Universidad de las Ciencias Informáticas

Facultad 3

"Sistema para viabilizar procesos de detección y bloqueo de ataques sobre aplicaciones web en la Universidad de las Ciencias Informáticas"

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

Autores: Sael José Alvarez Serrano y Raimer Salas González.

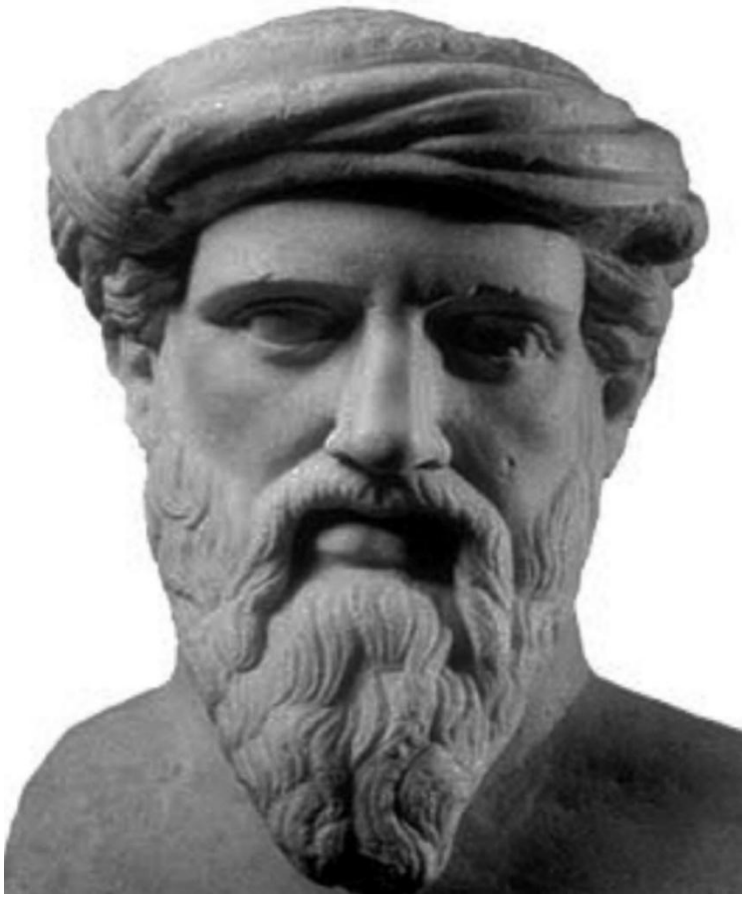
Tutores: Ing. Zénel Reyes Pérez**.**

Ing. Dennis Barreras Pérez.

La Habana, septiembre del 2020
"Año 62 de la Revolución"

"...Con orden y tiempo se encuentra el secreto de hacerlo todo, y de hacerlo bien..."

Pitágoras



DECLARACIÓN DE AUTORÍA

Declaro ser los autores del presente trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo. Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____

Sael José Alvarez Serrano

Firma del autor

_____

Raimer Salas González

Firma del autor

_____

Ing. Ing. Zenel Reyes Pérez

Firma del tutor

_____

Ing.Dennis Barreras Pérez

Firma del tutor

DATOS DE CONTACTO

Autor: Sael José Alvarez Serrano

Correo electrónico:sjalvarez@estudiantes.uci.cu

Autor: Raimer Salas González

Correo electrónico:rgsalas@estudiantes.uci.cu

Tutores: Ing. Zenel Reyes Pérez

Correo electrónico:zenel@uci.cu

Tutor: Ing. Dennis Barreras Pérez

Correo electrónico:dbperez@uci.cu

DEDICATORIA

Quisiera dedicar este trabajo de diploma, primeramente, a mis padres por ser las personas más importantes de mi vida, después se lo dedicaría

a mis abuelas una que me crio y me hizo ser el ser humano que soy y la otra que lo poco que la conocí se desvivió por mí,

por lo que hoy soy una persona privilegiada tengo una abuela en la tierra que sigue educándome y otra en el cielo que me cuida y

me indica el camino a seguir, por ultimo quisiera dedicárselo a esas personas de mi familia que saben que están en mi corazón y

a esa familia que la vida me ha dado la oportunidad de crear y que estoy orgulloso de que estén al lado mío. Para ustedes y por

ustedes todo mi esfuerzo, sacrificio y perseverancia.

Sael José Alvarez Serrano

Me gustaría dedicar este trabajo a mis padres que representan dentro de la arquitectura de mi vida, el artefacto más importante, a los familiares de

sangre por su constante preocupación e incondicional apoyo, y a los que no son de sangre, a esa familia que la vida te entrega por lo que vas sembrando

también va dedicado este trabajo, gracias por brindar tanto amor y saber estar ahí en los momentos más oscuros," siempre".

Raimer Salas González

RESUMEN

El ciberespacio se ha convertido en el nuevo escenario de operaciones para todos los países del mundo y, es por ello, que proteger los activos que operan en este, es fundamental para concretar su desarrollo. Cuba no está exenta de esta realidad y en su proceso de informatización ha orientado que se realicen acciones en torno a la protección de su espacio digital, principalmente, de las aplicaciones web que están públicas en Internet, las cuáles reciben distintos tipos ataques y que son difíciles de bloquear o detectar en el menor tiempo debido a la cantidad de fuentes que se deben revisar y los diversos patrones de ataques existentes.

En el presente trabajo, los autores, realizaron una revisión exhaustiva de varias fuentes para determinar los principales ataques a los que se pueden ver sometidos las aplicaciones del ciberespacio cubano, principalmente las de la universidad, pues estas constituyen el núcleo base para las pruebas de la aplicación, además, se muestran los artefactos generados para la aplicación por cada una de las fases de la metodología escogida, así como las herramientas y tecnologías a usar para la obtención de la solución.

El resultado final es un sistema informático que permite integrar a la herramienta OSSIM una de serie de reglas, a partir de la extracción de información de diversas fuentes de internet y del procesamiento, de estas, por los administradores permitiendo, con esta integración, una actualización del fichero viabilizando el proceso de detección o bloqueo de nuevos ataques.

Palabras Clave: *Bloqueo, ciberespacio, detección, patrones de ataque.*

ABSTRACT

Cyberspace has become the new stage of operations for all the countries of the world and, for this reason, protecting the assets that operate in it is essential to achieve its development. Cuba is not exempt from this reality and in its computerization process it has directed actions to be carried out around the protection of its digital space, mainly, of the web applications that are public on the Internet, which receive different types of attacks and that are difficult to block or detect in the shortest time due to the number of sources that must be reviewed and the various existing attack patterns.

In the present work, the authors carried out an exhaustive review of various sources to determine the main attacks to which Cuban cyberspace applications can be subjected, mainly those of the university, since these constitute the base core for the tests of the application, in addition, the artifacts generated for the application by each of the phases of the chosen methodology are shown, as well as the tools and technologies to be used to obtain the solution.

The end result is a computer system that allows a series of rules to be integrated into the OSSIM tool, based on the extraction of information from various internet sources and the processing of these by administrators, allowing, with this integration, an update. of the file enabling the process of detection or blocking of new attacks.

Keywords: Blocking, cyberspace, detection, attack patterns.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURA

ÍNDICE DE TABLA

INTRODUCCIÓN

La introducción de las tecnologías de la información y la comunicación (TIC) en la sociedad, trajo consigo una transformación en todos los sectores empresariales y sociales e incluso en la vida personal de los individuos. Provocando que sea difícil encontrar un proceso, administrativo, financiero, industrial y comercial que se siga realizando sin la intervención necesaria de las TIC [1].

Según[1]en la actualidad se depende críticamente de los sistemas construidos sobre la base de las TIC, de modo que su inoperatividad o malfuncionamiento, sea accidental o intencionado reporta graves consecuencias para los actores que los sufren. Al analizar estos incidentes se concluye que si se producen de manera accidental es muy poco probable prevenirlos, sin embargo, intencionalmente sí se pueden prevenir, por ejemplo, una persona mal intencionada que logre acceder a un sistema en donde se guardan registros de cuentas bancarias robándolas aplicando algún método determinado resulta en un incidente evitable y por tanto no accidental. Por lo que para su prevención aparece lo que se conoce como Ciberseguridad.

La Ciberseguridad es la protección de activos de información, mediante el tratamiento de las vulnerabilidades y amenazas. Entendiéndose como vulnerabilidad cibernética a la debilidad o fallo de un sistema de información que puede ser capaz de poner en riesgo la seguridad de toda o parte de la información, a su vez amenaza cibernética es la acción que se vale de esa vulnerabilidad para actuar contra el sistema de información. Con el uso de las Tecnologías de la Información y la Comunicación (TIC), se facilita un desarrollo sin precedentes en el intercambio de información y comunicaciones, que conlleva serios riesgos y amenazas en un mundo globalizado; y las amenazas en el espacio digital adquieren una dimensión global que va más allá de la tecnología. Por tanto, la ciberseguridad constituye una condición para permitir que los ciudadanos, puedan beneficiarse del uso de las ventajas de las tecnologías de la información [2].

Con la evolución de las TIC y el uso acentuado del internet, las aplicaciones web han tomado protagonismo ya que son utilizadas en todos los estratos de la sociedad. Estas poseen una arquitectura cliente-servidor en la cual el navegador que actuaría como cliente envía una solicitud por el protocolo http al servidor que a su vez envía una respuesta siendo procesada por el navegador.

Esta arquitectura posee vulnerabilidades que pueden ser aprovechadas para ser explotadas por algún atacante con el fin de obtener alguna ganancia, por ejemplo, si el envío de los datos de algún formulario se realiza utilizando el método GET no existiría ningún encapsulamiento de los datos mencionados provocando que fueran legibles para una persona que estuviera interceptando de alguna forma las solicitudes enviadas a través del protocolo http. Por tanto, se puede entender entonces que las aplicaciones Web brindan servicios y guardan datos sensibles de sus usuarios, ventajas que pueden quedar expuestas a las malas intenciones de alguien.

En concreto, las aplicaciones web a partir de lo anteriormente mencionado constituyen un punto de inflexión en lo que a la ciberseguridad se refiere. Y para constatar esta información se expondrán datos acerca del estado actual de la ciberseguridad en el mundo:

- Según un reciente informe de Google, en 2016 la compañía detectó un aumento de un 32 % respecto al año anterior en lo que al número de páginas atacadas se refiere [3].
- Se reportaron 12,6 millones de víctimas de usurpación de identidad identificadas en los EE. UU [4].
- La compañía VeriSign afirma que los resultados del año pasado han puesto de manifiesto que los ataques de tipo Denegación de servicio Distribuida (DDoS por sus siglas en inglés), no sólo están creciendo, sino que se están haciendo más sofisticadas y que cada vez es más complicado defenderse de ellas [5].

Cuba no está exenta de estos ataques mostrándose en los siguientes datos:

- El 18 de abril del 2013, a solo cuatro días de las elecciones en Venezuela, varios sitios cubanos en Internet, entre ellos, el portal Cubasí, recibieron un ataque de denegación de servicio que los afectó por varias horas [6].
- El Instituto Cubano de Amistad con los Pueblos (ICAP) denunció el 21 de agosto de 2019 un ataque cibernético contra Cubainformación.tv, medio alternativo del País Vasco, los que intentaban acceder a Cubainformacion.tv encontraban un mensaje: la página web “ha sufrido un fuerte ataque que ha eliminado parte de su

contenido. Se mantendrá cerrada hasta nuevo aviso. Seguimos en comunicación con usuarias y usuarios a través de cubainformacion@cubainformacion.tv" [7].

- La Oficina de Seguridad para las Redes Informáticas (OSRI) registró en octubre de 2017 más de 600 incidentes de ciberseguridad a páginas web, de mayor o menor gravedad [8].

