ser ejecutado por los administradores del sistema Linux para ayudar a frustrar los ataques al servidor SSH (también conocidos como ataques basados en diccionario y ataques de fuerza bruta). (ACERO, 2008) Es una herramienta de seguridad de prevención de intrusos basada en registros para servidores SSH escrita en Python. Su objetivo es 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. (ACERO, 2008)

Está restringido a conexiones que usan IPv4. No funciona con IPv6. Puede ejecutarse manualmente, como un demonio[11] o como un trabajo cron[12].

A través del análisis puede encontrar todos los intentos de inicio de sesión y filtros fallidos e intentos exitosos. El modo de sincronización permite a los demonios de DenyHosts la capacidad de compartir datos a través de un servidor centralizado para frustrar proactivamente los ataques. Se puede ejecutar desde la línea de comando, cron o como un demonio. Registra todos los intentos fallidos de inicio de sesión para el usuario y el host infractor. Por cada host que exceda un recuento de umbral, se registra como malvado. Realiza un seguimiento de cada usuario inexistente cuando falla un intento de inicio de sesión; y de cada usuario existente cuando falla un intento de inicio de sesión.

---

9 **SSH**: (Secure Shell) Protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse remotamente a un host.

10 **scftpd: (**very secure FTP daemon) Es un servidor FTP seguro y extremadamente rápido distribuido para sistemas Unix incluido GNU/Linux.

11 **Demonio: (**de sus siglas en inglés Disk And Execution MONitor), es un tipo especial de proceso **informático** que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario

12 **Cron:** administrador regular de procesos en segundo plano que ejecuta procesos o guiones a intervalos regulares.

También sigue a cada infractor e inicios de sesión sospechosos (es decir, inicios de sesión exitosos para un dispositivo que tuvo muchos errores de inicio de sesión). Atiende también al desplazamiento del archivo, para que pueda volver a analizar el mismo archivo continuamente.

Para el registro de actividad mantiene un historial de todos los usuarios, dispositivos, dupla usuario/host e inicios de sesión sospechosos encontrados, que incluye los datos y el número de intentos de inicio de sesión fallidos correspondientes. Para mayor organización y mejor lectura de la información mantiene los intentos fallidos de inicio de sesión de usuario válido e inválido en archivos separados, de modo que es fácil ver qué usuario válido está bajo ataque (lo que le daría la oportunidad de eliminar la cuenta, cambiar la contraseña o cambiar su shell[13] predeterminado. En cada ejecución, el script cargará los datos previamente guardados y los reutilizará para agregar nuevas fallas. Resuelve direcciones IP a nombres de host, si están disponibles. (ACERO, 2008)

1.3.5 OSSEC (OPEN SOURCE HIDS SECURITY)

OSSEC es un sistema de detección de intrusos (HIDS) gratuito y de código abierto. Realiza análisis de registros, verificación de integridad, monitoreo del registro de Windows, detección de rootkits[14], alertas basadas en el tiempo y respuesta activa. Proporciona detección de intrusos para la mayoría de los sistemas operativos, incluidos Linux, OpenBSD, FreeBSD, OS X, Solaris y Windows. OSSEC tiene una arquitectura centralizada y multiplataforma que permite que múltiples sistemas sean monitoreados y administrados fácilmente. Ofrece la posibilidad a través de configuración de definir las adaptaciones específicas en nuestro 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.

OSSEC posibilita monitorizaciones basadas en agentes pero también sin agentes como los componentes de red, enrutadores y cortafuegos. El soporte a la supervisión sin agentes permite que los clientes que tengan restricciones de instalación de software en los sistemas puedan cumplir también las necesidades de seguridad. 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.

---

13 **Shell** o intérprete de órdenes o intérprete de comandos es el programa informático que provee una interfaz de usuario para acceder a los servicios del sistema operativo.

14 **Rootkit** es un conjunto de software que permite un acceso de privilegio continuo a un ordenador, de forma oculta al control de los administradores.

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.

1.4 COMPARACIÓN ENTRE LOS SISTEMAS DE PREVENCIÓN DE INTRUSOS ESTUDIADOS.

Con el objetivo de seleccionar y definir con que sistema de prevención se trabaja en la investigación, se realiza una comparación entre los sistemas de prevención de intrusos estudiados. De ellos se excluye en la comparación a OPEN WIPS-NG pues este está diseñado solo con tecnologías inalámbricas, lo cual no cumple con las necesidades de Nova Servidores 7.

1.4.1 CALIFICACIÓN Y SELECCIÓN DEL SOFTWARE

La calificación y selección del software de código abierto (QSOS) es una metodología para evaluar el software de código abierto gratuito / libre. Esta metodología se publica bajo la licencia GFDL. (SEMETEYS, 2008)

El método propone cuatro etapas: definición, evaluación, calificación 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

Etapa 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.

Etapa de evaluación: consiste en realizar una caracterización del software.

Etapa de Calificación: consiste en la ponderación de los criterios definidos para realizar la comparación de las herramientas analizadas.

Para la confección de este análisis se establecen los parámetros a analizar siguientes:

Monitoreo constante: Se establece “Sí” si se monitorea la red ante posibles prevenciones constantemente, y “No” en caso de que no se haga.

Bloqueo de ataques en tiempo real: Se establece “Sí” en caso de que sea capaz de bloquear ataques en tiempo real y “No” en cualquier otro caso.

Funcionalidad para diferentes servicios: Se establece “Sí” en caso de que la herramienta pueda ser configurada para más de un servicio o protocolo, y “No” si no puede hacerlo.

Mecanismo de notificación: Se establece “Sí” en caso de que la herramienta presente mecanismos que notifiquen de los ataques, o posibles ataques y “No” en caso de que no los posea.

Se le otorga mayor peso a la Funcionalidad en Diferentes Servicios pues es de mayor interés para el cliente que el Sistema de Prevención de intrusos de Nova Servidores 7 pueda configurarse en la mayor cantidad de servicios o protocolos posible.

Fase de Selección: El objetivo de este paso es identificar el software que cumple los requisitos del usuario o, más generalmente, comparar el software de la misma familia. Para esta fase se usa la selección estricta que está basada en la eliminación directa tan pronto como el software no cumpla con los requisitos formulados en la fase anterior. En la tabla 1 se muestra la comparación y demás aspectos referentes al proceso de selección en esta fase.

Tabla 1 Comparación entre sistemas de prevención de intrusos estudiados.

Fuente: Elaboración Propia.

IPS	**Monitoreo constante**	**Bloqueo de ataques en tiempo real**	**Diferentes servicios**	**Mecanismo de notificación**
Splunk	Sí	Sí	No (Está pensado para situaciones específicas)	Sí
Fail2Ban	Sí	Sí	Sí	Sí
DenyHosts	No(Debe ser ejecutado)	Sí	No(Solo funciona para SSH)	Sí
Ossecs	Sí	Sí	Sí	No

Luego de realizado el análisis comparativa mediante el método QSOS se definió como solución Fail2Ban por cumplir con la mayor cantidad de parámetros de selección posible.

Fail2Ban es un marco de software de prevención de intrusos que protege los servidores de la computadora de ataques de fuerza bruta. Escanea los archivos de registro y prohíbe las IP que muestran signos maliciosos: demasiadas fallas de contraseña, búsqueda de vulnerabilidades, etcétera. En general, Fail2Ban se usa para actualizar las reglas del cortafuego para rechazar las direcciones IP durante un período de tiempo específico, aunque cualquier otro arbitrario puede aceptarse. También se puede configurar la acción (por ejemplo, enviar un correo electrónico). Fail2Ban viene con filtros para varios servicios (Apache, Samba, SSH, etcétera.). Puede reducir la tasa de intentos de autenticación incorrectos, sin embargo, no puede eliminar el riesgo que presenta una autenticación débil. Configure

los servicios para usar solo dos factores o mecanismos de autenticación públicos / privados si realmente desea proteger los servicios. (CARLES, 2018)

Fail2Ban es una herramienta de seguridad escrita en Python fundamental para cualquier servidor que preste servicios públicos. Su principal función es dar seguridad a un servidor del siguiente modo:

- Evitando accesos indeseados al equipo o servidor.
- Evitando ataques de fuerza bruta para que un tercero averigüe una contraseña o inhabilite el servidor.

Por un lado Fail2Ban está monitorizando las autenticaciones que una determinada IP hace a determinado/s puerto/s y servicio/s. Para ello Fail2Ban está consultando permanente los logs de autenticación del sistema como por ejemplo el /var/log/auth.log.

Si Fail2Ban detecta un número determinado de intentos de conexión fallidos bloqueará la IP que está intentando acceder a nuestro equipo o el servicio. La forma de bloquear la IP será introduciendo una regla en el cortafuegos del equipo o servidor durante un tiempo determinado que por ejemplo pueden ser 600 segundos. Una vez transcurridos los 600 segundos, o el tiempo deseado, se borrará la regla del cortafuego. Por lo tanto la IP que fue bloqueada podrá intentar conectar de nuevo al servidor. (CARLES, 2018)

1.5 TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO DE UN SISTEMA DE PREVENCIÓN DE INTRUSOS PARA NOVA SERVIDORES 7

Las tecnologías y herramientas que fueron definidas para este sistema de prevención fueron determinadas mediante un estudio realizado por el equipo de especialistas en migración de servicios telemáticos, pertenecientes al departamento Servicios Integrales de Migración de Asesoría y Soporte (SIMAYS) de CESOL. Teniendo en cuenta las características del entorno donde se utilizará la herramienta, se definió que las tecnologías a utilizar deben ser libres.

1.5.1 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 (RUMBAUCH JAMES y GRADY, 2015).

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, 2015):

- 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.

1.5.2 HERRAMIENTA PARA EL MODELADO DE UN SISTEMA DE PREVENCIÓN DE INTRUSOS PARA NOVA SERVIDORES 7

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE por sus siglas en inglés), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las misma en términos de tiempo y costo. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida del desarrollo de software, en tareas como el proceso de realizar un diseño de un proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detención de errores entre otras (MENENDEZ ALONSO, 2011)

Visual Paradigm

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación. (TSANG, 2005)

Esta herramienta se selecciona por las facilidades que brinda para capturar los requisitos correctos y transformarlos en diseños precisos, además de su capacidad para diseñar cada uno de los diagramas y/o artefactos que propone UML para el análisis y diseño, implementación, pruebas y despliegue de software.

1.5.3 LENGUAJES SELECCIONADOS PARA EL DESARROLLO DE UN SISTEMA DE PREVENCIÓN DE INTRUSOS PARA NOVA SERVIDORES 7

JavaScript

JavaScript es un lenguaje de programación que te permite realizar actividades complejas en una página web — cada vez más una página web hace más cosas que sólo mostrar información estática — como mostrar actualizaciones de contenido en el momento, interactuar con mapas, animaciones gráficas 2D/3D etcétera. — puedes estar seguro que JavaScript está involucrado. Es la tercera capa de los estándares en las tecnologías para la web, dos de las cuales son HTML y CSS (los cuales se describen a continuación). (FLANAGAN, 2020)

Se usa crear contenido nuevo y dinámico, controlar archivos de multimedia y crear imágenes animadas. JavaScript se ejecuta por la parte del cliente y no requiere ningún software de servidor. Es un lenguaje versátil por su característica de ser multi-paradigmas.

HTML

El lenguaje de marcado de hipertexto (HTML) es el lenguaje de marcado estándar para documentos diseñados para mostrarse en un navegador web. Puede ser asistido por tecnologías como hojas de estilo en cascada (CSS) y lenguajes de script como JavaScript. (FLANAGAN, 2020)

HTML es el lenguaje para describir la estructura de las páginas web y brinda los medios para que:

- Publique documentos en línea con encabezados, texto, tablas, listas, fotos.
- Recupere información en línea a través de enlaces de hipertexto, con el clic de un botón.
- Diseñe formularios para realizar transacciones con servicios remotos, para usar en la búsqueda de información, hacer reservas, ordenar productos, etc.
- Incluya hojas de cálculo y otras aplicaciones directamente en sus documentos. (MCGRATH, 2020)

Con HTML, se describe la estructura de las páginas mediante el marcado. Los elementos del idioma etiquetan piezas de contenido como "párrafo", "lista", "tabla", etcétera. El uso de sus etiquetas permite funciones de gran utilidad como la capacidad para enlazar páginas.

CSS

Hojas de estilo de cascada (CSS) es el lenguaje para describir la presentación de páginas web, incluidas los colores, el diseño y las fuentes. Permite adaptar la presentación a diferentes tipos de

dispositivos, como pantallas grandes, pantallas pequeñas o impresoras. CSS es independiente de HTML y se puede usar con cualquier lenguaje de marcado basado en XML. La separación de HTML de CSS hace que sea más fácil mantener sitios, compartir hojas de estilo entre páginas y adaptar páginas a diferentes entornos. Esto se conoce como la separación de la estructura (o contenido) de la presentación. Los elementos HTML. Este tutorial le enseñará CSS de básico ha avanzado. (MCGRATH, 2020) Es recomendada como la primera tecnología que se debe comenzar a aprender después de HTML. (FLANAGAN, 2020)

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 estética, diseño y orden a páginas ya estructuradas con HTML y así hacer más agradable la experiencia de uso de una página web.

BASH

Bash es un lenguaje de comandos y shell de Unix, además es un procesador de comandos basado en el lenguaje C que generalmente se ejecuta en una ventana de texto donde el usuario escribe comandos que causan acciones. Bash también puede leer y ejecutar comandos desde un archivo, llamado script de shell. Al igual que todos los shells de Unix, es compatible con el agrupamiento de nombres de archivo (coincidencia de comodines), tuberías, aquí documentos, sustitución de comandos, variables y estructuras de control para pruebas de condición e iteración. Las palabras clave, la sintaxis, las variables de ámbito dinámico y otras características básicas del lenguaje se copian de sh. Bash es un shell compatible con la Interfaz Portátil de Sistema Operativo par Unix (POSIX), pero con varias extensiones. (HAMILTON, 2011)

Con este lenguaje es posible la navegación a través de directorios, en este caso de Nova Servidores 7. De la misma forma permite mostrar, modificar y guardar documentos o ficheros, como son los archivos de configuración, donde se garantiza la permanencia de datos de Fail2Ban.

1.5.4 FRAMEWORKS USADOS PARA EL DESARROLLO DEL SISTEMA DE PREVENCIÓN DE INTRUSOS PARA NOVA SERVIDORES 7.

Electron

Electron es un *framework* para JavaScript que permite el desarrollo de aplicaciones enriquecidas de escritorio mediante el uso de tecnologías web. Esta desarrollado por GitHub (lo que garantiza revisiones constantes), es de código abierto y multiplataforma (funciona bajo Linux, Mac y Windows). La aplicación Electron consiste en dos partes: un proceso principal, que es un proceso Node.js y una serie de procesos de renderizado. (VUIKA, 2019)

Sin tener que manejar librerías difíciles de aprender, se puede construir una aplicación de escritorio, integrada en el Sistema Operativo. De esta manera, se sustituye el desarrollo de la interfaz gráfica utilizando las librerías tradicionales, y así el desarrollador solo debe crear interfaces gráficas en HTML, CSS y JavasScript.