La Universidad de las Ciencias Informáticas(UCI) no está exenta a estos ataques ya que cuenta con un portafolios de aplicaciones web que se encuentran públicas en internet.

La Dirección de Seguridad Informática de la UCI que vela y controla todo lo referente a ataques tanto externos como internos hacia plataformas establecidas que funcionan de manera interna y de forma pública a internet. Esta área cuenta con varios procesos, los tres fundamentales son: la gestión de incidentes, el monitoreo de la red y la protección de sitios web mediante una herramienta llamada Open Source Information Security (OSSIM) siendo esta una fuente abierta de información de seguridad y gestión de eventos de sistema además se integra dentro una selección de herramientas diseñadas para ayudar a los especialistas en seguridad en la detección y prevención de ataques. Durante los intercambios con los especialistas de esta área, estos hicieron alusión que para el proceso de protección de sitios web el departamento lo realiza en gran parte de forma manual provocando que los tiempos necesarios para poder proteger algún sitio web de forma efectiva se vean afectados, trayendo consigo que la detección de patrones de ataques novedosos en ocasiones se vea afectada. Dígase un patrón de ataque a todo aquel registro de tráfico en la red que exponga una amenaza.

Una vez encontrados los patrones de ataques por el especialista de seguridad en fuentes de información como **Open Source Intelligent (OSINT)**, **Malware Information Sharing Platform (MISP)** y el **blog Segu. Info**, el proceso continuaría con el especialista realizando una actualización manual en el archivo de configuración (**archivo.cfg**) de la herramienta OSSIM para ello el mismo tendría que crear una regla utilizando una expresión regular.

Todos los patrones añadidos al archivo.cfg tienen prioridad alta, sin embargo, quizás no todos los patrones que se inserten tengan este tipo de prioridad, refiriéndose prioridad alta a la probabilidad alta de que sea un ataque el patrón detectado(pues puede existir casos

en los que se evidencie un falso positivo) por tanto su procesamiento que incluye el bloqueo inmediato del atacante pasaría dentro de la 'cola de procesamiento de patrones' al primer lugar, luego por consiguiente hay que analizar su comportamiento en profundidad, o sea detallar aspectos como la hora en la que fue detectado el IP[1] que generó el registro evitando así la menor cantidad de falsos positivos.

Para satisfacer esta cuestión el especialista escribe en otro archivo.cfg lo que se conoce como directiva que no es más que un conjunto de reglas que asocian a un evento generado por la detección de un patrón de ataque novedoso. Una directiva se inicia una etiqueta que genera una directiva llamada 'directiva' y luego con tres etiquetas indispensables [9]:

- Id. Identificador único de la directiva, comprendido dentro del rango de su categoría.
- Name. Nombre descriptivo de la directiva.
- Priority. Nivel de prioridad que es global a la directiva.

Cada directiva empieza con una regla que debe de coincidir con un evento detectado. Algunas de las etiquetas de las reglas que puede contener una directiva [9]:

- Reliability. Cuando se habla del riesgo de un ataque se podría definir como la "probabilidad" de que esto ocurra. Si una máquina se conecta a 5 máquinas distintas de su propia subred al puerto 445, puede presentar un comportamiento normal, pero si este número va aumentando y ya son 15 las máquinas, empieza a ser un tanto sospechoso, y si se comunica con más de 30 máquinas en menos de una hora, se puede decir que el comportamiento extraño y el ataque comienza a ser evidente. Cada regla tiene su propia fiabilidad, determinando la fiabilidad de esa regla en particular dentro de la cadena de reglas completas de la directiva. El valor de la fiabilidad puede ser entre 0 y 10, se puede especificar un valor absoluto (ej. 6) o relativo (ej. +3 significa tres más que en nivel anterior).
- Ocurrence. Indica cuántas ocurrencias debe poseer.
- Time_out. Indica el tiempo de vida de una regla, esperando a que el evento ocurra, hasta que la regla expira.

[1] Protocolo de Internet, sistema estándar mediante el cual funciona la internet, por medio de un proceso de envío y recepción de información.

Entre otras, que determinan características que debe poseer un evento para que la herramienta OSSIM internamente pueda determinar qué acción realizar para contrarrestar un posible ataque.

Todo este proceso de creación de las directivas e inserción de patrones de ataque, a partir de la información recopilada por los autores en los intercambios con los especialistas, tiende a ser poco viable. Estos procesos se realizan de forma manual, quedando propensos al error humano ya que el especialista puede equivocarse en añadir el valor de alguna etiqueta, proporcionando que no se correlacionen de manera correcta los eventos trayendo consigo que la herramienta OSSIM no sea capaz de realizar los procesos de detección y bloqueo. Se han registrado a través de análisis internos del departamento que un 60% de los ataques fructíferos hechos en la universidad en el mes de octubre de 2017(manteniéndose en un rango de 55% y 60% en meses posteriores) se han debido a errores humanos provocados por la complejidad de escribir los patrones y directivas.

Por otra parte, este proceso también carece de la velocidad necesaria, pues el tiempo en que se demora el especialista en escribir la regla puede ser aprovechado por algún atacante para explotar alguna vulnerabilidad. Para constatar lo antes expuesto se realiza un estudio interno en el cual se afirma que el 76.3 % de los ataques fructíferos hechos en marzo del 2018(manteniéndose en un rango del 66.2% y 77% en los meses posteriores) se podían haber evitado si el tiempo de actualización de la regla y directivas hubiese sido menor valorando para esto el registro de la actualización de la regla y la directiva junto al momento en el que fue hecho el ataque. Esperando entonces que los porcentajes de ataques proporcionados por estas dos variables: error humano y tiempo pueden ser disminuidos a menos del 5 % en los meses posteriores, convirtiéndose esto en una forma medible para viabilizar los procesos de detección y bloqueo.

A partir de la situación descrita anteriormente se propone como **problema a resolver:** ¿Cómo viabilizar los procesos de detección y bloqueo de ataques sobre las aplicaciones web del ciberespacio de la Universidad de las Ciencias Informáticas?

Delimitando como **objeto de estudio:** La seguridad de la información en aplicaciones web.

Se define como **objetivo general:** Desarrollar un sistema informático que permita

viabilizar los procesos de detección y bloqueode ataques a través de patrones ya conocidos, que contribuya a la seguridad informática de la Universidad de las Ciencias Informáticas**,** enmarcado en el **campo de acción:** Informatización de los procesos de detección y bloqueo de ataques a aplicaciones web.

Se plantea la siguiente **idea a defender:** Si se desarrolla un sistema informático que sea permita viabilizar los procesos de detección y bloqueo de ataques a través de patrones de ya conocidos, entonces aumentará la seguridad de las aplicaciones web de la Universidad de las Ciencias Informáticas públicas en Internet.

Desglosándose en los siguientes **objetivos específicos:**

- Fundamentar la investigación mediante la elaboración del marco teórico para sustentar los conceptos y la propuesta solución.
- Implementar el sistema informático que sea capaz de viabilizar los procesos de detección y bloqueo de ataques a través de patrones de ataques ya conocidos.
- Validar la solución propuesta mediante pruebas unitarias, pruebas de aceptación, pruebas de valor límite, pruebas de comportamiento y pruebas a los requerimientos no funcionales además de pruebas para demostrar que el sistema en efecto es viable.

Para darle cumplimiento a los objetivos planteados se utilizaron los siguientes **métodos de investigación:**

Métodos teóricos:

- **Histórico-Lógico:** para llevar a cabo la formulación de conclusiones y el estudio de los principales cambios que ha sufrido la web desde sus inicios, lo cual ha traído consigo que sus características y las restricciones hayan venido en aumento en los últimos años; así como la evolución que han sufrido los procesos de desarrollo de software guiados por metodologías y a su vez contrastar la necesidad de adaptar las mismas para dar un mayor entendimiento y calidad.
- **Análisis Sintético:** para descomponer el problema de investigación en elementos por separado, de esta forma se puede profundizar en el estudio de cada elemento, para luego sintetizarlos en la solución de la propuesta.

- **Modelación:** para la generación de los artefactos ingenieriles necesarios en la concepción de la solución, así como para entender el contexto en el que se enmarca la investigación.

Métodos Empíricos:

- **Entrevista**: Permitió obtener la información necesaria para esbozar el proceso de gestión de la asignación.
- **Observación**: Permitió estudiar el comportamiento de las variables de la investigación.
- **Análisis documental:** proporciono la revisión de la literatura necesaria para obtener la información del estado actual del objeto de investigación considerándose diversos autores que han trabajado el tema y sus resultados.
- **Revisión Bibliográfica:** se empleó en las actividades de búsqueda, identificación y análisis de la información existente sobre la evolución de las metodologías de desarrollo de software, en particular para el desarrollo de aplicaciones web, para así poder comprender y poder tomar una decisión coherente con las particularidades del negocio.

Posibles resultados:

Se obtendrá un sistema informático que permita la viabilización de los procesos de detección, bloqueo de ataques a través de patrones de ataques ya conocidos.

El presente trabajo se encuentra estructurado como sigue:

- **Capítulo 1**. “Fundamentación teórica”: en este capítulo se abordan los conceptos asociados a la investigación para el desarrollo de la propuesta de solución, centrándose en el estudio del marco teórico asociado a la investigación y se construye el estado de arte asociado a la misma. Para el desarrollo del estado de arte se realizó una revisión bibliográfica de las metodologías de desarrollo de software, para posteriormente seleccionar la metodología base [2] y por consiguientes las adaptaciones que se le harían a la misma.
- **Capítulo 2**. “Propuesta de solución”: se detalla la propuesta solución mostrando los artefactos generados a partir de las fases de análisis y diseño según la metodología de desarrollo utilizada.

[2] Definimos metodología base, como la metodología principal que guía una investigación.

- **Capítulo 3.** “Implementación y validación de la solución”: se detalla la propuesta de solución mostrando los artefactos generados relacionados con las fases de implementación, prueba y despliegue. Se muestran los resultados de las pruebas realizadas para validar técnicamente la solución, así como los elementos que contrastan la incidencia de la solución en las variables definidas en el problema de la investigación.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se presentan elementos sobre los que se establece la investigación, mediante la revisión exhaustiva de la literatura, con el propósito de estudiar, identificar y adaptar las metodologías que más se adecúan a los propósitos expuestos con anterioridad siendo estas las que guían de manera consistente el proceso de desarrollo de software para poder obtener un producto final de calidad.

1.1 Principales Conceptos

A continuación, se conformará el sustento teórico de la presente investigación; se profundizará en los conceptos fundamentales asociados a las metodologías de desarrollo de software para una mayor comprensión de la investigación realizada.

Aplicaciones Web

Las aplicaciones web, llamadas "*webapps*", es un tipo de software que está centrado en redes agrupa una amplia gama de aplicaciones. En su forma más sencilla, las webapps son poco más que un conjunto de archivos de hipertexto vinculados que presentan información con uso de texto y gráficas limitadas. Sin embargo, desde que surgió Web 2.0, las webapps están evolucionando hacia ambientes de cómputo sofisticados que no sólo proveen características aisladas, funciones de cómputo y contenido para el usuario final, sino que también están integradas con bases de datos corporativas y aplicaciones de negocios[10].

Detección

Es el producto de la acción de detectar, o sea de localizar algo que es difícil observar a simple vista, o de advertir; es lo que no muestra evidencia [11].

Bloqueo

Es una restricción (total, parcial, permanente o temporal) que se impone a un usuario dentro de un sistema informático. Lo habitual es que el usuario sea bloqueado (o baneado de uso común no oficial) cuando viola las normas de uso de un servicio determinado [12].

Patrón de ataque

Los patrones de ataques (PA) constituyen una herramienta de conocimiento para el diseño, desarrollo y despliegue de software seguro, que permite capturar y comunicar la perspectiva del atacante [9].