VUE.js

Vue es un *framework* de código abierto, de JavaScript, el cual permite construir interfaces de usuarios y aplicaciones de una sola página de una forma muy sencilla. La curva de aprendizaje es relativamente baja si previamente se conoce JavaScript. Está pensado para ser adoptado incrementalmente y principalmente en la capa de Frontend. Vue.js presenta una arquitectura incrementalmente adaptable que se enfoca en la representación declarativa y la composición de componentes. La biblioteca principal se centra únicamente en la capa de vista. Las funciones avanzadas requeridas para aplicaciones complejas como enrutamiento, administración de estado y herramientas de compilación se ofrecen a través de bibliotecas y paquetes de soporte mantenidos oficialmente. Permite extender HTML con atributos HTML llamados directivas. Las directivas ofrecen funcionalidad a las aplicaciones HTML y vienen como directivas integradas o definidas por el usuario. (MACRAE, 2018)

Vue es un *framework* muy sencillo de aprender incluso sin previo dominio de Java Script. Por lo tanto el código es simple de mantener. Posee un sistema de enlace de datos que une las fuentes de datos del proveedor y el consumidor, sincronizándolas. Además la velocidad de ejecución es mayor que la de otros *frameworks* o bibliotecas.

1.5.5 ENTORNO DE DESARROLLO INTEGRADO

Un Entorno de desarrollo integrado (IDE) es una aplicación compuesta por un conjunto de herramientas útiles para un desarrollador. Suele consistir de un editor de código (con facilidades como resaltado de sintaxis, completamiento de código y navegación entre clases), un compilador y herramientas de automatización de la compilación, un depurador y en algunos casos un constructor de interfaz gráfica. (ZEIL, 2017)

Visual Studio

Visual Studio soporta varios lenguajes de programación. Une en un mismo entorno diseñadores visuales y presenta un conjunto de herramientas y otras tecnologías de desarrollo de software basado

en componentes para crear aplicaciones eficaces y de alto rendimiento, permitiendo a los desarrolladores crear sitios y aplicaciones web, así como otros servicios web en cualquier entorno que soporte la plataforma, diseñadores de esquemas y recursos, editores de código específicos para múltiples lenguajes, así como los compiladores y utilidades necesarias para generar aplicaciones a partir de todos esos elementos. (CHARTE, 2002)

Permite la adición de complementos para facilitar el trabajo con diferentes lenguajes y/o marcos de trabajo. Estos tienen una forma de instalación sencilla. Entre ellos destaca como de mayor interés Vue.js, una extensión que permite el trabajo con el *framework* Vue. Puede además acoplarse con Electron permitiendo, junta a la funcionalidad de Visual Studio de ejecutar el código, visualizar directamente la aplicación que se implementa con tecnologías web como una aplicación de escritorio.

1.6 METODOLOGÍA DE DESARROLLO DE SOFTWARE

Las metodologías de desarrollo de software son el marco de trabajo que colecciona un conjunto de pasos y procedimientos que se deben seguir para organizar, controlar y planear el proceso de desarrollo de un software. Surge ante la necesidad de trabajar mediante el uso de procedimientos, técnicas, herramientas y documentos durante el desarrollo de software. (PRESSMAN, 2010) Para la presente investigación se define como metodología de desarrollo de software AUP-UCI.

El Proceso Unificado Ágil (AUP, del inglés Agile Unified Process) describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles. Se decide la variación UCI de AUP pues no existe una metodología de software universal, toda metodología debe ser adaptada a las características de cada proyecto. La misma define que para el ciclo de vida de los proyectos de la UCI, de las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición), mantener la fase de Inicio, las 3 fases restantes de AUP se agrupan en una sola, que es Ejecución y se agrega la fase de Cierre.

Este trabajo de investigación se centra en la fase de 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, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

AUP propone siete disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener siete disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en tres disciplinas: Pruebas Internas, de

Liberación y Aceptación. Además incluye las tres disciplinas restantes las cuales son Gestión de la Configuración, Planeación de proyectos y Monitoreo y Control de Proyectos.

De entre estas disciplinas destaca Requisitos por tener cuatro escenarios posibles para modelar el sistema en los proyectos. De ellos para esta investigación se seleccionó el cuarto: Historias de Usuarios (HU). Estas aplican a los proyectos que hayan definido muy bien el negocio. El cliente estará siempre presente durante el desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Es recomendado en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. (RODRÍGUEZ, 2015).

1.7 CONSIDERACIONES FINALES DEL CAPÍTULO

El estudio de los conceptos asociados al dominio del problema permite una mejor comprensión del tema de investigación. La comparación entre las herramientas de prevención identificadas, a partir de un análisis de sus características y funcionalidades desemboca en la selección de Fail2Ban como Sistema de Prevención de intrusos para Nova Servidores 7; elementos decisivos para ello fue la posibilidad de configurar y trabajar con varios servicios o protocolos a la vez que brinda esta herramienta y la cantidad de información en línea que hay respecto al trabajo con la misma. Se definió el uso de ambos *frameworks*, VUE.js y Electron, que en su conjunto permiten desarrollar una aplicación de escritorio, mediante Visual Studio como IDE. El estudio del concepto de herramienta CASE para el diseño permitió la selección de Visual Paradigm. Se obtuvo una fase y un escenario de trabajo definidos, estableciendo AUP-UCI como metodología de desarrollo de software.

Capítulo 2: ANÁLISIS Y DISEÑO DEL SISTEMA DE PREVENCIÓN DE INTRUSOS PARA NOVA SERVIDORES 7

2.1 INTRODUCCIÓN

A partir de este capítulo se inicia el diseño y desarrollo de una propuesta de solución que dé respuesta al problema planteado en el capítulo anterior, aplicando la fase de Ejecución de la metodología AUP variación UCI. Además se presentan diferentes artefactos definidos por la misma. Se expone la interpretación de las necesidades referentes al sistema las cuales se muestran en Requisitos e Historias de Usuarios.

2.2 PROPUESTA DE SOLUCIÓN

Para lograr la correcta prevención de intrusos en Nova Servidores 7 se desarrolla una herramienta que permite instalar y configurar Fail2Ban, esta última permite la restricción de accesos por tiempo de expulsión, tiempo entre búsquedas y número máximo de reintentos. La herramienta mediante el acceso remoto es capaz de acceder desde cualquier computadora que use la distribución de GNU/Linux Nova a la configuración de Fail2Ban en una computadora que use el sistema Nova Servidores 7.

La herramienta mencionada debe permitir la instalación y desinstalación de Fail2Ban. Una vez instalado permite el intercambio entre protocolos para su posterior modificación de parámetros. Además debe poderse iniciar, reiniciar y detener Fail2Ban. Y también debe asegurar el guardado de los cambios realizados en la configuración.

Instalar Fail2Ban

En el sistema Debian y en distribuciones derivadas de Debian se ejecuta el siguiente comando en la terminal:

- sudo apt-get install Fail2Ban

En Fedora y distribuciones derivadas de Fedora se ejecuta el siguiente comando en la terminal:

- sudo dnf install Fail2Ban

En sistemas operativos como por ejemplo CentOS se ejecuta el siguiente comando:

- sudo yum install Fail2Ban

Si el paquete Fail2Ban no está disponible en los repositorios se instalarán siguiendo las instrucciones que se encuentran en la plataforma de desarrollo de Fail2Ban. Si además se desea que

se informe mediante un correo electrónico cada vez que se bloquee una IP o se reinicie una de las jaulas se debe instalar sendmail. (CARLES, 2018)

Para ello en Debian y distribuciones derivadas de Debian ejecutar el siguiente comando en la terminal:

- sudo apt-get install sendmail

En Fedora y distribuciones de Fedora ejecutar el siguiente comando en la terminal:

- sudo dnf install sendmail

En sistemas operativos como por ejemplo CentOS ejecutar el siguiente comando en la terminal:

- sudo yum install sendmail

Configuración sencilla de Fail2Ban

Configuración y consulta de los filtros:

La ubicación que contiene la totalidad de filtros de Fail2Ban es la siguiente:

- /etc/Fail2Ban/filter.d

Algunos de los filtros que se pueden encontrar son:

- ✓ sshd.conf: Filtro para detectar autenticaciones erróneas a un servidor SSH
- ✓ proftpd.conf: Filtro para detectar autenticaciones erróneas al servidor ftp Proftp
- ✓ exim.conf: Filtro para detectar autenticaciones a un servidor de correo Exim.

La función de los filtros es definir las expresiones regulares para detectar autenticaciones erróneas o ataques al equipo o servicio. Una vez definida una expresión regular se comprueba que esta expresión no aparezca en ninguno de los logs de autenticación de nuestros servicios. En el caso que la expresión regular aparezca en los logs se contabiliza un intento fallido de autenticación. Si se pretende usar los servicios estándares predeterminados de Fail2Ban no será necesario modificar ni crear ningún filtro.

Otra ubicación interesante para la configuración de Fail2Ban es la siguiente:

- /etc/Fail2Ban/action.d

En esta ruta se guardan la totalidad de scripts que definen diferentes tipos de acciones a aplicar cuando se detecta un intento de ataque, se arranca alguna de las jaulas, etcétera. En principio no se necesita modificar ni configurar parámetro de este apartado. Fail2Ban ya trae predefinidas multitud de acciones. (CARLES, 2018)

Archivo jail.conf

Es el archivo de configuración más importante. En este archivo es donde se activa, desactiva y configura Fail2Ban para que proteja determinados servicios. De forma genérica se puede decir que en el archivo *jail.conf* se realizan las siguientes acciones:

- Activar o desactivar la protección de Fail2Ban para un servicio determinado.
- Definir el filtro y las acciones que se aplican a cada uno de los servicios.
- Se define el puerto de funcionamiento de un servicio determinado para que Fail2Ban pueda funcionar de forma adecuada.
- Se establece el log a controlar para detectar los intentos de autenticación erróneos.
- Se establece el número de intentos fallidos que permitimos antes que se aplique la acción para bloquear los intentos de autenticación erróneos.

Antes de iniciar la configuración se recomienda duplicar el archivo de configuración *jail.conf*. Para ello se debe ejecutar el siguiente comando en la terminal:

- sudo cp /etc/Fail2Ban/jail.conf /etc/Fail2Ban/jail.local

Una vez duplicado el archivo se deshabilitará la configuración del archivo *jail.conf* y se aplicará la del archivo *jail.local*. De esta forma siempre se tiene una copia de la configuración predeterminada de Fail2Ban. Para acceder a la configuración de las reglas se ejecuta el siguiente comando en la terminal:

- sudo nano /etc/Fail2Ban/jail.local

El contenido del fichero *jail.local* contiene preconfiguraciones estándares para cada uno de los servicios que se pueden proteger. Para activar y configurar las preconfiguraciones tan solo se debe comentar, descomentar y modificar los parámetros de cada una de la secciones.

A modo de ejemplo se localiza la sección destinada a proteger un servidor SSH contra ataques DdoS[15].

[ssh-ddos]

[15] Ataque DdoS (ataque de denegación de servicio): tiene como objetivo inhabilitar un servidor, un servicio o una infraestructura. Existen diversas formas de ataque DDoS: por saturación del ancho de banda del servidor para dejarlo inaccesible, o por agotamiento de los recursos del sistema de la máquina, impidiendo así que esta responda al tráfico legítimo.

enabled = false
port = ssh
filter = sshd-ddos
logpath = /var/log/auth.log
maxretry = 6

Una vez localizada la sección se puede apreciar que la protección está desactivada porque el parámetro enabled tiene el valor false. Por lo tanto, para activar la protección se modifica el valor campo enabled de false a true.

[ssh-ddos]
enabled = true
port = ssh
filter = sshd-ddos
logpath = /var/log/auth.log
maxretry = 6

El resto de parámetros de la sección se adaptaran en función de las necesidades. El significado de los parámetros que se encuentran dentro de una sección es el siguiente:

- ✓ enabled = Con los valores true y false activamos y desactivamos la protección que ofrece Fail2Ban para un determinado servicio.
- ✓ port = Definición de los puertos en que están trabajando los servicios que queremos proteger.
- ✓ filter = Se define el nombre del filtro a aplicar para detectar intentos de autenticación fallidos. Para ver la totalidad de filtros disponibles se puede visitar la ubicación /etc/Fail2Ban/filter.d
- ✓ logpath = Definición del log a monitorizar para detectar los intentos fallidos de autenticación.
- ✓ maxretry = Número de intentos de autenticación máximos fallidos antes de aplicar una acción de bloqueo.
- ✓ action = Para definir las acciones de bloqueo que se aplicarán en cada uno de los servicios que queremos proteger. La totalidad de acciones disponibles se hallan en la ubicación /etc/Fail2Ban/action.d

- ✓ findtime = Definimos el tiempo ha transcurrir para que el contador de intentos fallidos de una determinada IP se resetee.
- ✓ bantime = Definimos el tiempo en segundos que queremos bloquear una determinada IP. Normalmente un valor de 600 segundos es más que apropiado.

Si la sección de un servicio no dispone de los parámetros mencionados en la tabla, entonces se toman los parámetros por defecto configurados en la sección [DEFAULT]. Una vez modificados la totalidad de parámetros de la sección [SSH-DDoS] se guardan los cambios y se procede a salir del fichero. (CARLES, 2018) Para que los cambios se hagan efectivos se ejecuta el siguiente comando para reiniciar Fail2Ban:

- o sudo service Fail2Ban restart

Consultar y definir la configuración general de Fail2Ban

Inicialmente es recomendable duplicar el archivo de configuración genérico de Fail2Ban. Para ello se ejecuta el siguiente comando en la terminal:

- o sudo cp /etc/Fail2Ban/Fail2Ban.conf /etc/Fail2Ban/Fail2Ban.local

Una vez duplicado el archivo se deshabilitará la configuración del archivo Fail2Ban.conf y aplicará del archivo Fail2Ban.local. De esta forma siempre se obtendrá una copia de la configuración predeterminada de Fail2Ban. Para acceder a la configuración general de Fail2Ban se ejecuta el siguiente comando en la terminal:

- o sudo nano /etc/Fail2Ban/Fail2Ban.local

Dentro de este fichero se pueden modificar los siguientes parámetros:

- ✓ loglevel = Según la versión de Fail2Ban disponen de diversos valores. Estos valores determinan el nivel de detalle que tendrán los logs de Fail2Ban. Por defecto el nivel de detalle es el 3 o el INFO.
- ✓ logtarget = Se determina la ubicación del archivo que almacenará los logs de Fail2Ban. La ubicación por defecto es la /var/log/Fail2Ban.log
- ✓ socket = Se debe especificar la ruta del archivo que se usará para que las partes que actúan como cliente y servidor de Fail2Ban se puedan comunicar. La ruta por defecto es /var/run/Fail2Ban/Fail2Ban.sock y no es necesario modificarla.

- ✓ pidfile = Ruta del fichero que se usa para almacenar el número de proceso del servidor Fail2Ban. Por defecto el fichero es el /var/run/Fail2Ban/Fail2Ban.pid
- ✓ dbfile = A partir de la versión 0.9 Fail2Ban guarda los datos de bloqueo en una base de datos sqlite. En este apartado se debe indicar la ruta del archivo que contendrá la base de datos. La ruta por defecto es la /var/lib/Fail2Ban/Fail2Ban.sqlite3. Si únicamente se pretende almacenar los logs de bloqueo en memoria, el valor de dbfile tendrá que ser :memory:. De esta forma los logs de bloqueo solo se guardarán mientras Fail2Ban esté activo. Si no se desea almacenar ninguna base de datos fijaremos el valor de dbfile en None
- ✓ dbpurgeage = se debe indicar el periodo de tiempo que se quiere almacenar las IP bloqueada en nuestra base de datos sqlite.