Directiva

Una directiva es un conjunto de reglas que se deben de cumplir para que Ossim genere una alarma con un determinado nivel de riesgo [9].

1.2 Patrones de Ataques

Los patrones de ataque (PAs) conforman una descripción de los métodos utilizados para explotar software, orientados a un objetivo relativamente "destructivo" y que son un elemento clave para identificar y comprender las diferentes perspectivas en las que una amenaza se puede convertir en una vulnerabilidad [13].

Ofrecen información que permite determinar factores de riesgo, evaluaciones de impacto, detección proactiva de ataques y apoyo en las decisiones de seguridad [14]. De esta manera, brindan una forma coherente de enseñar a diseñadores y desarrolladores como sus sistemas pueden ser atacados y como pueden efectivamente defenderse. Los PAs constituyen también una herramienta de conocimiento para el diseño, desarrollo y despliegue de software seguro, que permite capturar y comunicar la perspectiva del atacante (completan la definición de una amenaza). Podemos resumir los beneficios que se obtienen de su uso de la siguiente manera:

- Ayudan a categorizar los ataques de una forma significativa, haciendo posible un análisis efectivo de los problemas y las soluciones.
- Los PAs permiten identificar los tipos de ataques conocidos a los que una aplicación puede estar expuesta y así construir en la propia aplicación las mitigaciones adecuadas.
- Contiene el nivel de detalle suficiente acerca de cómo tienen lugar los ataques de talforma que los desarrolladores pueden ayudar a prevenirlos. Sin embargo, por lo general, no contienen detalles específicos inapropiados acerca de los exploit[3] reales a fin de no favorecer el

[3] Exploit: es un programa informático, una parte de un software o una secuencia de comandos que se aprovechan de un error o vulnerabilidad para provocar un comportamiento no intencionado o imprevisto en un software, hardware o en cualquier dispositivo electrónico.

aprendizaje no deseado por parte de atacantes menos expertos. La idea de fondo es que no se los pueda utilizar directamente para crear exploit automatizados.

Para dar una descripción más detallada se utilizará un ejemplo de un patrón de ataque.
En diciembre del 2015 se publicó una vulnerabilidad de ejecución remota de comandos que afectaba a todas las versiones desde la 1.5 hasta la 3.3 del sistema de gestión de contenido **Joomla**. La forma de explotación está compuesta por la siguiente petición HTTP que representa un PA [15]:

```
X.X.X.X – – [12/Dec/2015:16:49:40 -0500] "GET /contact/
HTTP/1.1" 403 5322 "http://google.com/"
"}__test|O:21:\x22JDatabaseDriverMysqli\x22:3: ..
{s:2:\x22fc\x22;O:17:\x22JSimplepieFactory\x22:0:{}s:21:\x22\x
5C0\x5C0\x5C0disconnectHandlers\5:..{s:8:\x22sanitize\x22;O:
20:\x22JDatabaseDriverMysql\x22:0:{}s:8:\x22feed_url\x22…
```

Figura 1 Ejemplo de patrón de ataque

Fu*ente:*[15]

Esta forma de PAs representa una petición HTTP parseada y su uso se encuentra estandarizada en el formato .cv para que pueda ser procesable por el sistema. Para poder obtener un PAs es necesario extraerlo de fuentes de información que ofrezcan confiabilidad en los datos que brindan [16]. Dentro de este tipo de fuentes confiables se encuentra las de tipo libre, o como también se les conoce **Open Source Intelligence** (**OSINT** por sus siglas en inglés). Al momento de realizar un proceso de selección de estas fuentes dentro del ámbito de seguridad se tuvo en cuenta los siguientes factores:

- Medios y recursos que dispone.
- Aceptación que tiene en la comunidad de la seguridad.
- Credibilidad que se le da al contenido que aporta encontrada a través de la valoración de expertos y especialistas en seguridad además de la triangulación y contrastación del contenido mencionado a través de otras fuentes.
- Formato de la fuente estructurado de tipo .cv que sea capaz de ser procesado por el sistema.

Encontrándose en este proceso tres fuentes **blocklist.de**, **misp-project.org** y **cve.mitre.org** que servirían de base para la extracción de los PAs. En la primera mencionada se proporciona una **Application Programming Interface**[4] (**API** por su siglas en inglés) para la extracción de una lista de negra de IPs que ha resultado ser muy beneficiosa para miembros de la comunidad y en la segunda también se proporciona una API que cuenta con una lista estructurada de PAs detectados, la tercera es un portal en donde se describe PAs con un formato de tipo **Common Vulnerability and Exposure** (**CVE** por su siglas en inglés) distinto al requerido para el procesamiento del sistema, sin embargo puede resultar de gran utilidad al especialista ya que aunque no es procesable por el sistema a realizar, el especialista en seguridad si puede entender y procesar la información que se brinda ahí pudiendo hacer una actualización manual de los PAs.

1.3 Soluciones Afines

Un análisis del panorama actual del software, arroja como resultado que la implementación de medidas de seguridad mientras se desarrolla un software, no es económicamente factible, debido a los costos y los recursos necesarios, por lo cual un enfoque común es implementar una capa adicional encima de las aplicaciones web [17]. Uno de los mecanismos utilizados como parte de esta capa adicional son los **Detectores de Intrusos de Red (NIDS)** como es el caso de **Snort** y el **Suricata**.

Snort

Es un sniffer y registrador de paquetes basado en libpcap [PCAP94] que puede usarse como un sistema de detección de intrusiones de red (NIDS) liviano. Cuenta con un registro basado en reglas para realizar una coincidencia de patrones de contenido y detectar una variedad de ataques y sondas, como desbordamientos de búfer [ALE96], escaneos de puertos furtivos, ataques CGI, sondas SMB y mucho más. Tiene capacidad de alerta en tiempo real, con alertas que se envían a syslog, mensajes de `` WinPopup '' del Bloque de mensajes del servidor (SMB) o un archivo de “alerta” separado [18].

Esta herramienta se configura utilizando conmutadores de línea de comando y comandos opcionales Berkeley PacketFilter [BPF93]. El motor de detección se programa utilizando

[4] API: conjunto de reglas (código) y especificación que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

un lenguaje simple que describe las pruebas y acciones de paquetes. La facilidad de uso simplifica y acelera el desarrollo de nuevas reglas de detección de exploit [18].

Suricata

Es una herramienta gratuita y de código libre que se ha desarrollado para controlar el tráfico de red y su principal objetivo es que rastree o busque los eventos de seguridad que pueden indicar un ataque o posible intrusión en un servidor o en algún equipo de la red. Está basada en el sistema de Snort IDS, que también es un sistema de detección de intrusos. Es capaz de realizar análisis multihilo, de forma nativa decodificar flujos de red y ensamblar archivos de secuencias de red mientras realiza el análisis [19].

Los protocolos más comunes son reconocidos automáticamente por Suricata, tanto http, https, ftp, smtp, pop3 y otros, permitiendo así que podamos configurar reglas para los permisos y el filtrado del tráfico que entra y sale, además controlamos el puerto por el que se accede a cada protocolo [19].

Se concluye que un NIDS está basado en la inspección de paquetes a nivel de red, usualmente utiliza dos enfoques para la detección de ataques, uno basado en firmas y otro basado en anomalías. Sin embargo, cuando se trata de aplicaciones web, los mismos carecen de efectividad por las siguientes razones [15]:

- Si el tráfico HTTP está encriptado, los NIDS no pueden inspeccionarlo.
- Están diseñados para operar sobre las Capas 3 y 4 del Modelo OSI, mientras que las aplicaciones web se encuentran en la Capa 7 (Aplicación).
- Los atacantes pueden utilizar técnicas de evasión de IDS como codificación HTTP, desfragmentación, etc.

Por otra parte, se encuentran los denominados **Firewall de Aplicaciones Web (WAF)**. Los WAF actúan como intermediario entre la aplicación y el visitante que navega por un sitio web, interceptando y eliminando solicitudes maliciosas antes de que puedan causar daños, como es el caso de **AQTRONIX WebKnight** y **Citrix Web App Firewall**.

AQTRONIX WebKnight

Es un firewall de aplicaciones para Internet Information Server (IIS) y otros servidores web y se publica bajo la Licencia Pública General de GNU. Más particularmente, es un filtro Interfaz de Programación de Aplicaciones del Servidor Internet (ISAPI) que asegura su

servidor web al bloquear ciertas solicitudes. Si se activa una alerta, este se hará cargo y protegerá el servidor web [20].

Lo hace escaneando todas las solicitudes y procesándolas según las reglas de filtro, establecidas por el administrador. Estas reglas no se basan en una base de datos de firmas de ataque que requieren actualizaciones periódicas. En cambio, utiliza filtros de seguridad como desbordamiento de búfer, inyección SQL, recorrido de directorio, codificación de caracteres y otros ataques. De esta manera, puede proteger su servidor contra todos los ataques conocidos y desconocidos [20].

Debido a que es un filtro ISAPI, tiene la ventaja de trabajarestrechamente con el servidor web, de esta manera puede hacer más que otros firewalls y sistemas de detección de intrusos, como escanear el tráfico encriptado [20].

Citrix Web App Firewall

Es un firewall de aplicaciones web (WAF) que protege las aplicaciones y sitios web de ataques conocidos y desconocidos, incluidas las amenazas de capa de aplicación y de día cero. A pesar de un panorama de amenazas en constante evolución, ofrece protección integral sin degradar el rendimiento ni los tiempos de respuesta de las aplicaciones. Disponible como una solución en la nube o integrado en la plataforma. Citrix ADC, los controles de configuración simplificados mitigan aún más el riesgo. Nuestras opciones de licencias agrupadas le permiten crecer de forma incremental y escalar a demanda [21].

Se concluye que los WAF tienen como principal desventaja la alta tasa de falsos positivos[15].

1.4 Proceso de desarrollo de software

Los procesos de desarrollo representan una estructura aplicada a la confección de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso de desarrollo, cada uno de los cuales describe un enfoque diferente para las actividades que tienen lugar durante el proceso. Algunos autores como [10] y [22] consideran un modelo de ciclo de vida un término más general que un determinado proceso de desarrollo de software, por ejemplo, hay varios procesos de desarrollo de software específicos que se ajustan a un modelo de ciclo de vida espiral.

La mayoría de las organizaciones de desarrollo de software implementan metodologías para estandarizar el proceso y obtener un producto de calidad siempre dependiendo de las características implícitas dentro del contexto que se va a desarrollar que son variables ajustables como pueden ser cantidad de integrantes en el equipo de desarrollo, experiencias del mismo, acceso a la tecnología y presupuesto. Por tanto, es entendible que para lograr que se desarrolle lo que se tiene planteado como premisa en este trabajo se hago una selección sobre la metodología que acampañará al proceso de desarrollo y que más se ajuste a las características internas de este trabajo con tal de conseguir un software de calidad.

Metodologías de desarrollo de software.

Las metodologías de desarrollo de software surgieron en la década de los años 60. Estas se dividen en dos tipos de desarrollo de software: Tradicionales o Prescriptivas y Ágiles o Adaptativas [23].

Estas metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente, su principal objetivo es aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que todas deben ser adaptadas a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable [10].

Las metodologías de desarrollo de software tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el objetivo de conseguir un software más eficiente y predecible. Para ello, se hace un especial hincapié en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto. Este tipo de metodología no se adapta adecuadamente a los cambios, por lo que no son métodos convenientes cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar [23].

Según [23] en contraposición a estas metodologías tradicionales o prescriptivas, en la década de los años 90 apareció un nuevo grupo de metodologías, que se identifica como Ágil o Adaptativa. Esta nueva propuesta es buena cuando se trabaja con requisitos variables. Si no existen requisitos estables, no existe una gran posibilidad de tener un diseño estable y de seguir un proceso totalmente planificado, que no vaya a variar ni en

tiempo ni en dinero. En estas situaciones, un proceso adaptativo será mucho más efectivo que un proceso prescriptivo.

Por otra parte, al igual que en apartados anteriores [23] menciona que los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previos a la entrega final, lo cual agradará al cliente. Pero la mayor barrera que habrá que salvar será convencer al cliente de que no existen una planificación y una forma fija de hacer las cosas. En cualquier caso, lo que se garantiza es un menor riesgo ante la posibilidad de cambios en los requisitos.

Para la selección de la metodología de desarrollo se tuvo en cuenta que:

- Los requisitos pueden sufrir cambios durante el proceso de implementación.
- El cliente, en este caso el departamento de Seguridad Informática de la Universidad de las Ciencias Informáticas, está en constante comunicación con el equipo de desarrollo.
- Se cuenta con un equipo de desarrollo pequeño, debido a que está integrado por 2 estudiantes.
- Se conoce que la primera versión funcional del sistema se puede realizar en pocos meses.
- El equipo cuenta con una buena formación y capacidad de aprender.
- Existe la posibilidad de hacer entregas en mini versiones ya que se cuenta con el conocimiento y la tecnología necesaria por lo que se podría obtener resultados palpables para el cliente en pequeños plazos.
- El cliente tiene conocimiento acerca del tipo de software a realizar.
- El plazo de entrega es corto.

Analizando lo expuesto anteriormente se decide que la metodología utilizar debe ser ágil, ya que es la que más se ajusta a las características y propósito del trabajo.

Debido a las peculiaridades del trabajo para la definición de la metodología, después de varias entrevistas con los especialistas del área, los autores de la presente investigación tienen en cuenta un grupo de variables que son vitales para el software en cuestión. Dichas variables son: simplicidad, seguridad, excelencia técnica, colaboración y adaptabilidad.

Al consultar la bibliografía de las metodologías adaptativas, se decide tomar como base del análisis el estudio hecho por Letelier en [24]. En este artículo el autor expone una comparación entre distintas aproximaciones ágiles en base a tres parámetros:

la vista del sistema como algo cambiante, la colaboración entre los miembros del equipo y características específicas de las metodologías homologas a las variables predefinidas.

Como resultado, el mismo arroja que las metodologías más ágiles destacan **Adaptive Software Development (ASD)**, **Scrum** y **Extreme Programming (XP)** como se muestra en figura:

	CMM	ASD	Crystal	DSDM	FDD	LD	Scrum	XP
Sistema como algo cambiante	1	5	4	3	3	4	5	5
Colaboración	2	5	5	4	4	4	5	5
Características Metodología (CM)								
- Resultados	2	5	5	4	4	4	5	5
- Simplicidad	1	4	4	3	5	3	5	5
- Adaptabilidad	2	5	5	3	3	4	4	3
- Excelencia técnica	4	3	3	4	4	4	3	4
- Prácticas de colaboración	2	5	5	4	3	3	4	5
Media CM	2.2	4.4	4.4	3.6	3.8	3.6	4.2	4.4
Media Total	1.7	4.8	4.5	3.6	3.6	3.9	4.7	4.8

Fu*ente:*[24]

Después de analizar el resultado de Letelier en [24] , se creó una tabla comparativa entre estas tres metodologías como se muestra en la figura 2, evaluando en cada una de ellas la variables predefinidas. Cabe resaltar que al conjunto de variables los autores le asignaron un peso **(P*)** con el objetivo definir la importancia que tienen esta para la realización del presente trabajo. Como resultado se obtuvo que la metodología base sería

Figura 2 Ranking de agilidad de las metodologías.

XP, haciendo la salvedad de su adaptación en cuanto a varios aspectos que son vitales para el trabajo.

Peso (P)	Variables	ASD	Scrum	XP
1*	Colaboración	5	5	5
2*	Simplicidad	4	5	5
1*	Adaptabilidad	5	4	3
2*	Excelencia Técnica	3	3	4
3*	Seguridad	-	-	-
		4.8	5	5.2

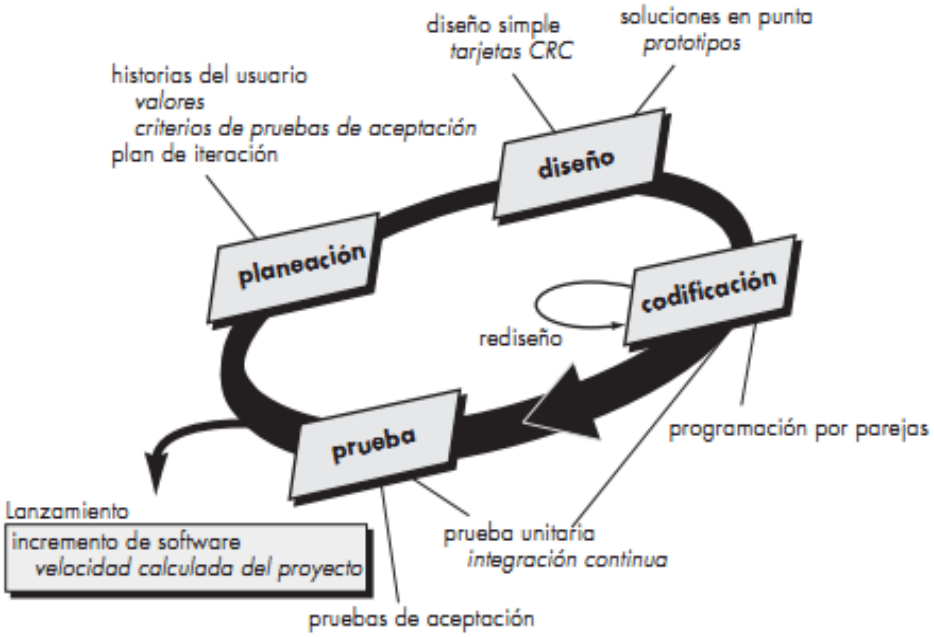
Figura 3 Comparativa en base a Variables Predefinidas.

Fuente: Elaboración personal

Extreme Programming según [25] es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Esta se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. La misma es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Según [10] el ciclo de vida de un proyecto XP incluye, al igual que las otras metodologías, entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente. Sin embargo, XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto.

Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP [10].



A continuación, el proceso de Extreme Programming:

Figura 4 Fases de XP.

Fuente: [10]

Según se muestra el proceso, XP no contempla en ninguna de sus fases la seguridad del sistema, por lo que se decide realizar un estudio bibliográfico de las principales metodologías enfocadas la seguridad de software.

Como resultado se encuentra el artículo [26], donde el autor realiza una comparación entre Microsoft SDL y CBYC las metodologías más usadas en cuanto a seguridad de software se refiere. En este documento el autor expone que Microsoft SDL contienen una suite de instalaciones de pago para su correcto funcionamiento y CBYC para su implementación recomienda el lenguaje spark.

Por otra parte, se encontró el artículo [27] que expone, los estándares de seguridad informática que existen tanto nacionales como internacionales y que estos a pesar de no poseer un diseño completo para llegar a ser una metodología poseen fuertes pautas y buenas prácticas para garantizar cinco pilares que se tienen que seguir para el desarrollo de esta aplicación web al igual que en cualquier otra, como son: integridad de los datos, la confidencialidad de la información, la disponibilidad del sistema, el no repudio de los usuarios y autenticidad de la información.

Analizando lo anteriormente expuesto las metodologías Microsoft SDL y CBYC no pueden ser usadas en su totalidad, por lo que los autores por las experiencias descritas en el artículo analizado deciden escoger dos fases de la metodología CBYC que garantizan la seguridad del sistema, así como incluir en la primera fase escogida los estándares de seguridad informática:

- La primera fase escogida es Fase de Especificación de los Módulos, en esta fase CBYC define el estado y el comportamiento encapsulado en cada módulo del software tomando en cuenta el flujo de información. Los módulos deberán de tener un enfoque de bajo acoplamiento y alta cohesión; así los efectos serían mínimos en caso de producirse una falla en el flujo de información.
- La segunda fase es la de especificaciones de las pruebas, donde CBYC toma en cuenta las características del software, los requerimientos y el diseño de alto nivel. En esta fase se efectúan pruebas de valores límites y pruebas de comportamiento. Cabe destacar que CBYC no hace pruebas de caja blanca ni pruebas de unidad, todas las pruebas son nivel de sistema y orientadas a las especificaciones.

A raíz del estudio evidenciado sobre el proceso de desarrollo de software y los aspectos metodológicos que lo guían se decidió escoger Extreme Programming como metodología base, realizándosele varias adaptaciones, incluyéndosele fases de otra metodología para que esta cumpla con todas las variables predefinidas anteriormente, y para lograr un mayor entendimiento en la presente investigación:

- Se decidió renombrar la Fase de Prueba del proceso original de XP a Fase de Prueba y Despliegue, ya que esta metodología originalmente no contempla en ninguna de sus fases el despliegue del sistema. Además de que esta fase es

de vital importancia para el desarrollo de la presente investigación, debido a que esta modela la arquitectura en tiempo de ejecución del sistema.

- Se elimina tanto de la Fase de Codificación como de Prueba y Despliegue la integración continua ya que para el desarrollo del sistema no se utiliza ninguna herramienta de integración continua.
- Se incluye en el proceso original de desarrollo de XP dos fases originarias de la metodología CBYC, específicamente estas dos fases estarían ubicadas en la fase de Diseño y Prueba y Despliegue respectivamente. Estas fases son: Fases de Especificación de Módulos y Fase de Especificación de Pruebas.
- Se añaden todos los estándares de seguridad informática tanto nacionales como internacionales a la Fase de Especificación de Módulos.

Como resultado de estas adaptaciones se obtiene un proceso de desarrollo de la metodología base más ajustado a las peculiaridades de la investigación como se muestra en figura 5, garantizándose la calidad y el correcto funcionamiento del producto final.

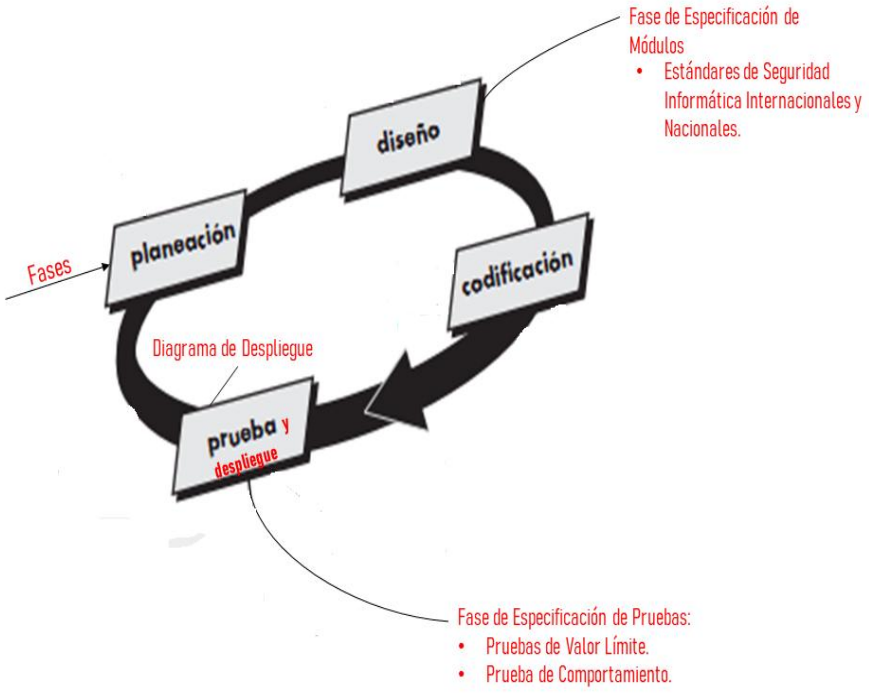


Figura 5 Adaptaciones al proceso de XP.