El fichero de configuración genérica es válido para 99,9% de los casos. Por lo tanto rara vez tendremos que modificar ningún parámetro en este fichero. (CARLES, 2018) Fail2Ban viene configurado de forma predeterminada para poder usarse en multitud de servicios como por ejemplo los siguientes:

1. SSH
2. Servidores web como por ejemplo Lighttpd, Nginx y Apache
3. Servidores ftp como por ejemplo vsftpd.
4. En servidores de email como por ejemplo Postfix, Exim, Squirrelmail, Courier, Dovecot, SASL, etc.
5. Servicios de proxy como Squid.
6. Otros servicios como Asterisk, FreeSWITCH, Drupal, WordPress, etcétera.

2.2.1 DESCRIPCIÓN DEL NEGOCIO

Para describir la propuesta de solución se definió hacerlo mediante un modelo conceptual.

Un modelo conceptual muestra los conceptos presentes en el dominio del problema. Un concepto para este caso, es un objeto del mundo real, es decir, es la representación de cosas del mundo real y no de componentes de software. En él no se definen operaciones (o métodos). En este modelo se pueden mostrar los conceptos, los atributos de los conceptos (opcionalmente) y la relación o asociación entre ellos. Para su diseño es necesario analizar la descripción del sistema y los requerimientos. (ISLAS, 2014).

A continuación, la Ilustración 1 muestra el modelo conceptual diseñado.

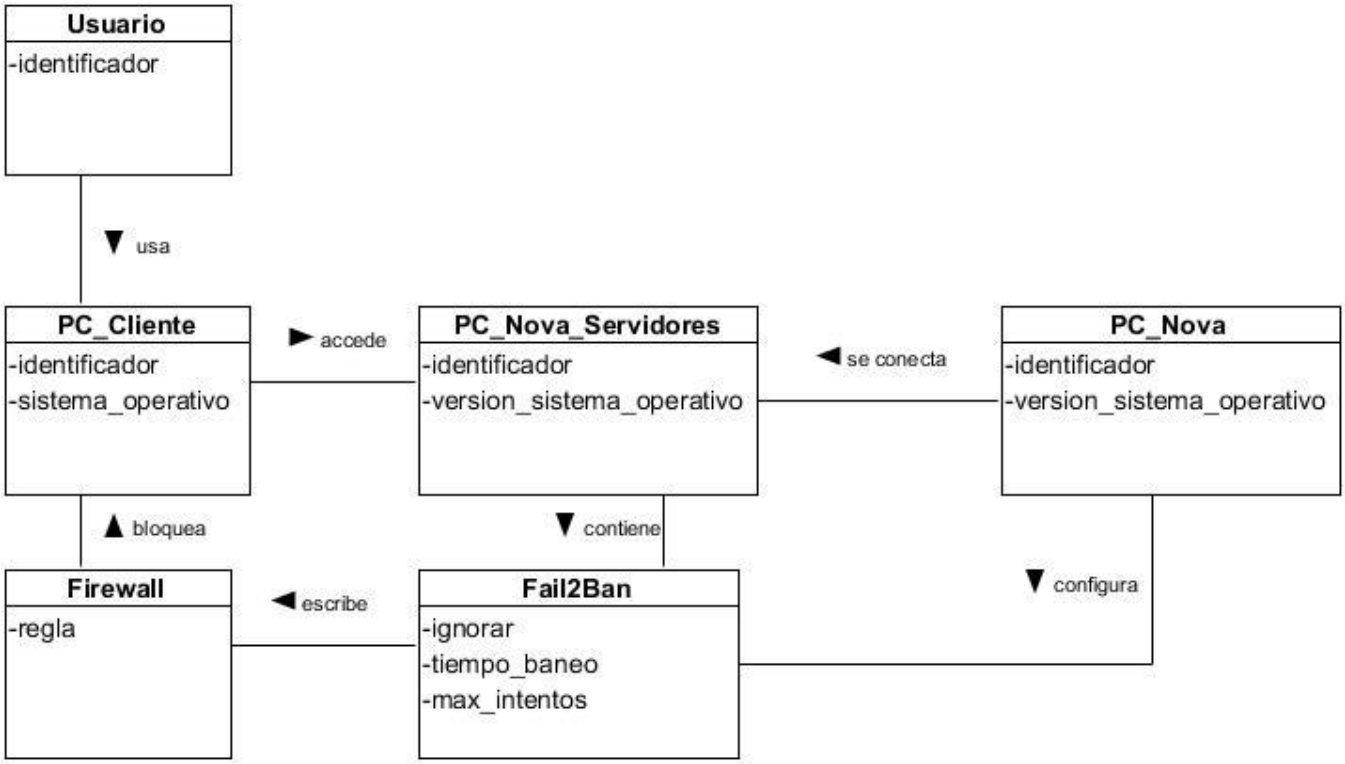


Ilustración 1 Modelo conceptual del Sistema de prevención de intrusos para Nova Servidores.

Fuente: Elaboración propia

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

Usuario: persona que intenta acceder a **PC_Nova_Servidores** con una **PC_cliente.**

PC_cliente: computadora que se conecta a Nova Servidores 7.

PC_Nova_Servidores: computadora que recibe la conexión de la **PC_cliente,** y que tiene el sistema Nova Servidores 7, el cual permite el alojamiento de servidores.

PC_Nova: Computadora que se conecta a **PC_Nova_Servidores**, y que mediante la administración remota trabaja con **Fail2Ban**.

Fail2Ban: Herramienta de prevención de intrusos que se encarga del bloqueo de acceso de los clientes al servidor a partir de reglas, y que es administrada remotamente por **PC_Nova** en **PC_Nova-_Servidores** (debe estar instalada en esta última).

Firewall: en un sistema o una red permite bloquear el acceso no autorizado (del **Usuario** en el caso de este modelo conceptual), permitiendo al mismo tiempo comunicaciones autorizadas mediante reglas. **Fail2Ban** puede actualizar estas reglas.

2.3 REQUISITOS

Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema. (SOMMERVILLE, 2011).

2.3.1 FUENTES PARA LA OBTENCIÓN DE REQUISITOS

Las fuentes de obtención de requisitos utilizadas fueron:

- Propuesta de solución del Sistema de Prevención de Intrusos para Nova Servidores 7
- Análisis de las herramientas existentes (Ver epígrafe 1.3).
- Especialistas de CESOL.

2.3.2 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, 2010).

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 13 requisitos funcionales y 7 requisitos no funcionales, como se aprecia en el Anexo 4.

2.3.3 REQUISITOS FUNCIONALES (RF)

Los requisitos funcionales definen una función de un sistema o sus componentes, donde una función es descrita como una especificación del comportamiento entre datos de entrada y salida (LAPLANTE, 2009). A continuación se presenta la tabla de requisitos funcionales para la propuesta de solución.

Tabla 2 Requisitos funcionales. Fuente: Elaboración Propia.

Fuente: Elaboración Propia.

No.	Nombre	Descripción	Prioridad
RF1.	Instalar Fail2Ban	Permite la instalación de la herramienta Fail2Ban. Debe poseer una opción “Instalar”. Al finalizar la instalación deberá mostrar un mensaje de confirmación. De lo contrario lanza una notificación de error	Alta
RF2.	Desinstalar Fail2Ban	Permite la desinstalación de la herramienta Fail2Ban. Debe poseer una opción “instalar”. Al finalizar la instalación deberá mostrar un mensaje de confirmación. De lo contrario lanza una notificación de error.	Media
RF3.	Iniciar Fail2Ban	Permite al usuario iniciar la herramienta. El sistema debe tener la opción “iniciar”	Media
RF4.	Detener Fail2Ban	Permite al usuario apagar la herramienta. El sistema debe tener la opción “apagar”	Media
RF5.	Reiniciar Fail2Ban	Permite al usuario reiniciar la herramienta. El sistema debe tener la opción “reiniciar”	Media
RF6	Configurar protección para Apache	Permite configurar la protección para Apache, modificando las reglas de Fail2Ban.	Alta
RF7.	Configurar protección para Samba	Permite configurar la protección para Samba	Alta

		modificando las reglas de Fail2Ban.	
RF8.	Configurar protección para protocolo SSH	Permite configurar la protección para SSH modificando las reglas de Fail2Ban.	Alta
RF9.	Habilitar servicios de Fail2Ban	El sistema debe permitir habilitar los servicios de Fail2Ban.	Alta
RF10.	Deshabilitar servicios de Fail2Ban	El sistema debe permitir deshabilitar los servicios de Fail2Ban.	Alta
RF11.	Mostrar configuración de Fail2Ban	Permite mostrar la configuración de Fail2Ban.	Media
RF12.	Modificar configuración de Fail2Ban	Permite al usuario modificar la configuración de Fail2Ban	Alta
RF13.	Guardar Cambios	Permite guardar los cambios realizados con la herramienta. Al finalizar la deberá mostrar un mensaje de confirmación. De lo contrario lanza una notificación de error.	Alta

2.3.3 REQUISITOS NO FUNCIONALES (RNF)

Un RNF es un requisito que especifica criterios que pueden usarse para juzgar el funcionamiento de un sistema, en lugar de comportamientos específicos. Se contrastan con los requisitos funcionales que definen el comportamiento o las funciones específicas. El plan para implementar los requisitos funcionales se detalla en el diseño del sistema. (CHEN, BABAR Y NUSEIBEH, 2013)

A continuación se presentan los requisitos no funcionales correspondientes a la propuesta de solución.

Tabla 3 Listado de Requisitos no Funcionales.

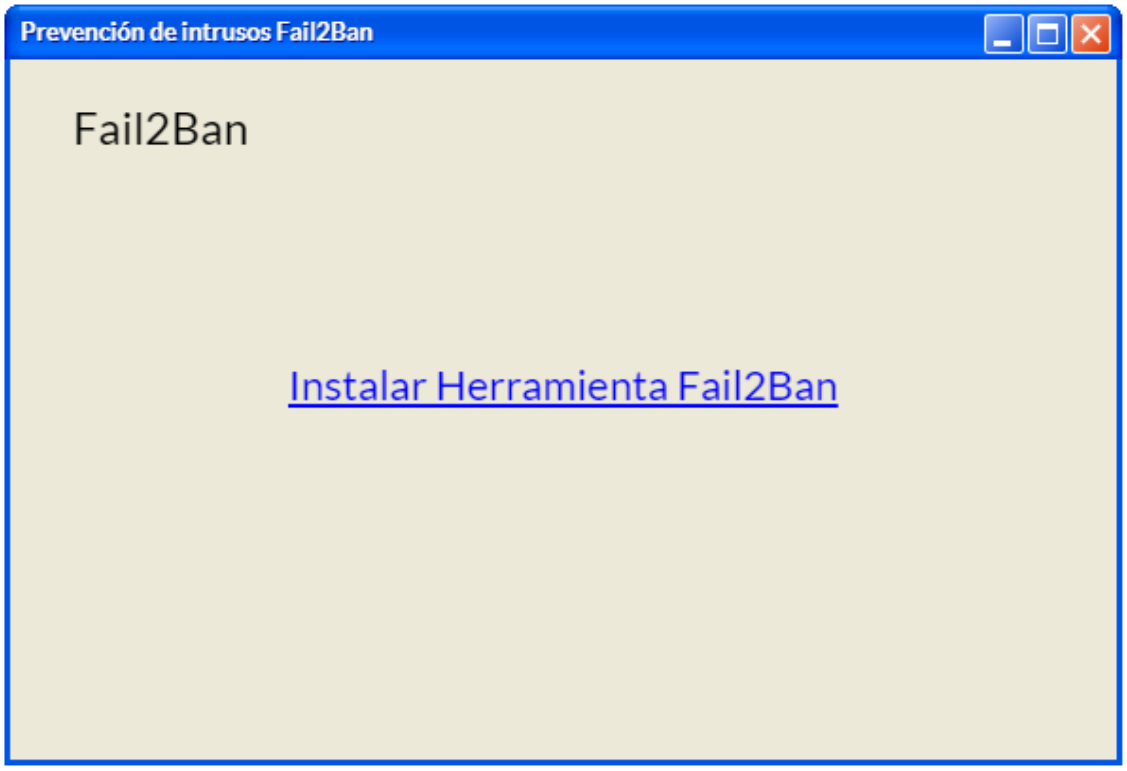
Fuente: Elaboración Propia.

No.	Requisito no funcional	Descripción
RnF1	Software	La herramienta debe funcionar sobre Nova GNU/Linux.
RnF2	Software	Debe poderse administrar y configurar de forma remota la herramienta de prevención de intrusos desde una PC con Nova hacia una PC con Nova Servidores 7
RnF3	Restricciones en el diseño y la implementación:	El framework utilizado para el desarrollo de la propuesta de solución es VUE.js.
RnF4	Restricciones en el diseño y la implementación:	La interfaz visual debe mantener el estilo y diseño de la distribución de GNU/Linux Nova.
RnF5	Mantenimiento	El sistema debe ajustarse a los estándares de configuración establecidos
RnF6	Soporte	Debe poderse acceder al sistema como una herramienta nativa de Nova.
RnF7	Almacenamiento	La permanencia de datos debe ser garantizada mediante archivos de configuración propios del sistema de prevención seleccionado.

2.3.4 DESCRIPCIÓN DE LOS REQUISITOS MEDIANTE HISTORIAS DE USUARIO (HU)

Las HU son aquellas que representan las funcionalidades que el cliente quiere que estén presentes en la solución. (DIMITRIJEVIC, JOVANOVIC Y DEVEDZIC, 2015) Durante la fase Planificación-Definición se establecieron 13 HU, las cuales responden a cada una de las funcionalidades del sistema. A continuación se describe una HU definida en la propuesta de solución como ejemplo, el resto de HU se presentan en los Anexos.

Tabla 4 Historia de Usuario: Instalar Fail2Ban

Número: HU-1	**Requisito: Instalar Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración Asignada: 1**
Prioridad: Alta	**Tiempo Estimado: 72 horas**
Riesgo en Desarrollo: No Aplica	**Tiempo Real: -**
Descripción: Permite la instalación de la herramienta Fail2Ban. Debe poseer una opción "Instalar". Al finalizar la instalación deberá mostrar un mensaje de confirmación. De lo contrario lanza una notificación de error.	
Observaciones: El archivo *exports* propio de Fail2Ban deberá guardarse en /etc. 	

2.3.5 VALIDACIÓN DE LOS REQUISITOS OBTENIDOS

La validación de los requisitos tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo de tiempo o presupuesto que eso conlleva. Los parámetros a validar en los requisitos son:

- Validez: no basta con preguntar a un usuario, todos los potenciales usuarios pueden tener puntos de vista distintos y necesitar otros requisitos.
- Consistencia: no debe haber contradicciones entre unos requisitos y otros.
- Completitud: deben estar todos los requisitos. Esto es imposible en un desarrollo iterativo, pero, al menos, deben estar disponibles todos los requisitos de la iteración en curso.
- Realismo: se pueden implementar con la tecnología actual.
- Verificabilidad: tiene que existir alguna forma de comprobar que cada requisito se cumple. (SOMMERVILLE, 2011)

A continuación se presentan las técnicas de validación de requisitos de la propuesta de solución con sus respectivas descripciones.

Reviews o walk-throughs (revisión)

El objetivo de la revisión formal es descubrir errores en la función, la lógica o la implementación de cualquier producto del software, verificar que satisface sus especificaciones, que se ajusta a los estándares establecidos, señalando las posibles desviaciones detectadas. Así como descubrir los errores en funcionamiento, lógica o implementación de cualquier representación del software, verificar el cumplimiento de sus requisitos y obtener uniformidad en su desarrollo. (PRESSMAN, 2010)

La revisión para estos requisitos fue realizada por especialistas de CESOL.

Prototipos

Los prototipos son modelos que sin tener la totalidad de la funcionalidad del sistema, permitan al usuario tener una idea de la estructura de la interfaz del sistema con el usuario. (ESCALONA Y KOCH, 2002)

La revisión para estos requisitos fue realizada por especialistas de CESOL.