Fuente: Elaboración Personal.

1.5 Herramientas y tecnologías

Lenguajes de programación

Como lenguaje de programación para el backend se seleccionó Python en su versión 3.7.5. La decisión técnica de esta elección está dada por las características del lenguaje y cómo contribuye con la solución propuesta.

Según [28]**Python** es considerado como uno de los lenguajes más potentes en la actualidad debido a su simplicidad, es un lenguaje de propósito general lo cual significa que puede usarse para el desarrollo de aplicaciones Web o aplicaciones de escritorio.

Otra característica es que es multiparadigma, significa que puede usarse para la programación orientada a objetos y también para otros tipos de paradigmas como la programación funcional o imperativo.

La misma fuente enuncia otras características que se mencionan a continuación:

- Soporte para múltiples bases de datos, incluida la utilizada para la solución.
- Tiene soporte garantizado debido a la gran comunidad que posee, por lo que las dudas que puedan surgir en el aprendizaje del mismo pueden ser respondido disminuyendo la curva de aprendizaje.
- Multiplataforma, esta característica garantiza que el código escrita en Linux, puede ser reutilizado en otros sistemas operativos sin ningún problema.

Django como su framework, ya que este que fomenta el desarrollo rápido y el diseño limpio y pragmático. Creado por desarrolladores experimentados, se ocupa de gran parte de la molestia del desarrollo web, por lo que puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto

Por otra parte, se utilizó la herramienta Python BeautifulSoup para realizar el scrapping de las fuentes de información.

Como lenguaje de programación para el frontend JavaScript en su versión ECMAScript 6, específicamente la biblioteca React en su versión 16.12.0 para la construcción de interfaces de usuarios. Diseña vistas simples para cada estado en tu aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien. Crea componentes encapsulados que manejen su propio estado, y conviértelos en interfaces de usuario complejas. Ya que la lógica de los

componentes está escrita en JavaScript y no en plantillas, puedes pasar datos de forma sencilla a través de tu aplicación y mantener el estado fuera del DOM [29].

Lenguajes de representación gráfica

Como lenguaje de representación gráfica se seleccionó **UML** en su versión 2.0. Es el lenguaje de modelado de sistemas de software más conocido y usado en la actualidad. En la solución propuesta, aunque se usa como metodología de desarrollo XP y esta no define la creación de ningún artefacto que necesite modelarse con UML, se hizo necesario, usar este lenguaje para lograr una mejor compresión debido a que la modelación que permite se hace utilizando conceptos de programación orientada a objetos.

Lenguaje de Base de Datos

Existen un sinnúmero de gestores de bases de datos que clasifican para la solución, no obstante, siendo consecuentes con la política de usar tecnología libre se decidió el uso de **PostgreSQL** en su versión 11.0.2, entre otras cosas por ser el gestor de base de datos conocido por la autora.

Entorno Integrado de Desarrollo (IDE)

Los IDE seleccionados fueron el **PyCharm Professional** 2018.2, entre las tantas características basta con mencionar su integración con Python y **Webstrom** 2019.3.3 entre las tantas características basta con mencionar su integración con lenguajes como HTML, CSS, JAVASCRIPT, específicamente en con su biblioteca React.

Herramienta CASE

Como herramienta CASE se usó el **Visual Paradigm** en su versión 13.2. Esta herramienta es favorita de muchos desarrolladores porque entre sus características se encuentra el soporte UML y BPMN para los artefactos, además, de brindar la posibilidad de modelar artefactos de todas las etapas del ciclo de vida de un software.

En la solución propuesta se utilizó para modelar el diagrama de paquete y el diagrama de despliegue.

Herramienta de Diseño de Prototipo

Como herramienta para el diseño de prototipo se usó el **Balsamiq Mockups** en su versión 3.0. Esta herramienta es muy utilizada para el diseño de prototipos, ya que

permite diseñar de forma rápida y muy sencillas las maquetas de interfaces para webs y aplicaciones móviles.

1.6 Conclusiones parciales

El presente capítulo permitió llegar a las siguientes conclusiones parciales:

- La existencia de los NIDS y WAF como soluciones afines, no resuelven el problema en cuestión, aunque estas trabajen procesos similares, siguen careciendo de la actualización continua de sus reglas de asociación.

- El análisis de las diferentes metodologías de desarrollo de software permitió concluir que para aplicaciones web resulta necesario agregar elementos de la metodología CYBC y los distintos estándares de seguridad informática a la base de la metodología XP para así poder garantizar la seguridad, aspecto fundamental para la realización de la investigación. Por otra parte, resulta necesario agregar a la base de XP el despliegue del sistema debido a que este aspecto constituye la modelación de la arquitectura en período de realización del software.

- Las herramientas seleccionadas posibilitan la realización optima del sistema, garantizando la protección de los ataques externos a las aplicaciones web de la Universidad de las Ciencias Informáticas.

CAPÍTULO II: PROPUESTA DE SOLUCIÓN

Introducción

En este capítulo se describen los elementos asociados al análisis y diseño del software a desarrollar. Se detallan las fases de la metodología escogida y de cada una se mostrarán los fundamentos teóricos, herramientas y técnicas utilizadas.

2.1 Fase de Planeación

2.1.1 Descripción de la Solución

El sistema a desarrollar es autónomo[5] y contribuye a viabilizar los procesos de detección y bloqueo de ataques a aplicaciones web, a través de patrones de ataques ya conocidos. Para la obtención de los patrones se consultaron diferentes fuentes de información teniendo en cuanta los siguientes criterios de selección: fuentes públicas, de origen hispanas y de compartición de indicadores de Compromiso[6] (IOC); concluyendo que **Inteligencia de Fuentes Abiertas (OSINT)**, la **Plataforma de intercambio de información de malware (MISP)** y **Sitio Segu. Info** son las fuentes más favorables.

El portafolio de patrones encontrado se utiliza para actualizar el archivo.cfg y el archivo.xml del sistema de información de seguridad y gestión de eventos (SIEM) que utiliza la universidad, este portafolio el sistema también lo utiliza para actualizar su base de dato, con el objetivo de poder exportar listas negras de IP y un archivo.cfg que va a contener toda la información almacenada por el sistema en su base de dato. La figura 5 ilustra lo antes expuesto.

[5] Autónomo: para la presente investigación autónomo, significa que el funcionamiento del sistema es totalmente independiente a los otros sistemas que estén conectados a él.

[6] Indicadores de Compromiso: es la descripción de un incidente de ciberseguridad, actividad y/o artefacto malicioso mediante patrones para ser identificado en una red, pudiendo mejor así las capacidades antes la gestión de incidente.

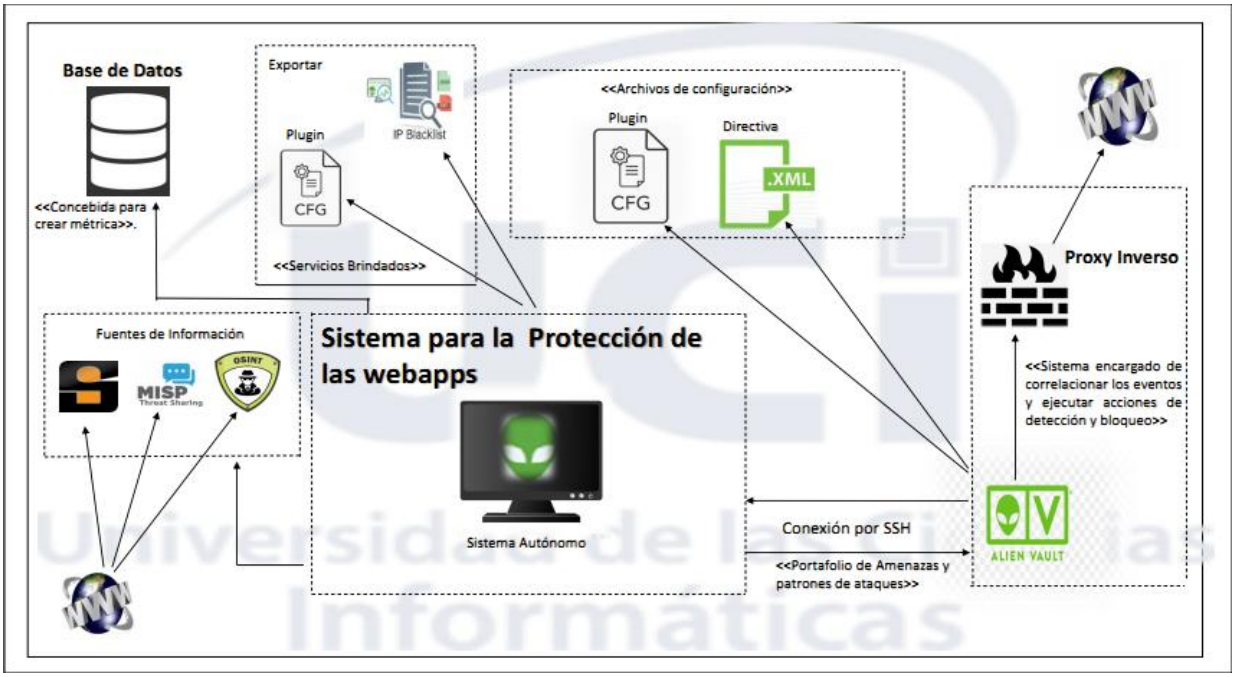


Figura 6 Descripción de la Propuesta Solución.

Fuente: Elaboración Personal.

2.1.2 Obtención de requerimiento

La tarea principal de la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.

Para la obtención o captura de requisitos en la propuesta de solución se emplearon las técnicas:

- **Entrevista:** es la técnica de licitación más utilizada, y de hecho son prácticamente inevitables en cualquier desarrollo ya que son una de las formas de comunicación más naturales entre personas [30]. Esta técnica tiene como objetivo obtener información de los involucrados, ya sea una o un grupo de personas, en el que el entrevistador se encarga de realizar preguntas pertinentes acerca del sistema y documenta las mismas. También se debe considerar que dependiendo del nivel de claridad proporcionado durante la entrevista la documentación resultante puede estar sujeta a la interpretación del que realizó la entrevista, por lo que la recopilación y análisis de los datos de la entrevista, puede ser un proceso complejo y costoso [31].

- **Lluvia de Ideas:** es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Las sesiones suelen estar formadas por un número de cuatro a diez participantes, uno de los cuales es el jefe de la sesión, encargado más de comenzar la sesión que de controlarla [30]. Esta técnica no tiene gran dificultad, es una técnica que permite generar ideas, sin embargo, se debe tomar en cuenta que se tiene que moderar la lluvia de ideas. Algunas herramientas para facilitar esta técnica son: los mapas mentales y diagramas de contexto [31].

Una vez escogidas las técnicas de captura de requerimientos, se comenzó la obtención de los requisitos funcionales (RF). Los RF son capacidades o condiciones que el sistema debe cumplir. Expresan una especificación detallada de las responsabilidades del sistema en cuestión y permiten determinar de una manera clara, lo que debe hacer el sistema [10]. Según la metodología escogida para la realización de esta investigación los RF son especificados utilizando la técnica de Historias de Usuario (HU).

Las HU son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad[32]. La presente investigación se identificaron 10 historias de usuarios para la propuesta de solución, la siguiente tabla se muestran cada uno.

Tabla 1: Listado de historias de usuarios.