2.4 ARQUITECTURA DEL SISTEMA

La arquitectura de un software muestra la estructura, funcionamiento e interacción entre las partes del software; especifica, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Para la implementación del software se

escoge la arquitectura n-capas. Las arquitecturas en capas constituyen uno de los estilos de programación que aparecen con mayor frecuencia mencionados en las literaturas informáticas. El estilo de programación en capas constituye una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior, y se sirve de las prestaciones que le brinda la inmediata inferior. (CLEMENTS, 2010)

El sistema se divide en las siguientes dos capas:

Capa presentación: es la capa con la que interactúa el usuario mediante interfaces recibiendo información, también captura los datos que este introduce en la aplicación. Esta capa se comunica únicamente con la Capa de Negocio enviando peticiones a la misma. También es conocida como interfaz gráfica y tiene característica de ser extensible y fácil de usar para el usuario. En el desarrollo de la aplicación esta capa está compuesta por todas las vistas correspondientes a la misma.

Capa del negocio: es donde residen los programas presentes durante la ejecución de la aplicación. Esta capa se comunica con la de presentación, para recibir las peticiones y presentar los resultados e interactúa con los datos de configuración para modificarlos y/o mostrarlos.

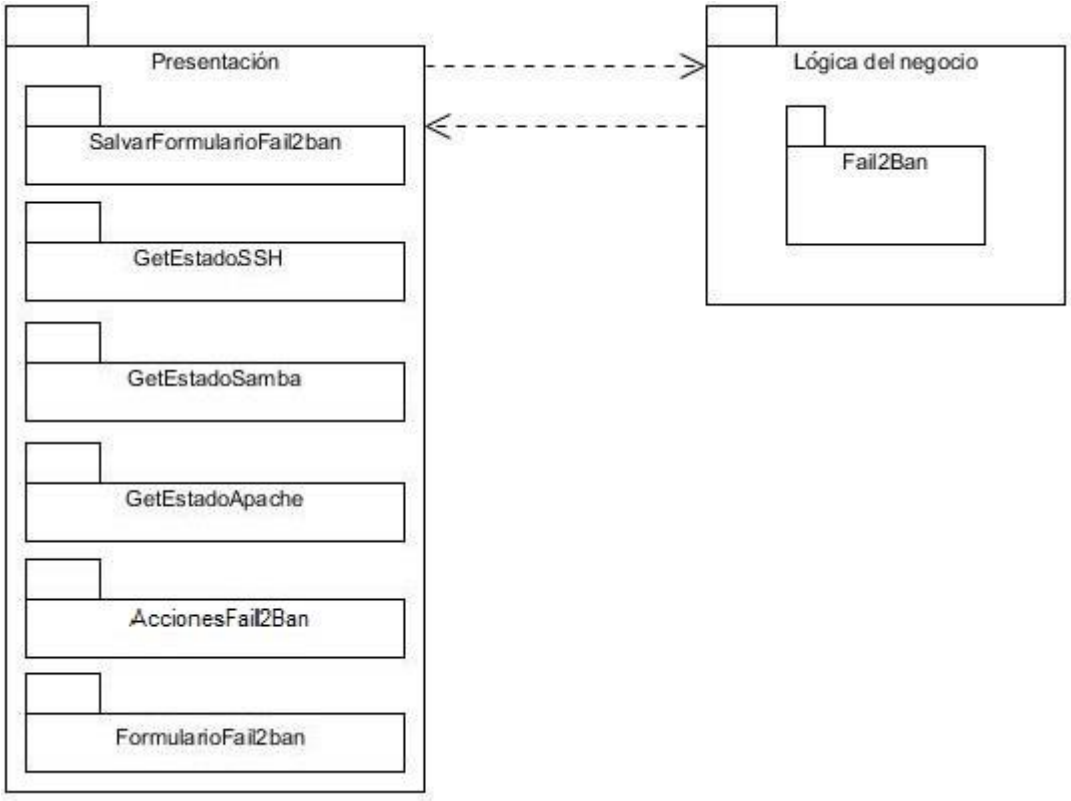


Ilustración 2 Modelo arquitectónico de la propuesta de solución. Fuente: Elaboración Propia.

FormularioFail2Ban: Interfaz que muestra los formularios para la configuración de Fail2Ban para los diferentes servicios de Nova Servidores 7.

AccionesFail2Ban: Muestra las acciones de iniciar, reiniciar y detener Fail2ban.

GetEstadoApache: Muestra la configuración anterior de Fail2Ban para Apache.

GetEstadoSamba: Muestra la configuración anterior de Fail2Ban para Samba.

GetEstadoSSH: Muestra la configuración anterior de Fail2Ban para SSH.

SalvarFormularioFail2Ban: Pantalla de salvado de la configuración establecida.

Fail2Ban: Hace referencia a los servicios y herramientas que son usados dentro del sistema de prevención de intrusos, que permite configurar fail2ban en Nova Servidores y posteriormente guardar dicha configuración, así como mostrarla. También permite realizar acciones como iniciar, reiniciar o detener el servicio.

2.5 MODELO DE DISEÑO

Este producto de trabajo es un modelo de objeto que sirve como una abstracción del modelo de implementación y el código fuente, 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 MARTÍNEZ 2015).

2.5.1 DIAGRAMA DE CLASES

Los diagramas de clases 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. (PRESSMAN, 2010)

Para el diseño de la propuesta de solución se obtuvo un diagrama de clases que se presenta en la figura siguiente.

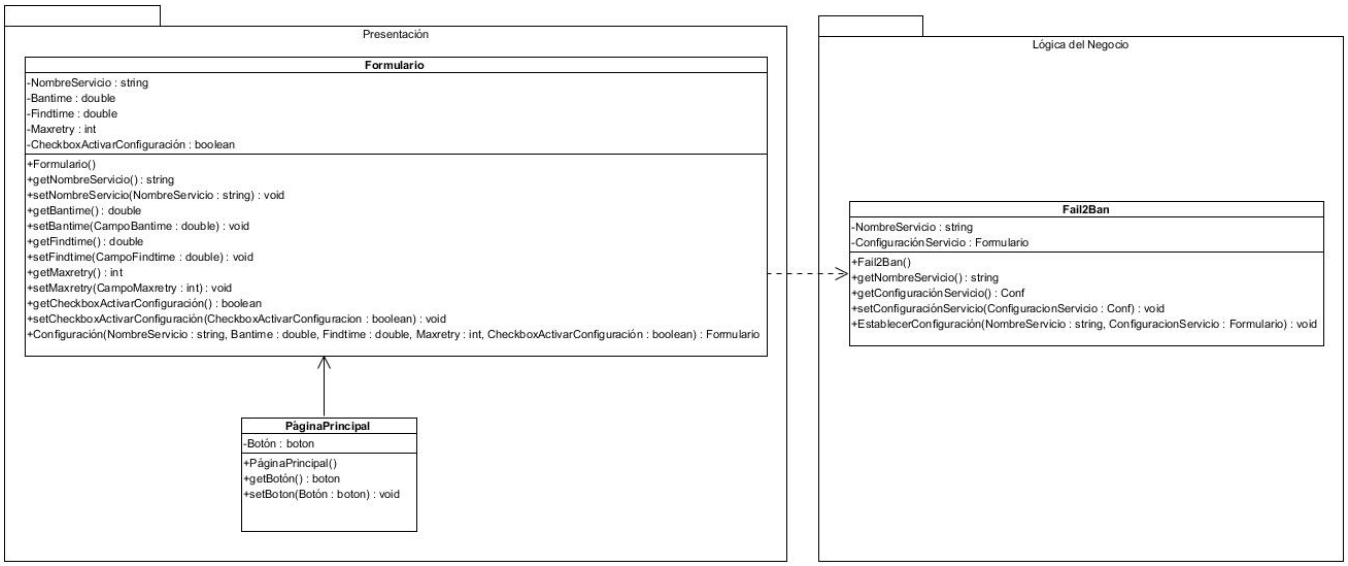


Ilustración 3 Diagrama de clases del Sistema de Prevención de Intrusos para Nova Servidores 7.

Fuente: Elaboración Propia.

PáginaPrincipal: es la vista de bienvenida que direcciona mediante un botón a los formularios de configuración.

Formulario: vista de los formularios del sistema que incluye el nombre del servicio para el cual se desea configurar la protección de Fail2Ban, así como los respectivos parámetros bantime (tiempo de expulsión), findtime (tiempo entre intentos) y maxretry (máxima cantidad de intentos), y si se desea que la configuración se guarde o no mediante una caja de selección. Esta clase muestra y toma datos que son o serán manejados por el sistema de prevención de intrusos.

Fail2Ban: en esta se manejan los datos adquiridos Formulario, realizando acciones con ellos: en dependencia de cual sea el nombre del servicio establecido y teniendo en cuenta si el usuario estableció que la protección de este debía configurarse, se modifican los tres parámetros antes mencionados en los archivos de configuración de Fail2Ban.

2.5.2 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, 2010). Para la implementación de la propuesta de solución fueron aplicados los siguientes patrones de diseño:

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. El patrón GOF usado en la solución propuesta fue Prototype.

Patrón Prototype

Es un patrón de diseño creacional, que tiene como objetivo crear a partir de un modelo. Especificar el tipo de objetos que se crearán mediante una instancia prototípica, y crear nuevos elemento copiando este prototipo. Su concepto es hacer una copia exacta de otro objeto, esto en lugar de crear uno nuevo, permite crear los objetos prediseñados sin la necesidad de conocer los detalles de la creación. (ITURRALDE, 2016)

Un ejemplo específico de ello es la creación de objetos de configuración que poseen las mismas características. Una configuración siempre tendrá un bantime, un findtime y un maxretry, acompañados de un nombre de servicio. Estos pueden tomar diferentes valores, pero los atributos siempre serán los mismos.

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.

Patrón 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. (ITURRALDE, 2016)

Este se evidencia en el método Configuración de la clase Formulario. Donde quiera que este método sea llamado, por ejemplo en Fail2Ban para obtener la configuración para un servicio, se generará un nuevo objeto de tipo Formulario que guarda en sí dicha configuración, este objeto de tipo Formulario se crea gracias al método creador de la clase Formulario.

Patrón 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. (ITURRALDE, 2016)

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 manejar todos los atributos de configuración.

2.6 CONSIDERACIONES FINALES DEL CAPÍTULO

Para la propuesta de solución, se obtuvieron 13 requisitos funcionales y 7 no funcionales mediante la técnica de entrevista. A partir de ello se realizaron 13 historias de usuarios lo cual permitió un mayor control y distribución acerca de las funcionalidades que debe cumplir la aplicación, y se produjeron el modelo conceptual y el diagrama de despliegue como lo establece la metodología propuesta, lo que permitió tener una guía de la situación real de la propuesta de solución. El uso de una arquitectura n-capas y el empleo de patrones de los diseño, garantiza la obtención de una solución de software con poca dependencia entre clases, flexible al mantenimiento y a la aceptación de cambios.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DE SISTEMA DE PREVENCIÓN DE INTRUSOS PARA NOVA SERVIDORES 7

3.1 INTRODUCCIÓN

En este capítulo están presentes los diferentes artefactos que se utilizan para la implementación y pruebas del sistema de prevención de intrusos para Nova Servidores 7, así como el estándar de codificación que se deben seguir para lograr el entendimiento y organización del código. Se especifican, de los tipos de pruebas existentes, los que serán empleados para darle validez a los requisitos funcionales y garantizar el óptimo funcionamiento de la aplicación.

3.2 MODELO DE IMPLEMENTACIÓN

El modelo de implementación lo comprende 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. (PRESSMAN, 2010)

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. (PRESSMAN, 2010)

La siguiente figura representa el diagrama de componentes correspondiente al Sistema de Prevención de Intrusos para Nova Servidores 7.

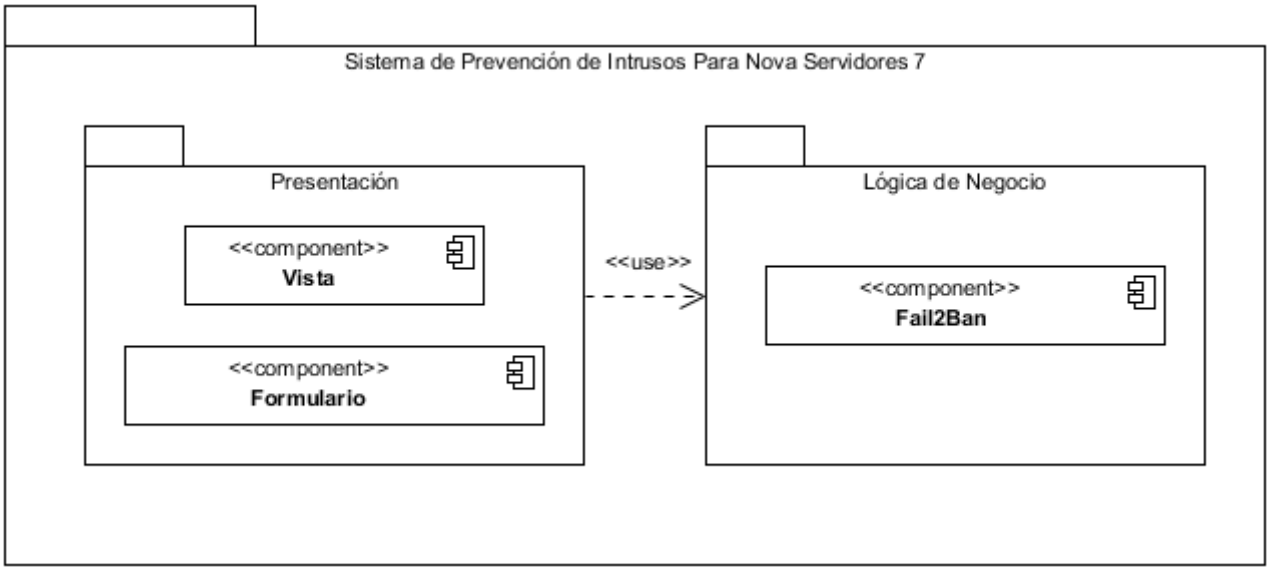


Ilustración 4 Diagrama de componentes del Sistema de Prevención de Intrusos para Nova Servidores 7.

Fuente: Elaboración Propia.

Descripción de los componentes del sistema

Presentación: paquete que agrupa a todos los componentes que interactúan con el paquete de Lógica del Negocio; estos componentes permiten trabajar con algunas utilidades sobre los formularios.

Lógica del Negocio: paquete que agrupa las clases que representan el dominio de los archivos de configuración.

3.3 ESTÁNDARES DE CODIFICACIÓN

Un estándar de codificación es un estilo que se predefine en un proyecto de software para establecer una unicidad en la forma del desarrollo de dicho proyecto. (VAN ROSSUM, WARSAW Y COGHLAN, 2013)

Para esta investigación y consiguiente propuesta de solución se establecen los siguientes:

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).

Máxima longitud de las líneas

- Todas las líneas deben estar limitadas a un máximo de 50 caracteres.
- Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas.

Importación

- Las importaciones deben estar en líneas separadas.

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 (“==”, “<”, “>”, “>=”, “<=”, “! =”, “<>”).
    4. Expresiones lógicas.
- 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.

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 nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúscula).

A continuación se muestra una porción de código donde se evidencian la aplicación de algunos estándares de codificación.