#	Nombre de las historias de usuarios
HU1	Gestionar Usuario.
HU2	Autenticar Usuario.
HU3	CRUD Gestión Manual de Amenazas y patrones de ataques.
HU4	CRUD Gestión Directiva en OSSIM.
HU5	Consumir datos de fuentes de inteligencia de amenazas con datos estructurados.
HU6	Agregar patrón de ataque a plugin de OSSIM.
HU7	Eliminar patrón de ataque a plugin de OSSIM.
HU8	Modificar patrón de ataque a plugin de OSSIM
HU9	Exportar los nuevos patrones de ataque creados a fichero.cfg.
HU10	Exportar lista negra de IPs.

Fuente: *Elaboración Persona.*

Del listado anterior se muestran las especificaciones de la HU con tres ejemplos definidos para implementar la solución, el resto se encuentran en los anexos del documento.

Tabla 2: Agregar patrón de ataque a plugin de OSSIM.

Fuente: Elaboración Persona.

Historia de usuario	
Nombre: Agregar patrón de ataque a plugin de OSSIM.	**Número:** 6
Usuario: Usuario	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:**
Complejidad de desarrollo: alta	**Interfaz de usuario asociada:**
Descripción: Permite a partir de listado de patrones de ataques descargados por el usuario de las fuentes de información de datos estructurados, seleccionar y agregar al fichero de configuración de OSSIM esas amenazas de prioridad alta.	
Observaciones: N/A	

Tabla 3: CRUD Gestión Manual de Amenazas y patrones de ataques.

Historia de usuario	
Nombre: CRUD Gestión Manual de Amenazas y patrones de ataques.	**Número: 3**
Usuario: Usuario	**Iteración asignada:** 1
Prioridad en el negocio: alta	**Puntos estimados:**
Complejidad de desarrollo: alta	**Interfaz de usuario asociada: 2**
Descripción: Permite insertar, modificar, eliminar y listar las amenazas y patrones de ataques encontrados en fuentes de inteligencia.	
Observaciones: N/A	

*Fuente:*Elaboración Persona.

Tabla 4: Consumir datos de fuentes de inteligencia de amenazas con datos estructurados.

Fuente: Elaboración Persona.

Historia de usuario	
Nombre: Consumir datos de fuentes de inteligencia de amenazas con datos estructurados.	**Número: 4**
Usuario: Usuario	**Iteración asignada:** 3
Prioridad en el negocio: alta	**Puntos estimados:**

Complejidad de desarrollo: alta	**Interfaz de usuario asociada: 1**
Descripción: Permite que se listen al usuarios las amenazas y patrones de ataques que fueron detectados en fuentes de inteligencia.	
Observaciones: N/A	

Una vez que se definen las HU se crear un plan de iteraciones, donde cada una recibe un orden preestablecido que indica la iteración en que se desarrollarán, como se muestra en la tabla 5.

Tabla 5:Plan de Iteraciones.

Fuente: Elaboración Persona.

Iteraciones	Historias de usuario	Duración total (semanas)
1	Gestionar Usuario.	1
1	Autenticar Usuario.	1
1,2	CRUD Gestión Manual de Amenazas y patrones de ataques.	3
2	CRUD Gestión Directiva en OSSIM.	1
3	Consumir datos de fuentes de inteligencia de amenazas con datos estructurados.	2
3	Agregar patrón de ataque a plugin de OSSIM.	2
3	Eliminar patrón de ataque a plugin de OSSIM	1
3	Modificar patrón de ataque a plugin de OSSIM	1
4	Exportar los nuevos patrones de ataque creados a fichero. conf.	1
4	Exportar lista negra de IPs.	1
Total(Semanas)		14
Tiempo del Proyecto	*3 meses y 2 semanas.*	

Ya definidos los requisitos funcionales del sistema a desarrollar, se comienza con la captura y obtención de los requisitos no funcionales (RNF). Los requerimientos no funcionales, como su nombre sugieren, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste, como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento [33]. A continuación, se muestran los requisitos no funcionales que se han definido para el desarrollo de la herramienta por parte de los autores de la presente investigación:

De hardware y software:

- **RNF1-** La aplicación tiene que ser multiplataforma.

La computadora donde se va a ejecutar la aplicación debe tener los siguientes requisitos mínimos:

- **RNF2-** Memoria RAM de 256 Mb o superior.
- **RNF3-** Microprocesador dual Core a 1.2 GHz o superior.

Portabilidad:

- **RNF4-** El sistema será multiplataforma por lo cual se podrá utilizar en cualquier sistema operativo.

Disponibilidad:

- **RNF5-** El sistema podrá ser usado en cualquier momento por todos los usuarios autorizados las 24 horas del día los 7 días de la semana.

Usabilidad:

- **RNF6-** El sistema debe permitir al usuario conocer las acciones que puede realizar, a partir de íconos sugerente, alternativas textuales u otro elemento que le permita al usuario un mejor trabajo con el mismo.
- **RNF7-** El sistema debe poder brindar facilidad de uso a los usuarios.

Para poder hacer énfasis en los aspectos de seguridad debido a la sensibilidad que posee este tema en este trabajo se decidió emplear una serie de requisitos recomendados por, en donde se establece un estándar para la confección de un sistema que cumpla con las características de seguridad que garantizarían la estabilidad del mismo. Existen cinco principios fundamentales que todo sistema debería poseer y estos son [27]:

- Integridad: garantiza que los datos no sean modificados desde su creación sin autorización y que ningún intruso pueda capturar y modificar los datos en tránsito.
- Confidencialidad: garantiza que la información, almacenada en el sistema informático o transmitida por la red, solamente va a estar disponible para aquellas personas autorizadas a acceder.

- Disponibilidad: garantiza el correcto funcionamiento de los sistemas de información y su disponibilidad en todo momento para los usuarios autorizados.
- No repudio: garantiza la participación de las partes en una comunicación. El uso y/o modificación de la información por parte de un usuario debe ser irrefutable, es decir, que el usuario no puede negar dicha acción.
- Autenticación o Autenticidad: asegura que sólo los individuos autorizados tengan acceso a los recursos.

A continuación, se definen los requisitos no funcionales que acompañan a estos cinco principios:

Integridad:

- **RNF8-** Utilizar marcos de trabajo que previenen automáticamente los ataques XSS (Cross-Site Scripting o inyección de código malicioso).
- **RNF9-** Validar los datos que se reciben y velar por la integridad de los datos que se devuelven.

Confidencialidad:

- **RNF10-** Evitar mostrar mensajes con información que ayude a recopilar información sobre el producto o las configuraciones del servidor.
- **RNF11-** Evitar la elevación de privilegios en las cuentas de usuarios.
- **RNF13-** Evitar almacenar datos sensibles de manera innecesaria.
- **RNF14-** Deshabilitar el almacenamiento en caché de datos sensibles.

Disponibilidad:

- **RNF15-** Utilizar tecnologías seguras para el desarrollo**.**
- **RNF16-** Utilizar componentes únicamente de orígenes oficiales y utilizando los canales seguros.
- **RNF17-** Analizar riesgos y vulnerabilidades del entorno de despliegue del cliente atendiendo a sus características.

No Repudio:

- **RNF18-** Cifrar todos los datos en tránsito utilizando protocolos seguros.
- **RNF19**- Identificar o firmar de forma única los mensajes intercambiados.

- **RNF20-** Almacenar los mensajes intercambiados en ficheros de registros (logs por sus siglas en inglés) para su posterior consulta.

Autenticación o Autenticidad:

- **RNF21-** Utilizar controles contra contraseñas débiles.
- **RNF22-** Alinear la política de longitud, complejidad y rotación de las contraseñas establecidas.
- **RNF23-** Limitar el tiempo de respuesta de cada intento fallido de inicio de sesión.
- **RNF24-** Controlar el ciclo de vida de las contraseñas.

Ya definidos los requerimientos funcionales y no funcionales los autores de la presente investigación deciden realizar un estudio de las herramientas y tecnologías que van a dar soporte al desarrollo exitoso del sistema en cuestión.

2.2 Fase de Diseño

2.2.1 Tarjetas CRC (Clase-Responsabilidad-Colaboración)

Las tarjetas CRC son utilizadas para representar las responsabilidades de las clases y sus interacciones. Estas tarjetas permiten trabajar con una metodología basada en objetos, permitiendo que el equipo de desarrollo completo contribuya en la tarea del diseño. En cada tarjeta CRC el nombre de la clase se coloca a modo de título, las responsabilidades se colocan a la izquierda y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente. Dichos resultados se observan en las tablas 6, 7 y 8.

Tabla 6: Tarjeta CRC Template.

Fuente: Elaboración Persona.

Clase: Template	
Responsabilidad:	Colaboradores:
• Crea e inicializa su controlador asociado. • Obtiene datos del modelo. • Despliega la información al usuario. • Implementa el procedimiento Modificar (Update por sus siglas en ingles).	• Vista • Modelo

Tabla 7:Tarjeta CRC Modelo.

Fuente: Elaboración Persona.

Clase: Modelo	
Responsabilidad:	Colaboradores:
• Proporciona el núcleo de la funcionalidad de la aplicación. • Registra los Template y vistas dependientes. • Notifica a los componentes dependientes acerca de los cambios en los datos.	• Template • Vista

Tabla 8: Tarjeta CRC Vista.

Fuente: Elaboración Persona.

Clase: Vista	
Responsabilidad:	Colaboradores:
• Acepta las entradas del cliente como eventos. • Traduce eventos en solicitudes de servicio para el modelo o el Template. • Si se precisa. Implementa el procedimiento update.	• Template • Modelo

2.2.2 Arquitectura del sistema

La arquitectura de software alude a la estructura general del software y las formas en que la estructura proporciona una integridad conceptual para un sistema. En su forma más simple, la arquitectura es la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan, y la estructura de datos que utilizan los componentes. En sentido más amplio, sin embargo, los componentes pueden generalizarse para representar elementos importantes del sistema y sus interacciones [10].

Cada estructura general de un software contiene un estilo arquitectónico. Este constituye una transformación que se impone al diseño de todo el sistema. El objetivo es establecer una estructura para todos los componentes del sistema. En el caso en el que ha de hacerse la reingeniería de una arquitectura ya existente, la imposición de un estilo arquitectónico dará como resultado cambios fundamentales en la estructura del software, incluida la reasignación de las funciones de los componentes [10].

El estilo arquitectónico MVT es una modificación que realiza el framework Django al patrón de arquitectura de software Modelo-Vista-Controlador (MVC). La modificación consiste, en que el estilo MVC contempla al controlador como esa variable que gestiona el

flujo de información entre el modelo y la vista y su vez transforma los datos para que puedan ser adaptados según la necesidad de cada uno; esta es una característica que Django maneja internamente como framework, por lo que este concentra su atención en las plantillas ya que esta es la capa que contiene las decisiones relacionadas a la presentación. A continuación, se muestra la figura 6 explicando el uso de la arquitectura MVT en el desarrollo del sistema.

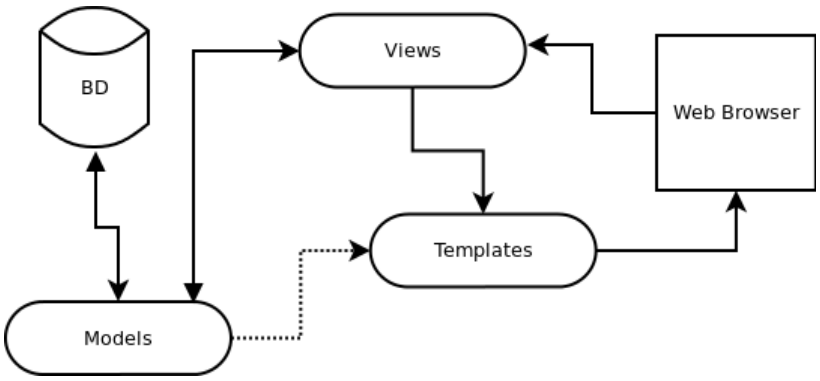


Figura 7 Diagrama de la arquitectura del Sistema.