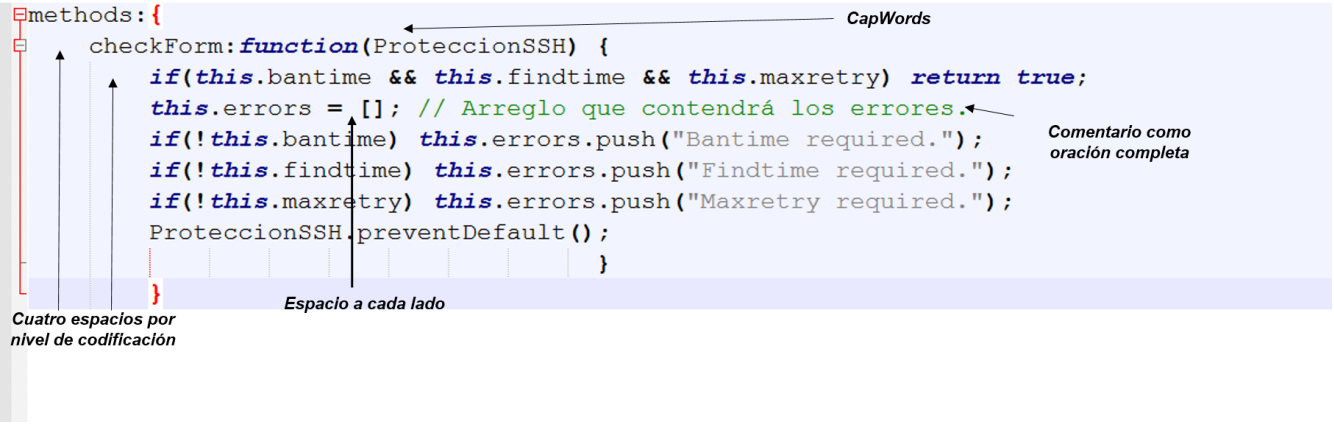


Ilustración 5 Aplicación de los estándares de codificación. Fuente: Elaboración propia.

3.4 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, 2010).

Una vez que se implementa el código del sistema es necesario probarlo para encontrar y remediar 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 sistema para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para Nova Servidores 7. 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 fueron construidos.

3.5 APLICACIÓN DE LAS PRUEBAS DE SOFTWARE

Se presenta una descripción de 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.

3.5.1 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, 2015). 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, 2010).

```
methods:{
    checkForm:function(ProteccionSSH) {
        if(this.bantime && this.findtime && this.maxretry) return true; //1
        this.errors = [];//2
        if(!this.bantime) this.errors.push("Bantime required.");//3
        if(!this.findtime) this.errors.push("Findtime required.");//4
        if(!this.maxretry) this.errors.push("Maxretry required.");//5
        ProteccionSSH.preventDefault();//6
                                    }
        }
```

Ilustración 6 Código de implementación usado para aplicar el método Camino Básico. Fuente: Elaboración Propia.

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

1) **Confeccionar el grafo de flujo**: usando el código de la Ilustración 5 se realiza la representación del grafo de flujo, el cual está compuesto por los siguientes elementos:

a) Nodos: son círculos que representan una o más sentencias procedimentales.

b) Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo

c) Regiones: son las áreas delimitadas por aristas y nodos.

A continuación se muestra el grafo obtenido.

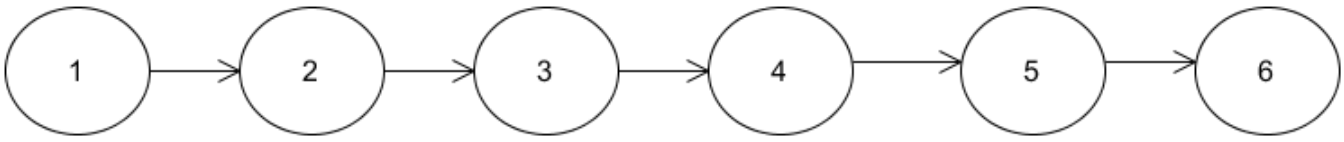


Ilustración 7 Grafo resultante del método caja blanca camino básico. Fuente: Elaboración propia

2) **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:

a) **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)= 5-6+2

V (G)= 1

b) **Variante 2:**

Teniendo en cuenta el número de regiones que presenta el grafo de flujo obtenido, se define la cantidad de regiones.

c) **Variante 3:**

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

V(G) = 0+1

V(G) = 1

3) **Determinar un conjunto básico de caminos linealmente independientes**: una vez aplicadas las tres variantes para calcular la complejidad ciclomática del método checkForm:function(ProteccionSSH), 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: 1-2-3-4-5-6

4) **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. A continuación, en la tabla se muestra el caso de prueba (CP) diseñado para el camino básico obtenido.

Tabla 5 Diseño de casos de prueba.

Fuente: Elaboración Propia.

Diseño de caso de prueba	
Descripción	Lista la cantidad de intentos de inicios de sesión que se hayan realizado a la plataforma Nova Servidores 7.
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.

3.5.2 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, 2010). 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 sistema 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

3.5.3 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, 2010). 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 sistema desarrollado y nova servidores 7.

Partiendo que el sistema 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.

3.5.4 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.

3.5.5 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, 2011).

Tabla 6 Caso de prueba de aceptación para la historia de usuario Configurar protección para Apache. Fuente: Elaboración Propia.

Caso de prueba de aceptación
Nombre de la historia de usuario: Configurar protección para Apache
Nombre de la persona que realiza la prueba: Hanny Valdés Hernández
Descripción de la prueba: El sistema permite modificar los archivos de configuración de Fail2Ban en lo que respecta a la protección de Apache, en Nova Servidores 7.
Condiciones de ejecución: El usuario debe ingresar en el formulario la configuración deseada.
Entrada/Pasos de ejecución: 1. Acceder al Sistema de Prevención de Intrusos. 2. Introducir los parámetros de configuración.
Resultado esperado: El sistema guarda en los archivos de configuración los parámetros modificados.

3.6 DESPLIEGUE

Son todas las actividades que hacen que un sistema de software esté disponible para su uso. Proveen la vista de implementación del sistema. Los diagramas de despliegue muestran la configuración en funcionamiento del sistema incluyendo su software y su hardware, describen la topología del sistema la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos y representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red. (PRESSMAN, 2010)

A continuación se presenta el diagrama de despliegue para el Sistema de Prevención de Intrusos para Nova Servidores 7.

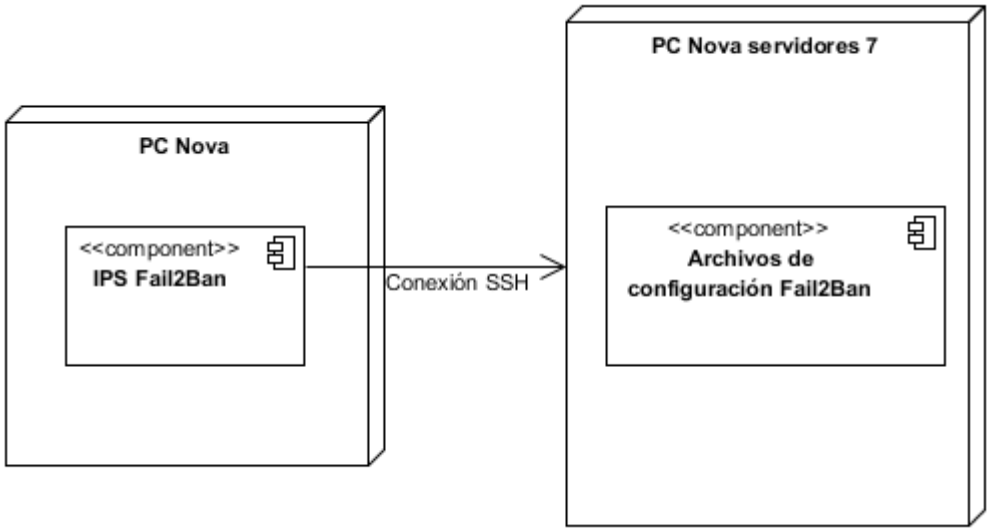


Ilustración 8 Diagrama de despliegue para el Sistema de Prevención de Intrusos para Nova Servidores 7. Fuente: Elaboración Propia

3.7 Validación 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 3 y cuya relación el encuestado desconoce.

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 7 Cuadro Lógico de IADOV Fuente: Elaboración Propia

4. 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 la prevención de intrusos en Nova Servidores 7?								
	No			**No sé**			**Sí**		
	3. ¿Considera usted factible la implementación de un sistema que prevenga de los intrusos en Nova Servidores 7?								
	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

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 deben estar respondidas 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”. La encuesta fue aplicada a 7 especialistas.

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 o 6; 4; 5 y donde N representa la cantidad total de encuestados (FERNÁNDEZ DE CASTRO, 2014)

Resultados obtenidos

Los resultados obtenidos de la aplicación de la encuesta se presentan en la Tabla 8.

Tabla 8: Resultados obtenidos de los encuestados. Fuente: Elaboración propia.

Categorías grupales de satisfacción	N = 7	Escala
Clara satisfacción	4	A
Más satisfecho que insatisfecho	3	B
No definido	0	C
Más insatisfecho que satisfecho	0	D
Clara insatisfacción	0	E
Contradictorio	0	C

2. Cálculo del ISG
ISG=A (+1) +B (+0.5)/N
ISG= (4(+1) +3(+0.5))/7=0.92

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 (0.9) y cualitativamente en los criterios emitidos en el centro de desarrollo CESOL, lo que refleja la aceptación de la propuesta, y el reconocimiento a su utilidad.

3.8 CONSIDERACIONES FINALES DEL CAPÍTULO

A partir del modelado de la implementación se obtuvo un diagrama de componentes el cual permite estructurar el modelo y definir sus componentes. Se logra determinar unicidad para la forma de

desarrollo de la propuesta de solución mediante la definición de los estándares de codificación. Con la implementación de las funcionalidades que quedan pendientes se podrá administrar y configurar los servicios de Fail2Ban. El diseño de las pruebas unitarias, integración, funcionales, seguridad y aceptación permitió crear un mecanismo para detectar errores que puedan afectar dichas funcionalidades. Se evaluó el objetivo general de la investigación y se pudo definir como factible para su uso mediante la aplicación de la técnica de IADOV.

CONCLUSIONES

Se concluye de la presente investigación:

- El análisis de los referentes teóricos y de las herramientas informáticas que implementan la prevención de intrusos estudiados evidenció la necesidad de desarrollar un Sistema que permitiera prevenir de los posibles intrusos a Nova Servidores 7.
- La selección de herramientas, lenguajes y tecnologías permitió la implementación del sistema de prevención de intrusos (IPS) para la puesta en funcionamiento de mecanismos de bloqueo a conexiones que pudieran ser maliciosas en Nova Servidores 7.
- Tras el inicio del proceso de implementación quedan pendientes las funcionalidades referentes a la administración y configuración de los servicios de Fail2Ban.
- Las pruebas diseñadas proporcionan un mecanismo para detectar las posibles deficiencias presentes en las funcionalidades implementadas y pendientes del sistema, transitando a su corrección para lograr un producto más seguro y funcional.
- La aplicación de la técnica de *IADOV* para la validación del objetivo general de la investigación permitió la evaluación satisfactoria del sistema para la implementación de mecanismos que prevengan la prevención de intrusos en Nova Servidores 7.

RECOMENDACIONES

Se recomienda

- Concluir la implementación de las funcionalidades pendientes, referentes a la administración y configuración de los servicios de Fail2Ban.
- Una vez completada dicha implementación, culminar las pruebas funcionales y los casos de prueba.

BIBLIOGRAFÍA

1. **ALEKSANDERSEN, DANIEL.** *Ipv6 support finally arrive in Fail2Ban.* 2016
2. ***BACA URBINA, GABRIEL.*** *Introducción a la seguridad informática (Ed. EBook).* 2016 (Disponible en https://books.google.com.cu/)
3. **BELL, DONALD.** *UML basics: the component diagram.* 2004 (Disponible en https://www.fing.edu.uy/inco/cursos/ingsoft/iis/files/uml-component%20diagram.pdf)
4. **BLEDSOE, GREG.** *Server hardening | linux journal.* 2016
5. **BOYLES, TIM.** *CCNA security study guide.* 2010.
6. **CARLES, JOAN.** *Instalar configurar y usar Fail2Ban para evitar ataques de fuerza bruta.* 2018 (Disponible en https://geekland.eu/instalar-configurar-y-usar-Fail2Ban-para-evitar-ataques-de-fuerza-bruta)
7. **CHARTE, FRANCISCO.** *Programación Microsoft Visual Studio.* 2015 (Disponible en https://eldeportedealbacete.es)
8. **CHEN, LIANPING; BABAR, MUHAMMAD A.; NUSEIBEH, BASHAR.** *Characterizing architecturally significant requirements.* 2013 (Disponible en https://ieeexplore.ieee.org/abstract/document/6365165/)
9. **CLEMENTS, PAUL.** *Documenting software architectures: views and beyond, segunda edición.* 2010 (Disponible en https://www.researchgate.net/publication/234787962_documenting_software_architectures_views_and_beyond)
10. **DIMITRIJEVIC, SONJA; JOVANOVIC, JELENA; DEVEDZIC, VLADAN.** *A comparative study of software tools for user story management.* 2015 (Disponible en https://www.sciencedirect.com/science/article/abs/pii/s0950584914001293)
11. **ERICKSSON, ULF.** *The difference between functional and non-functional requirements.* 2015 (Disponible en https://reqtest.com/requirements-blog/understanding-the-difference-between-functional-and-non-functional-requirements/)
12. **ESCALONA, MARÍA JOSÉ; KOCH, NORA.** *Ingeniería de requisitos en aplicaciones para la web – un estudio comparativo.* 2002 (Disponible en http://www.lsi.us.es/docs/informes/lsi-2002-4.pdf)
13. **FERNANDEZ DE CASTRO FABRÉ, ASTRID; LOPEZ PADRÓN, ALEXANDER.** *Validación mediante criterio de usuarios del sistema de indicadores para prever, diseñar y medir el impacto en los proyectos de investigación.* 2014 (Disponible en https://scielo.sld.cu)

14. **FLANAGAN, DAVID.** *JavaScript - The definitive guide (7th Ed.).* 2020 (Disponible en https://books.google.com.cu/)
15. **GALICIA**, X. D. *Ossec: sistema de detección de intrusos.* 2017
16. **GÓMEZ BLANCO, ANA.** Artículo: *¿Qué es un ataque DDoS y cómo evitarlo?* 2019 (Disponible en https://www.bbva.com/es/que-es-un-ataque-ddos-y-como-evitarlo/)
17. **HAMILTON, NAOMI** *The A-Z of Programming Languages: BASH/Bourne-Again Shell.* 2011 (Disponible en https://www.computerworld.com/)
18. **HERNÁNDEZ, E. O.** *El Lenguaje Unificado de Modelado (UML)*. 2015 (Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF)
19. **INGHAM, KENNETH; FORREST, STEPHANIE.** *A History and Survey of Network Firewalls.* 2002
20. **ISLAS HERNANDÉZ, LUIS.** *Análisis y diseño orientado a objetos. Apuntes digitales.* 2014 (Disponible en http://cidecame.uaeh.edu.mx/lcc/mapa/proyecto/libro10/index.html)
21. **ITURRALDE, OSCAR J.** *Introducción a Los Patrones de Diseño: Un Enfoque Práctico.* 2016
22. **KANER, CEM.** *Artículo: What is a good test case?* 2013 (Disponible en http://www.kaner.com/pdfs/goodtest.pdf)
23. **LAPLANTE, PHILLIP A.** *Requirements engineering for software and systems.* 2009 (Disponible en https://content.taylorfrancis.com/books/download?dac=c2012-0-02503-5&isbn=9781466560826&format=googlepreviewpdf)
24. **LÓPEZ MARTÍNEZ, J. A.** *Sistema para la gestión de oportunidades de negocio para el centro de informática industrial.* 2015
25. **MACRAE, CALLUM.** *Vue.js: Up and Running: Building Accessible and Performant Web Apps.* 2018 (Disponible en https://books.google.com.cu)
26. **MARLETTE, TRAVIS.** *Splunk Best Practices.* 2016 (Disponible en https://books.google.com.cu/)
27. **MARTELLINI, MAURIZIO; MALIZIA, ANDREA.** *Cyber and Chemical, Biological, Radiological, Nuclear, Explosives Challenges: Threats and Counter Efforts.* 2017 (Disponible en https://books.google.com.cu/)
28. **MCGRATH, MIKE.** *HTML, CSS & JavaScript in easy steps.* 2020. (Disponible en https://books.google.com.cu/)
29. **MENENDEZ ALONSO, EVELYN.** *Introducción a herramientas case.* 2011.
30. **NEWMAN, ROBERT C.** *Computer security: protecting digital resources.* 2010.