Fuente: Elaboración Persona.

2.2.3 Especificación de Módulos

El diagrama de paquete permite dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales. Se pueden utilizar para plantear la arquitectura del sistema a nivel macro. Tiene como objetivo obtener una visión más clara del sistema de información orientado a objeto, organizándolos en subsistemas, agrupando los elementos del análisis, diseño o construcción y detallando las relaciones de dependencia entre ellos[34] .

Estrictamente los paquetes y sus dependencias son elementos de los diagramas de los diagramas de caso de uso, de clases y de componentes, por lo que se podrían decir que el diagrama de paquete es una extensión de estos. Estos paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento extremo este ellos [34].

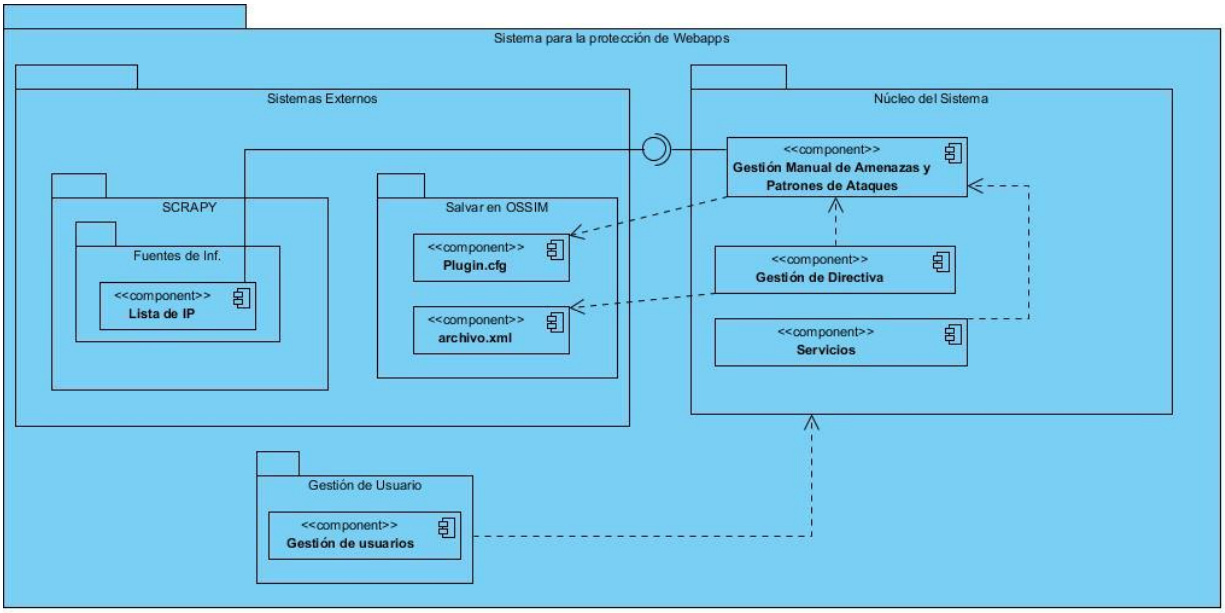


Figura 8 Diagrama de Paquete.

Fuente: Elaboración Persona.

2.3 Conclusiones parciales

- Al emplear las técnicas de obtención de requisitos posibilitó describir un compendio de requisitos funcionales expresados en historias de usuarios y de requerimientos no funcionales, identificándose entonces un listado de 24 requerimientos no funcionales y un total de 10 historias de usuarios.

- Los artefactos identificados en la fase de diseño posibilitaron que la construcción del sistema tuviera una estructura lógica de alto nivel y detalle.

- El uso de patrones de diseño permitió resolver problemas de implementación contextuales que determinaban en su solución que se pudiera tener un software reutilizable con módulos funcionales, independientes y con responsabilidades bien definidas, así como llegar a tener un resultado en donde se evidenciara una aplicación con seguridad de altos estándares.

CAPÍTULO III: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN

Introducción

En el presente capítulo se describe el estilo de codificación utilizado, se presenta el diagrama de despliegue y se realizan las pruebas al sistema.Se presentan los casos de prueba creados para validar las funcionalidades que responden a los requisitos identificados.

3.1 Codificación

3.1.1 Estándares de Codificación

Comprende todos los aspectos de la generación del código. Contribuyen al entendimiento del equipo de trabajo, limpieza del código y actividades de mantenimiento. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación permiten generar un código de alta calidad permitiendo al equipo de desarrollo efectuar mejor las revisiones posteriores. Ejemplo de estándares aplicados en el proyecto:

- Declaración
  - Número de declaraciones por línea: se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.
  - Inicialización: se debe inicializar cada variable en su declaración a menos que su valor inicial dependa de algún cálculo.
- Sentencias
  - Sentencias simples: cada
  - línea debe contener una sola sentencia.
- Convenciones de nombres: el prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, y debe ser uno de los nombres de dominio de alto nivel, actualmente com, edu, gov, mil, net, org, o uno de los códigos ingleses de dos letras que identifican cada país como se especifica en el ISO Standard 3166, 1981.
- Rompiendo líneas

- Romper después de una coma.
- Romper antes de un operador.

❖ Longitud de la línea: Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

❖ Comentarios: normas generales a seguir en toda documentación:

- Los comentarios deben añadir claridad al código. Deben contar el por qué y no el cómo.
- Deben ser concisos.
- Idealmente hay que escribir la documentación antes que el código.

❖ Asignación de nombres: esta es la sección más importante. Las normas genéricas son:

- Se usan descriptores en inglés que dejan claro el cometido de la variable, método o clase.
- Se usa terminología aplicable al dominio.
- Si se usan abreviaturas hay que mantener en algún sitio una lista de lo que significan.
- Evitar en lo posible los nombres largos (menos de 15 letras sería lo ideal).
- Evitar nombres que difieran en una letra o en el uso de mayúsculas.
- Un nombre no debería constar de más de dos palabras.
- No usar siglas en los nombres a menos que queden muy largos o sean siglas conocidas por todos.

3.1.2 Patrones de Diseño

Un patrón de diseño es una solución general re-utilizable, a un problema común dentro de un contexto en el diseño del software.

Según [35] se realiza una recopilación de 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Entre estos patrones se determinó la inclusión de los conocidos patrones GRASP debido a la sensibilidad que tiene la programación en el framework seleccionado pues utiliza como paradigma la programación orientada a objetos(POO), y este patrón interviene directamente brindando la solución a un problema planteado en este paradigma. ¿Cuál es el principio general para

asignar responsabilidades a los objetos?. Encontrándose su solución en asignar una responsabilidad al experto en información. A continuación, se refleja una tabla en donde se evidencia mejor estos argumentos:

Nombre del patrón	Problema que resuelve	Solución
Experto	¿Cuál es el principio fundamental mediante el cual se asignan responsabilidades a los objetos?	Asignar la responsabilidad a la clase que tiene la información necesaria para cumplirla.
Creador	¿Quién debe ser responsable de crear una instancia de alguna clase?	Asignarle a la clase C2 la responsabilidad de crear instancias de la clase C1 en uno de los siguientes casos: I. C2 agrega los objetos de C1. II. C2 contiene los objetos de C1. III. C2 registra las instancias de los objetos de C1. IV. C2 utiliza específicamente los objetos de C1. V. C2 tiene los datos de inicialización que serán transmitidos a C1 cuando este objeto sea creado. De iure, C2 es un creador de los objetos de C1.
Bajo acoplamiento	¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?	Asignar una responsabilidad para mantener bajo acoplamiento.
Alta cohesión	¿Cómo mantener la complejidad dentro de límites manejables?	Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Figura 9 Tabla de Problema-Solución Patrones Grasp.

Fuente:[36].

Por otra parte, siendo consecuentes con las herramientas pre planificadas, los patrones de diseño que utiliza el framework seleccionado por los autores, para el desarrollo de la presente investigación son, cabe aclarar que estos patrones ya están implementados por el framework. A continuación, se evidencia los patrones:

- Command Pattern: este patrón dentro del componente del framework se encarga específicamente de HttpRequest. Este encapsula una petición como un objeto, permitiendo parametrizar clientes con diferentes peticiones, en forma de cola o registro de eventos y soporta las operaciones un-do y re-do. A veces es necesario encapsular peticiones independientemente del tipo de petición o del cliente que la ejecuta [37].

- Observer Pattern: también conocido como Publisher-subscribers define una dependencia de uno a muchos entre objetos, de tal forma que cuando el estado de un objeto cambia, todos sus dependientes son notificados y actualizados automáticamente. Además de encargarse de las señales (Signals), que no es más que el mecanismo utilizado por el framework para transmitir el mensaje hacia los distintos objetos creados [37].

- Template method Pattern: este patrón se encarga de definir las vistas genéricas de las clases (Class-based generic views), este también es el esqueleto de un algoritmo en una operación, delegando alguno de los pasos a sus subclases. Este permite a las subclases redefinir algunos pasos de un algoritmo sin cambiar la estructura del algoritmo. Mediante la definición de alguno de los pasos de un algoritmo utilizando operaciones abstractas, este patrón fija su orden, pero deja a las subclases variar algunos pasos. Esto se evidencia cuando en el sistema de plantillas de Django a través de la herencia entre ellas, permitiendo reutilizar código [37].

3.2 Prueba y Despliegue

3.2.1 Pruebas

Las pruebas de software (en inglés software testing) intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. El proceso de prueba tiene dos metas definidas: demostrar al desarrollador y al cliente que el software cumple con los requerimientos, y encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación [10].

Según [10] un sistema debe pasar por tres etapas de pruebas:

- Pruebas de desarrollo: donde el sistema se pone a prueba durante el proceso para descubrir errores y defectos.
- Versiones de pruebas: donde un equipo de prueba por separado experimenta una versión completa del sistema antes de presentárselo al usuario final.

- Pruebas de usuario: donde los usuarios reales o potenciales del sistema prueban al sistema en su propio entorno.

Por lo importancia que recibe este epígrafe y siendo consecuente con la adaptación que se realizó a la metodología base a continuación se evidencian las pruebas aplicadas al sistema desarrollado.

Método de Caja Negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo en la interfaz del software. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del software [10].
Estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación [10].

Para llevar a cabo el método de Caja negra se utilizan la técnica de partición de equivalencia, el análisis de valor limites, la prueba a los RNF, la prueba de comportamiento y pruebas de aceptación, que se describen a continuación:

Técnica Partición de Equivalencia

Esta técnica divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El método se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada [10].
Para aplicar esta técnica, se deben primeramente realizar los Diseños de casos prueba (DCP) con el objetivo de obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Los DCP son un conjunto de condiciones o variables bajo las cuales un analista determinará si una aplicación o una característica de esta es parcial o completamente satisfactoria [10].

Como primer paso en el DCP, se muestra a continuación la descripción de las variables para el caso del requisito funcional:

Tabla 9:Descripción de variables de la HU: Crear Amenazas y Patrones de ataques.