31. **PAEZ, J.** *Análisis del método para la calificación de software para la selección de software aplicable a procesos educativos.* 2011, (Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=s2227-18992017000200007)
32. **PAGE SIGMAN, BETSY; DELGADO, ERICKSON; DIAKUN, JOSH.** *Splunk: Enterprise Operational Intelligence Delivered.* 2017
33. **PFLEEGER, CHARLES P.** *Security in computing.* 2015.
34. **POSTIGO PALACIOS, ANTONIO.** *Seguridad informática.* 2020 (Disponible en https://books.google.com.cu/)
35. **PRESSMAN, ROGER S.** *Software engineering: a practitioner's approach (7th ed.).* 2010.
36. **QUINTIERI FERNÁNDEZ, ALEX.** *Mecanismos de seguridad inteligente adaptativa.* 2006
37. **RODRÍGUEZ CANFRANC, PABLO.** *Ciberseguridad: Protegiendo la información vulnerable.* 2019
38. **RODRÍGUEZ S., TAMARA.** *Metodología de desarrollo para la actividad productiva de la uci.* 2015.
39. **RUMBAUCH JAMES, J. I. y GRADY, B.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2015
40. **SALCEDO, M. A.** *Sistema de prevención de intrusos basado en OpenBSD y Snort.* 2008.
41. **SANTOS LÓPEZ, FÉLIZ M.** *Implementation of non intrusive software component to monitor web applications in a public entity.* 2013. Disponible en (https://www.redalyc.org/articulo.oa?id=81632390016)
42. **SCARFONE, KAREN; MELL, PETER.** *Nist – Guide to intrusion detection and prevention systems (IDPS).* 2012.
43. **SEMETEYS, RAPHAËL.** *Method for qualification and selection of open source software.* 2008
44. **SOMMERVILLE, I.** *Ingenieria de software 9na s.l. Adison-Wesley*. 2011.
45. **TARAZONA T.** *Amenazas informáticas y seguridad de la información.* 2016
46. **TSANG, CURTIS H. K.** *Object-oriented Technology: From Diagram to Code with Visual Paradigm for UML.* 2005 (Disponible en https://books.google.com.cu/)
47. **VAN IMPE, KOEN.** *Defending against Apache web server DDoS attacks.* 2015.
48. **VAN ROSSUM, GUIDO; WARSAW, BARRY; COGHLAN, NICK.** *Style guide for python code.* 2013 (Disponible en https://www.python.org/dev/peps/pep-0008/)
49. **VUIKA, DENYS.** *Electron Projects: Build over 9 cross-platform desktop applications from scratch.* 2019 (Disponible en https://books.google.com.cu)

50. **WALLEN, JACK.** *How to protect secure shell on Centos 7 with Fail2Ban.* 2016.
51. **ZEIL, STEVEN J.** *Integrated Development Environments.* 2017 (Disponible en https://www.cs.odu.edu)

ANEXOS

Anexo 1: Entrevista realizada al especialistas de CESOL acerca de las vulnerabilidades de Nova Servidores.

Objetivo: Conocer, analizar y determinar las vulnerabilidades que presenta Nova Servidores 7 respecto a la seguridad de los datos que almacena.

1. ¿Qué servicios posee Nova Servidores 7?
2. ¿Considera que es propenso a los ataques e intentos de intrusión?
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 Usuarios

Tabla 9 Historia de usuario: Desinstalar Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-2	**Requisito: Desinstalar Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: media	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite la desinstalación de la herramienta Fail2Ban. Debe poseer una opción "instalar". Al finalizar la instalación deberá mostrar un mensaje de confirmación. De lo contrario lanza una notificación de error.	
Observaciones: para desinstalar la herramienta debe estar previamente instalada	

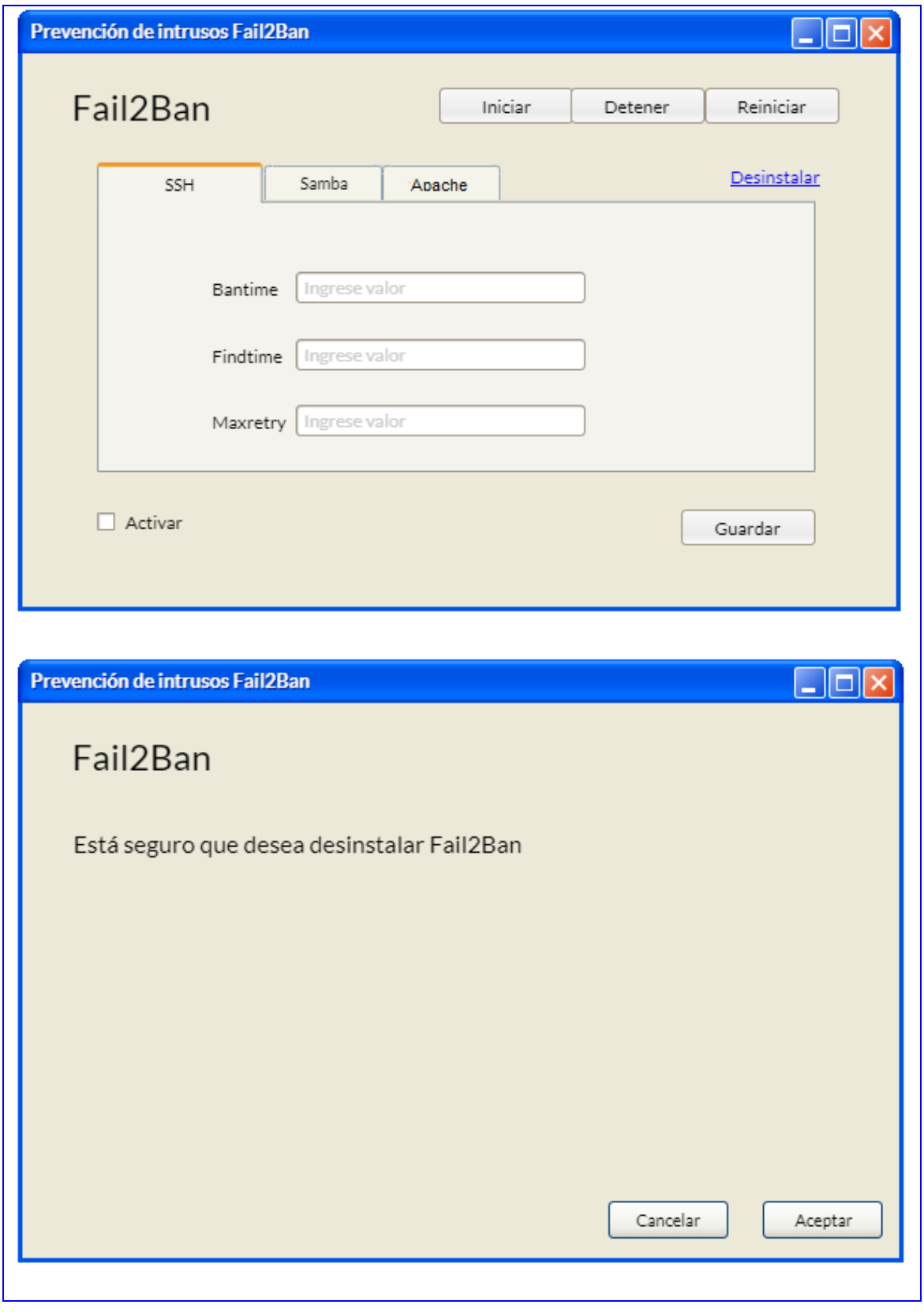

Prevención de intrusos Fail2Ban
Fail2Ban
Iniciar
Detener
Reiniciar
SSH
Samba
Apache
Desinstalar
Bantime
Ingrese valor
Findtime
Ingrese valor
Maxretry
Ingrese valor
Activar
Guardar
Prevención de intrusos Fail2Ban
Fail2Ban
Está seguro que desea desinstalar Fail2Ban
Cancelar
Aceptar

Tabla 10 Historia de usuario: Iniciar Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-3		
Requisito: Iniciar Fail2Ban**Programador: Michel Pedrera Suen**	**Iteración asignada: 1**	
Prioridad: media		**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica		**Tiempo real: -**
Descripción: permite al usuario iniciar la herramienta. El sistema debe tener la opción "iniciar"		
Observaciones: la herramienta debe estar previamente instalada. Si el servicio es iniciado satisfactoriamente, se muestra el siguiente mensaje de información: "el servicio ha sido iniciado". Si al momento de iniciar el servicio ocurre un error, se muestra un mensaje indicando los detalles de los errores.		

Tabla 11 Historia de usuario: Detener Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-4	**Requisito: Detener Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: media	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite al usuario apagar la herramienta. El sistema debe tener la opción “apagar”	
Observaciones: la herramienta debe estar previamente instalada. Si el servicio es apagado satisfactoriamente, se muestra el siguiente mensaje de información: “el servicio ha sido apagado”. Si al momento de iniciar el servicio ocurre un error, se muestra un mensaje indicando	

los detalles de los errores.

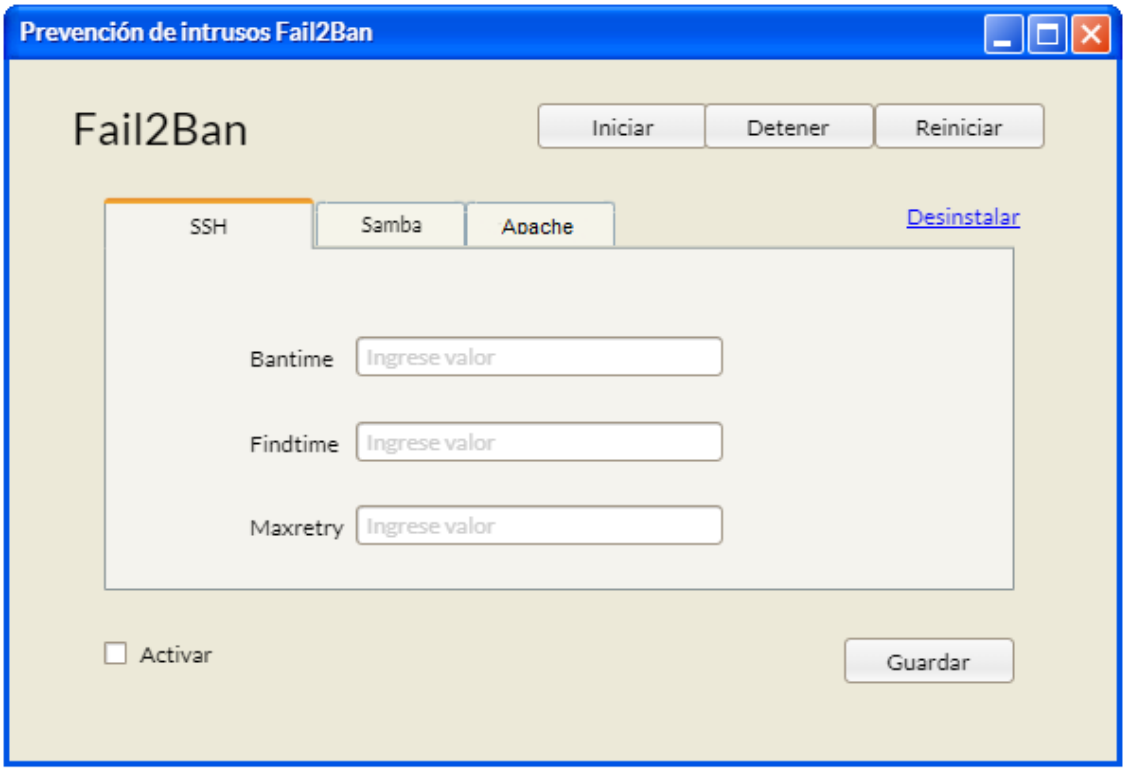


Tabla 12 Historia de usuario: Reiniciar Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-5	**Requisito: Reiniciar Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite al usuario reiniciar la herramienta. El sistema debe tener la opción "reiniciar"	
Observaciones: la herramienta debe estar previamente instalada. Si el servicio es reiniciado	

satisfactoriamente, se muestra el siguiente mensaje de información: “el servicio ha sido reiniciado”. Si al momento de iniciar el servicio ocurre un error, se muestra un mensaje indicando los detalles de los errores.

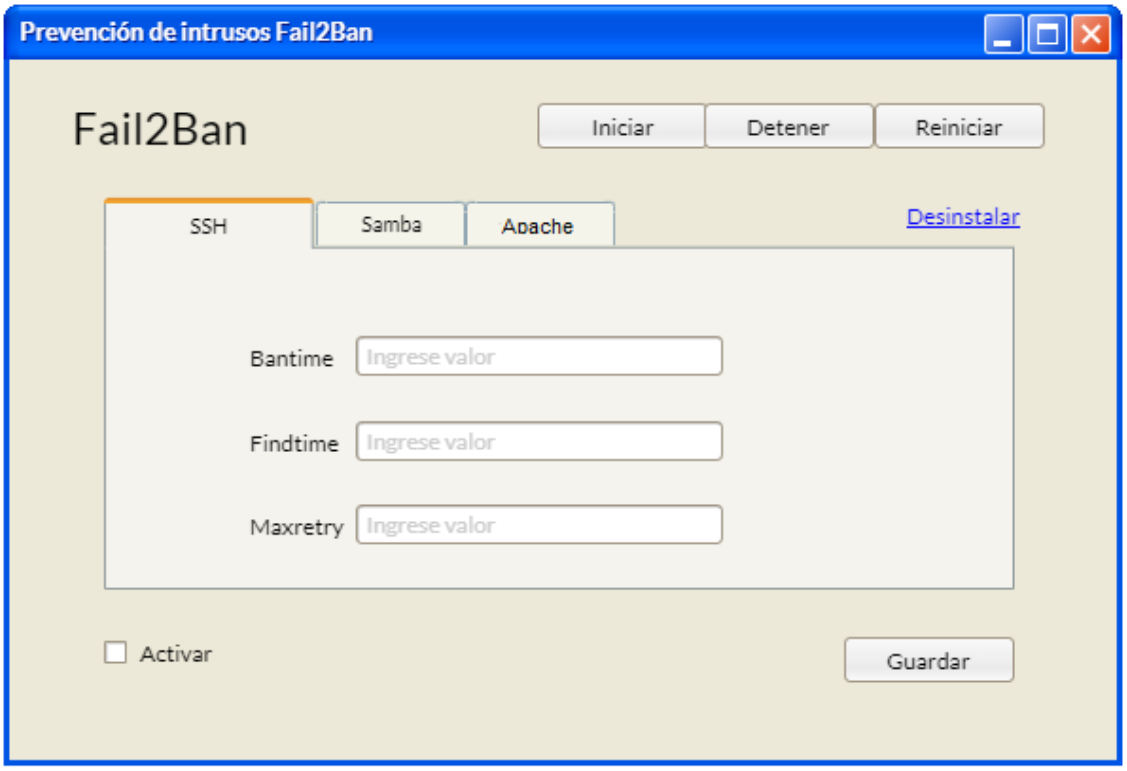


Tabla 13 Historia de usuario: Configurar protección para Apache.

Fuente: Elaboración Propia.

Número: HU-6	**Requisito: Configurar protección para Apache**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite configurar la protección para Apache, modificando las reglas de	

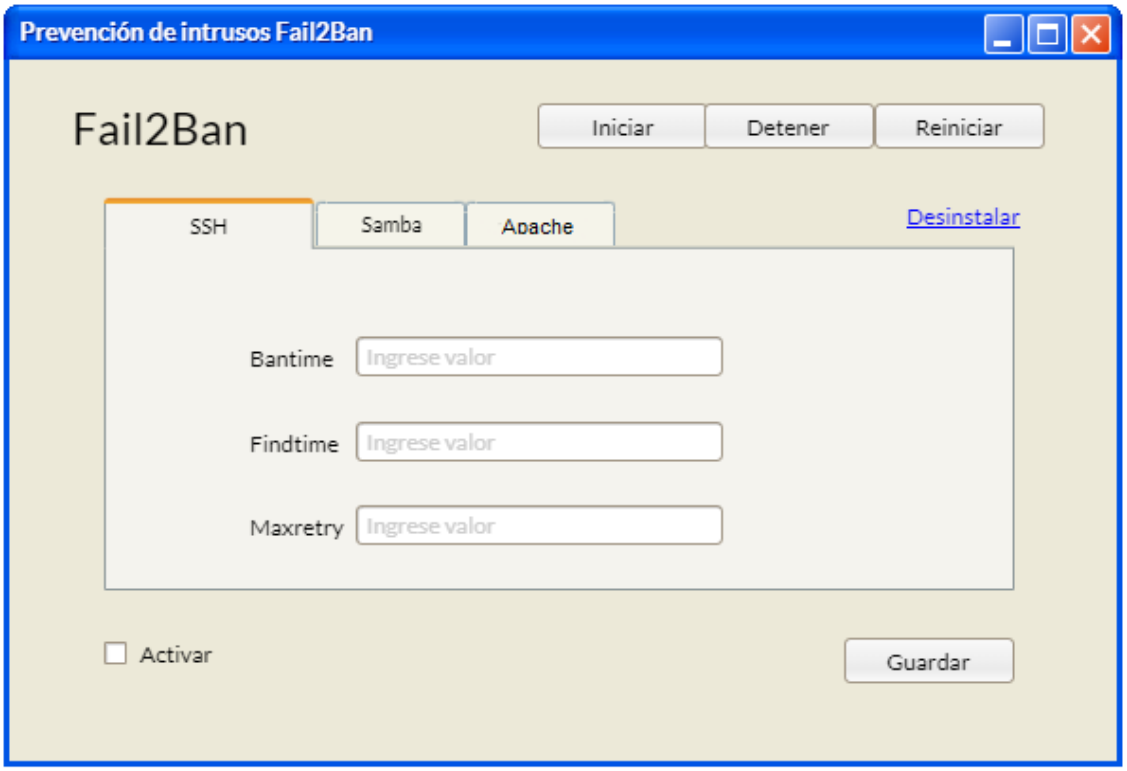
Fail2Ban.
Observaciones: la herramienta debe estar previamente instalada 

Tabla 14 Historia de usuario: Configurar protección para Samba.

Fuente: Elaboración Propia.

Número: HU-7	**Requisito: Configurar protección para Samba**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite configurar la protección para Samba, modificando las reglas de Fail2Ban.	

Observaciones: la herramienta debe estar previamente instalada

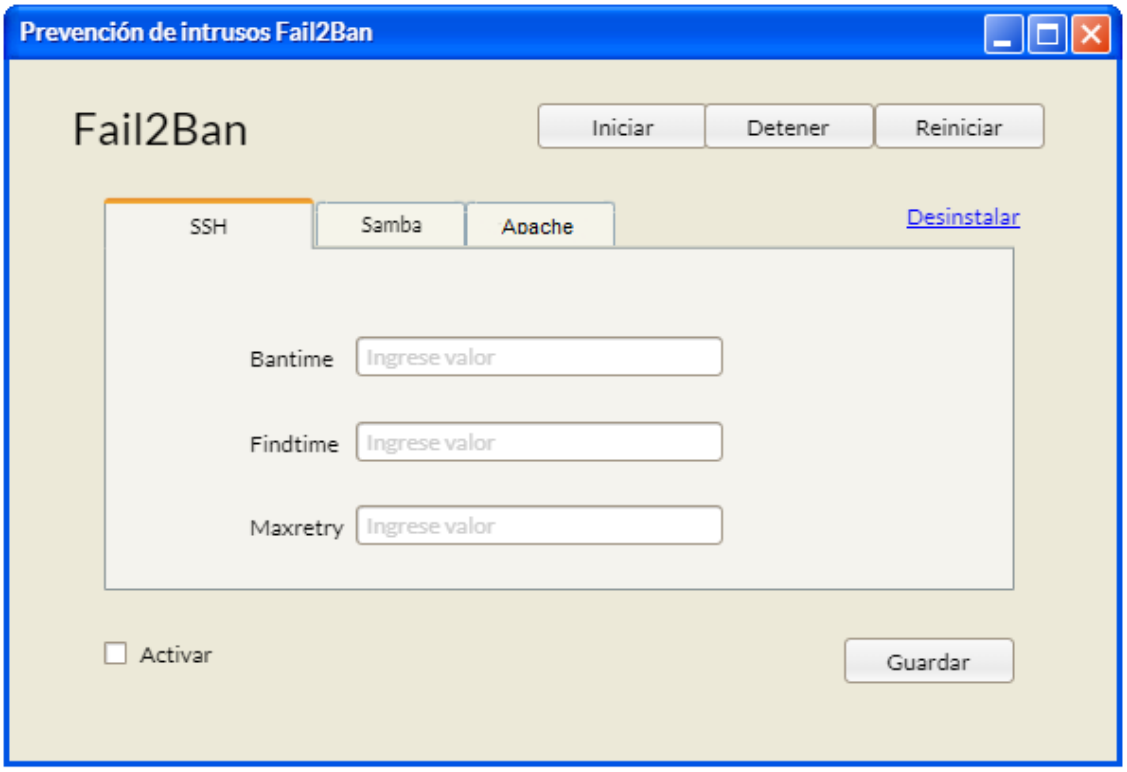


Tabla 15 Historia de usuario: Configurar protección para SSH.

Fuente: Elaboración Propia.

Número: HU-8	**Requisito: Configurar protección para SSH**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite configurar la protección para SSH, modificando las reglas de Fail2Ban.	
Observaciones: la herramienta debe estar previamente instalada	

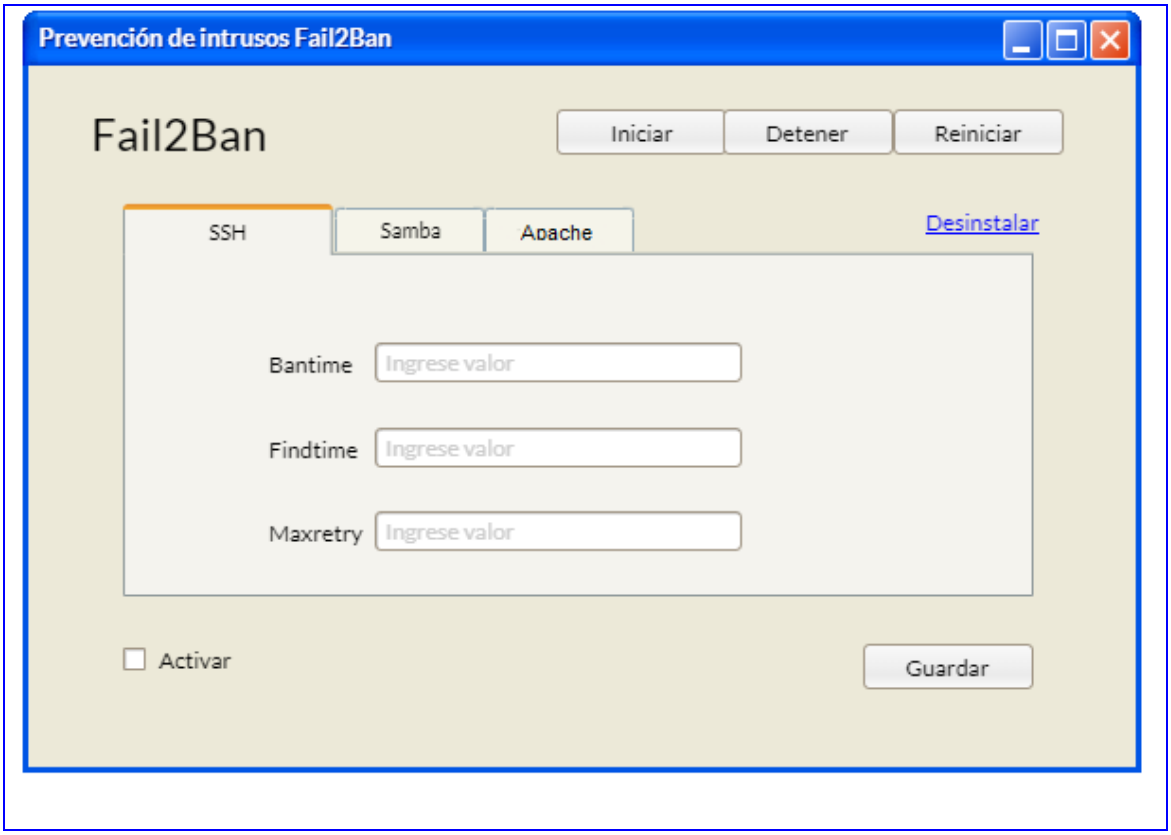


Tabla 16 Historia de usuario: Habilitar servicios de Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-9	**Requisito: Habilitar servicios de Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: el sistema debe permitir habilitar los servicios de Fail2Ban.	
Observaciones: la herramienta debe estar previamente instalada	

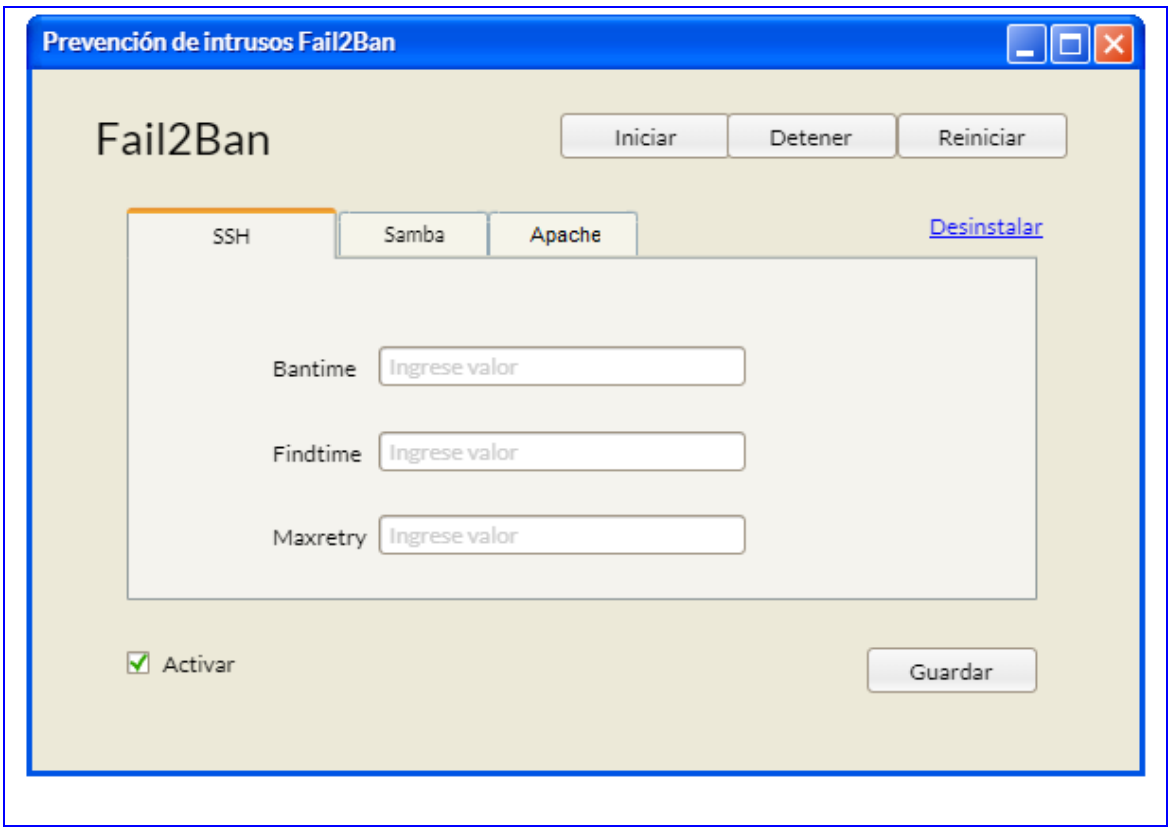


Tabla 17 Historia de usuario: Deshabilitar servicios de Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-10	**Requisito: Deshabilitar servicios de Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: el sistema debe permitir deshabilitar los servicios de Fail2Ban.	
Observaciones: la herramienta debe estar previamente instalada	

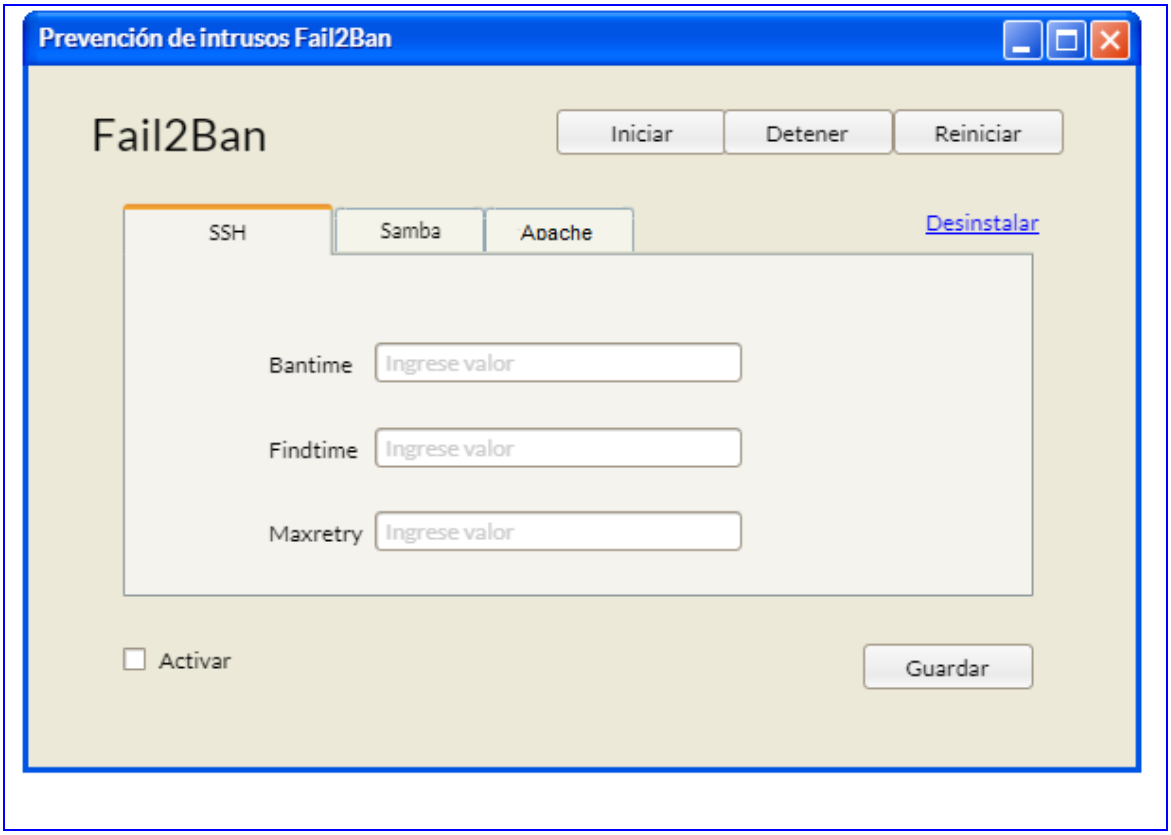


Tabla 18 Historia de usuario: Mostrar configuración de Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-11	**Requisito: Mostrar configuración de Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: media	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite mostrar la configuración de Fail2Ban.	
Observaciones: el servicio debe estar previamente instalado. De no existir información	

mostrará la notificación “vacío”

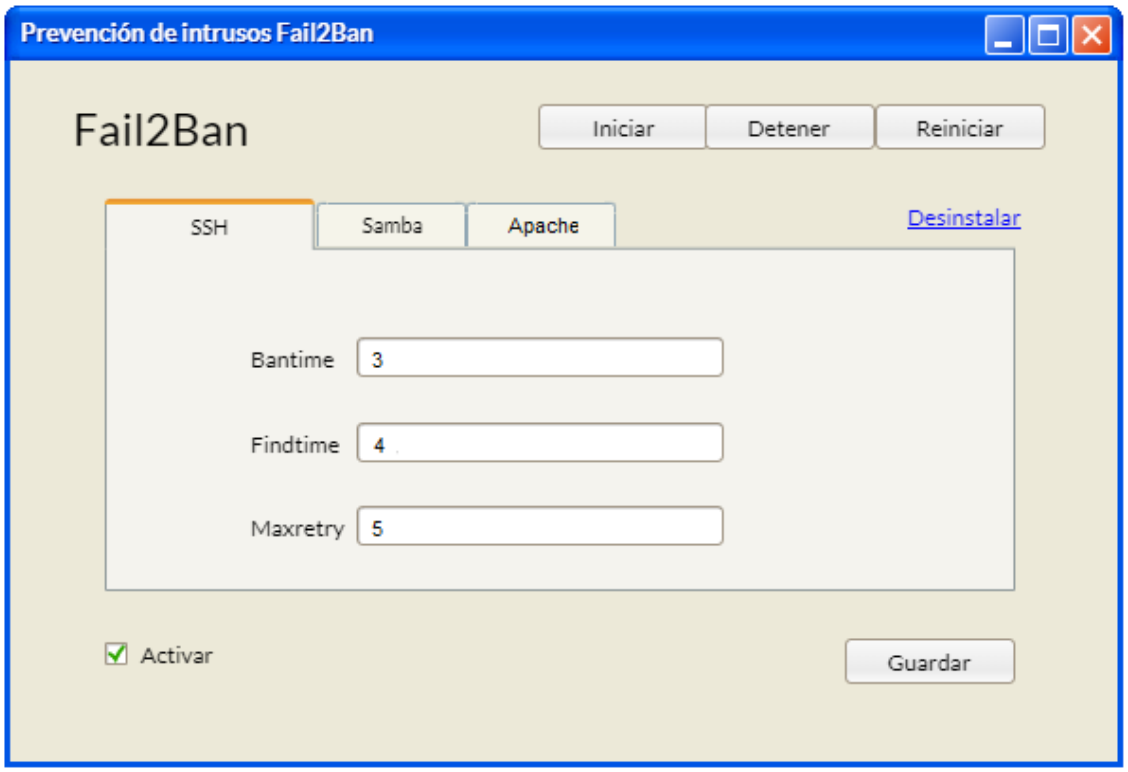


Tabla 19 Historia de usuario: Modificar configuración de Fail2Ban.

Fuente: Elaboración Propia.

Número: HU-12	**Requisito: Modificar configuración de Fail2Ban**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite al usuario modificar la configuración de Fail2Ban	
Observaciones: el servicio debe estar previamente instalado.	

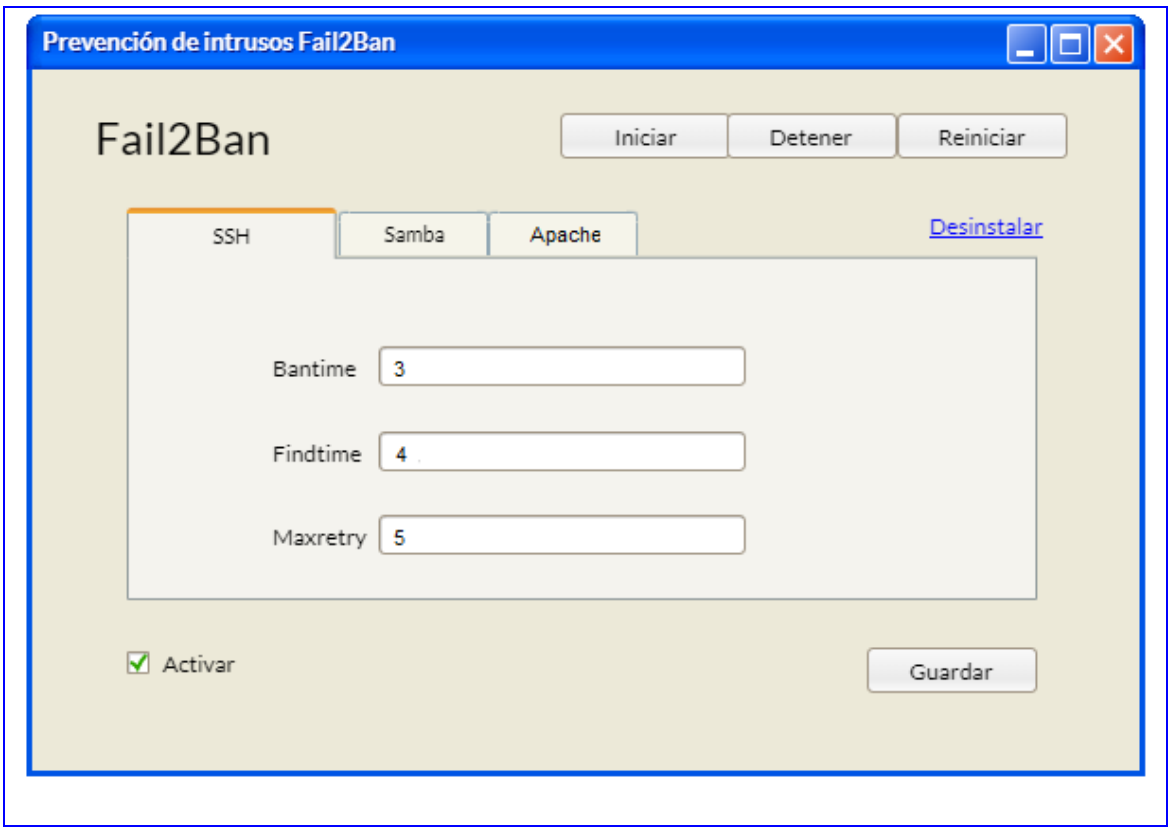
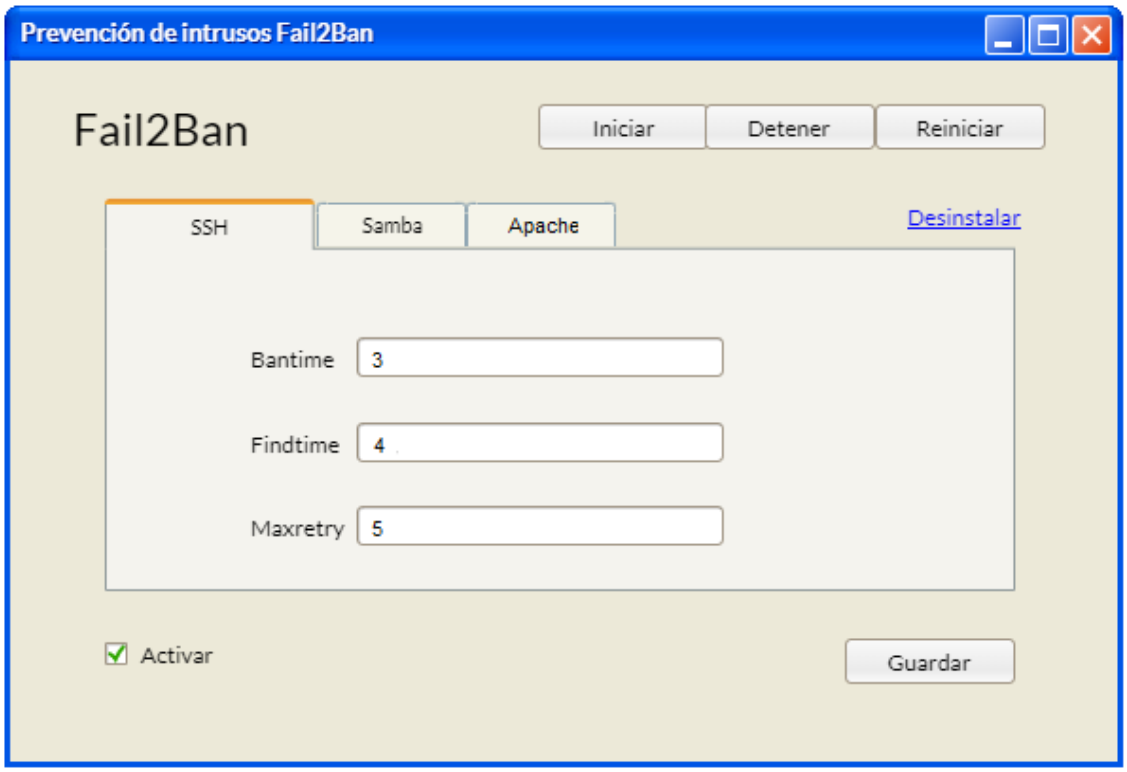


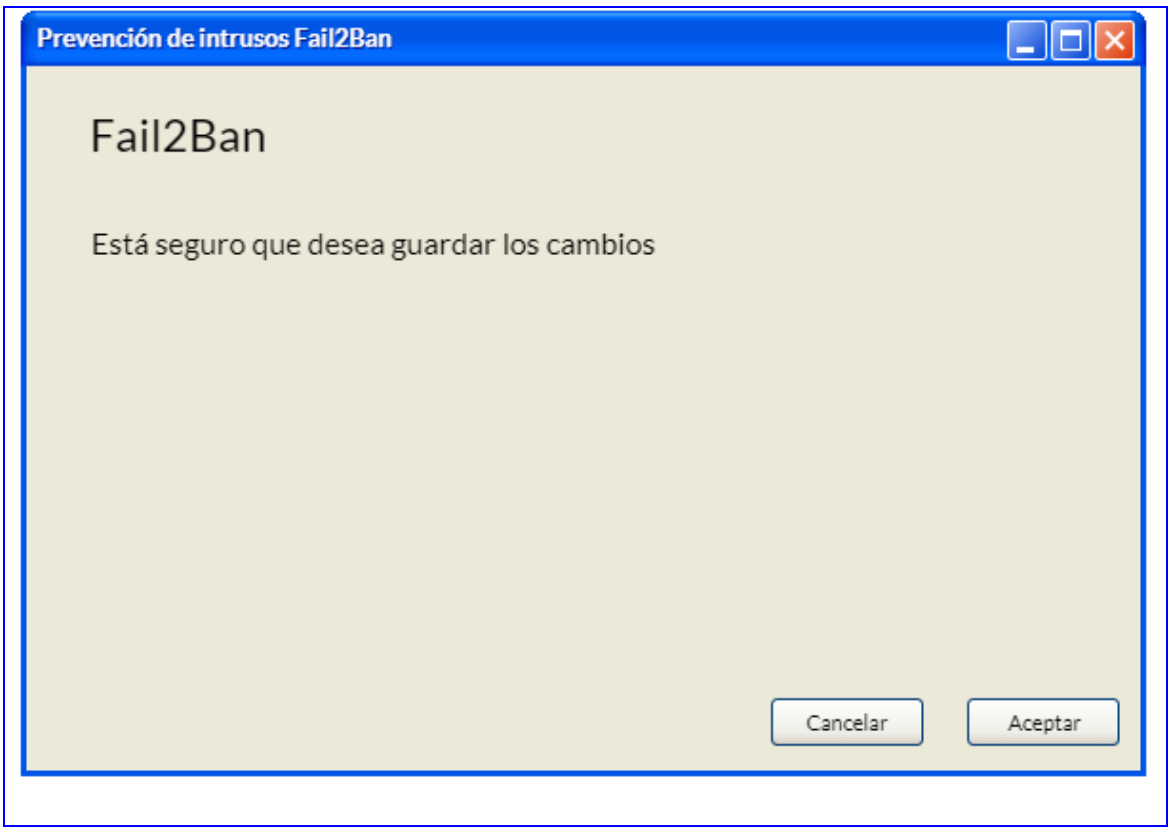
Tabla 20 Historia de usuario: Guardar cambios.

Fuente: Elaboración Propia.

Número: HU-13	**Requisito: Guardar cambios**
Programador: Michel Pedrera Suen	**Iteración asignada: 1**
Prioridad: alta	**Tiempo estimado: 72 horas**
Riesgo en desarrollo: no aplica	**Tiempo real: -**
Descripción: permite guardar los cambios realizados con la herramienta. Al finalizar la deberá mostrar un mensaje de confirmación. De lo contrario lanza una notificación de error.	

Observaciones: el servicio debe estar previamente instalado.





Anexo 3 Encuesta realizada a especialistas de CESOL

Objetivo: Evaluar la propuesta de solución desarrollada

Con el fin de conseguir su colaboración en la presente investigación, se le invita a responder el siguiente cuestionario, solicitando que exprese en sus respuestas criterios verídicos que guíen al autor de la investigación:

1. ¿Considera importante la implementación de un sistema que prevenga de los intrusos en Nova Servidores 7?

___Sí ___No ___ No sé

2. ¿Considera usted correcta la forma en que se realiza la prevención de intrusos en Nova Servidores 7?

___Sí ___No ___ No sé

3. ¿Considera usted factible la implementación de un sistema que prevenga de los intrusos en Nova Servidores 7?

___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 prevenir de los posibles intrusos en Nova Servidores 7?

Anexo 4 Encuesta realizada al cliente.

Objetivo: Identificar requisitos.

Con el fin de conseguir su colaboración en la presente investigación, se le invita a responder el siguiente cuestionario, solicitando que exprese en sus respuestas criterios verídicos que guíen al autor de la investigación:

1. ¿Cuáles son los protocolos y/o servicios más usados en Nova Servidores?

_____

2. ¿Cuáles son los protocolos y/o servicios que más vulnerabilidades han presentado en Nova Servidores?

_____

3. ¿Todos los protocolos y/o servicios necesitan ser protegidos a la vez?

_____

4. ¿Conoce la herramienta Fail2Ban y sus funcionalidades? ___Si ___No

5. ¿Ha trabajado antes con algún otro sistema de prevención de intrusos en Nova Servidores? ¿Cuál?

_____