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción	Clase de Equivalencia Válida	Clase de Equivalencia no Válida.
1	Dirección IP	Campo numérico	Si	Campo numérico que va a contar con el formato siguiente [**#. #. #. #]**. Los # varían entre 0 a 255.	1: Campo vació 2: **#.#.#. #**, siendo 0<= **#** =>255.	3: Valor no numérico. 4: Que el valor numérico no cumpla con el formato establecido: [**#.#.#. #**] y siendo 0<= **#** =>255.
2	Fecha	Campo de fecha	No	Campo de fecha que contiene el siguiente formato [día/ mes/ año: $\#_1$**:** $\#_2$**:** $\#_3$ **+** $\#_4$], siendo $\#_1$**,** $\#_2$**,** $\#_3$ números de dos cifras y $\#_4$ número de cuatro cifra.	5: El campo no puede estar vació. 6: Que el valor cumpla con el formato establecido: [día/ mes/ año: $\#_1$**:** $\#_2$**:** $\#_3$ **+** $\#_4$].	7: Que el valor introducido sea una cadena de caracteres.
3	Petición y Método HTTP.	Campo de texto	No	Campo de texto que contiene la petición realizada por el cliente y el método HTTP que pueden ser [**PUSH, GET, POST, DELETE**], siguiendo este formato: "[Método HTTP] http://...../.../.....? Protocolo/ Versión del Protocolo"	8: El campo no puede estar vació. 9: Que el valor cumpla con el formato establecido: "[Método HTTP] http://...../.../.....? Protocolo/ Versión del Protocolo".	10: Que el método HTTP no sean los indicados.
4	Respuesta del Servidor	Campo numérico	No	Campo numérico que contiene el código de estado [**# de tres cifras**] que el servidor devuelve al cliente.	11: El campo no puede estar vació. 12: Que el valor introducido sea un número de tres cifras.	13: Valor no numérico.

Fuente: Elaboración Persona

Análisis de Valor Límite

Para generar los casos de prueba, consideremos la técnica de Análisis de Valores Límite. Esta técnica se basa en la evidencia experimental de que los errores suelen aparecer con mayor probabilidad en los extremos de los campos de entrada [38], la misma conduce a que para determinadas clases de equivalencia se genere más de un caso de prueba, ya que representan un rango de valores para lo que la técnica utilizada indica que se generen dos casos de pruebas con el límite inferior y límite superior respectivamente, ejemplo de esto es la clase de equivalencia 1. Para identificar estos casos de pruebas se añadió el sufijo **a** y **b**a la clase de equivalencia correspondiente.

Tabla 10: DCP HU Crear Amenazas y Patrón de ataque.

Fuente: Elaboración Persona.

No Caso	Clase de Equivalencia	Dirección IP	Fecha	Petición y Método HTTP	Respuesta del Servidor	Resultado
1	4b,7,10,13.	300.299.304.290	Veinticuatro de mayo del dos mil dos	Select	ppoiuy	Campo 1 erróneo, Campo 2 erróneo, Campo 3 erróneo, Campo 4 erróneo.
2	4a,6,9,12.	-1	11/2/2000:11:33:66 + 0000	POST http://my.app.com/24h/status.xml? HTTP/2.1	400	Campo 1 erróneo.
3	3,7,9,12.	aaaaaaaaa	sdasd6798912asdsa	GET http://correo.estudiantes.uci.cu/18h/ index.html?HTTP/….	301	Campo 1 erróneo, Campo 2 erróneo.
4	1,7,10,12.		24 de febrero del 2016.	HEAD	199	Campo 2 erróneo, Campo 3 erróneo.
5	2,6,10.13	196.168.55.2	7/10/1996: 20:49:50 + 0000	Select	dkajsdjjj222	Campo 3 erróneo, Campo 4 erróneo.
6	2,6,9,12	10.0.0.1	3/1/1994: 29:49:59 + 0000	POST http:/www.humano.uci.cu/22h/fichero.xml? HTTP/2.1	405	Se agregaron exitosamente el patrón de ataque.

Prueba de comportamiento

Una vez generado el DCP HU: Crear Amenazas y Patrones de ataques, se considera aplicar las pruebas de comportamiento. Esta técnica se basa de analizar en un escenario sin necesidad de contar con el código fuente para así encontrar circunstancias donde el comportamiento del sistema desarrollado sea erróneo, indeseable o no esté de acuerdo con su declaración.

Para la aplicación de los casos de pruebas se realizaron 2 iteraciones para poder alcanzar un resultado satisfactorio atendiendo al correcto funcionamiento del software ante distintas situaciones. Se detectó en la primera iteración un total de 16 no conformidades (NC) y en la segunda un total de 1 NC.

Prueba de Aceptación

Las pruebas de aceptación se realizan para que el cliente certifique que el sistema cumple con sus especificaciones. La planificación detallada de estas pruebas debe haberse realizado en etapas tempranas del desarrollo del proyecto, con el objetivo de utilizar los resultados como indicador de su validez. Si se ejecutan las pruebas documentadas a satisfacción del cliente, el producto se considera correcto y, por lo tanto, adecuado para su utilización [39].

3.2.2 Despliegue

En el diagrama de despliegue se muestra cómo y dónde se despliega el sistema. Los elementos de diseño a nivel de despliegue indican como se ubicarán estos dentro del entorno computacional físico que soportará al software, los mismos pueden ser nodos, componentes y asociaciones [10].

Como la información manejada por la aplicación es bastante sensible se tomaron medidas para proteger la mismas de ataques informáticos. Para la sincronización con el sistema SIEM se utiliza el protocolo Cubierta Segura (SSH[7]) y para la comunicación con el servidor se utiliza el protocolo de transferencia de hipertexto (HTTP[8]).

[7] Cubierta Segura (Secure Shell, por sus siglas en inglés SSH) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura Cliente/servidor y que permita a los Usuarios conectarse a un host remotamente. Este protocolo encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

[8] Protocolo de transferencia de hipertexto (Hypertext Transfer Protocol, por sus siglas en inglés HTTP) en el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

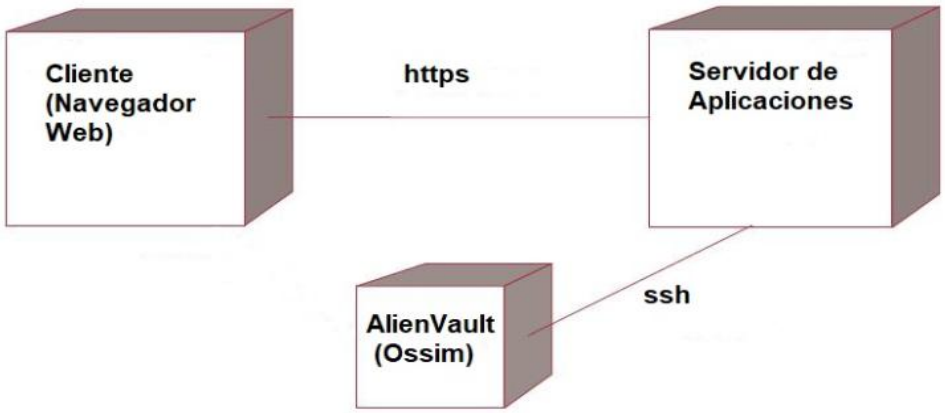


Figura 10 Diagrama de Despliegue.

Fuente: Elaboración Persona.

3.3 Conclusiones parciales

En este capítulo se abordaron los elementos de la implementación del sistema de protección de ataques externos a aplicaciones web en la Universidad de las Ciencias Informáticas, guiado por metodología de desarrollo Extreme Programming junto a las adaptaciones realizadas, así como las pruebas internas realizadas al sistema y los resultados arribando a las siguientes conclusiones:

- El diagrama de despliegue obtenido reflejó las relaciones existentes entre los elementos que intervienen en el proceso de digitalización de los expedientes.
- El correcto uso de los estándares de codificación permitió que el código del sistema desarrollado fuera legible para lograr una fácil comprensión, mayor organización y claridad del mismo.
- El proceso de pruebas de validación del software arrojó como resultado que el sistema implementado funciona correctamente en correspondencia con las solicitudes del cliente.

CONCLUSIONES GENERALES

En la realización de este trabajo se demostró la necesidad de desarrollar un sistema para la viabilización de los procesos de detección y bloqueo en ataques hechos a aplicaciones web en la Universidad de las Ciencias Informáticas, obteniéndose las siguientes conclusiones:

- El estudio de las condiciones en la Universidad además de una profundización en las soluciones existentes demostró que ninguno de estos sistemas daba solución a los problemas identificados por lo que se decide construir un sistema nuevo.
- La selección de lenguajes, tecnologías y herramientas permitió la implementación del sistema de detección y bloqueo guiado por la metodología de desarrollo base Extreme Programming, obteniéndose un software que satisface las exigencias funcionales y no funcionales definidas por el cliente.
- Los artefactos obtenidos en la etapa de planificación fueron de gran importancia para la construcción del sistema, facilitando una definición en detalle para permitir su interpretación e implementación.
- El proceso de validación del software arrojó como resultado que el sistema implementado funciona correctamente en correspondencia con las solicitudes del cliente, validado a través de las distintas pruebas realizadas además de la aceptación por parte del cliente.

RECOMENDACIONES

- Se recomienda el desarrollo de un módulo para generar conocimiento con los datos agrupados en las fuentes de información no estandarizadas a través de técnicas de inteligencia artificial.
- Se recomienda crear un repositorio estandarizado de ataques de uso nacional.

REFERENCIAS BIBLIOGRÁFICAS

[1] A. R. Garnacho, «Panorama actual de la ciberseguridad», *Economía industrial*, n.º 410, pp. 13–26, 2018.

[2] D. Fernández Bermejo y G. Martínez Atienza, *Ciberseguridad, ciberespacio y ciberdelincuencia*. Thomson Reuters Aranzadi, 2018.

[3] C. F. in S. on March 22, 2017, y 5:16 Am Pst, «Here are the top 6 ways websites get hacked, according to Google», *TechRepublic*. https://www.techrepublic.com/article/here-are-the-top-6-ways-websites-get-hacked-according-to-google/ (accedido mar. 06, 2020).

[4] «Estadísticas e informes». https://www.ccn-cert.cni.es/eu/component/tags/tag/estadisticas-e-informes.html (accedido mar. 06, 2020).

[5] «Advierten del crecimiento de las amenazas de denegación de servicio». https://www.ccn-cert.cni.es/eu/segurtasun-eguneratua/gaurkotasunari-buruzko-berriak/156-advierten-del-crecimiento-de-las-amenazas-de-denegacion-de-servicio.html (accedido mar. 06, 2020).

[6] «Hablando de ciberseguridad (IX)», *Cubadebate*, sep. 20, 2019. http://www.cubadebate.cu/especiales/2019/09/20/hablando-de-ciberseguridad-ix/ (accedido mar. 06, 2020).

[7] «ICAP condena ciberataque a sitio web que difunde la realidad de Cuba», *Radio Cadena Agramonte*. http://www.cadenagramonte.cu/articulos/ver/81639:icap-condena-ciberataque-a-sitio-web-que-difunde-la-realidad-de-cuba (accedido mar. 06, 2020).

[8] F. Rodríguez, «Registran en Cuba más de 600 incidentes de ciberseguridad este año • Trabajadores», *Trabajadores*, dic. 01, 2017. http://www.trabajadores.cu/20171130/registran-cuba-mas-600-incidentes-ciberseguridad-este-ano/ (accedido mar. 06, 2020).

[9] A. P. Olmos, «Análisis de la plataforma Ossim», p. 53.

[10] R. S. Pressman, *Ingeniería del software: un enfoque práctico*. 2013.

[11] «Concepto de detección - Definición en DeConceptos.com». https://deconceptos.com/general/deteccion (accedido feb. 21, 2020).

[12] «Definición de bloqueo — Definicion.de», *Definición.de*. https://definicion.de/bloqueo/ (accedido feb. 21, 2020).

[13] Ignacio Ramon, «Patrones de Ataques y de Seguridad como guia en el desarrollo de software», pp. 256-265, sep. 2015.

[14] Desarrollado en forma comunitaria, «CAPEC Common Attack Pattern Enumeration and Classification: “CAPEC List Version 2.0”», 2013, [En línea]. Disponible en: http://capec.mitre.org/.

[15] D. B. Pérez, H. R. G. Brito, y Y. S. Borrell, «Modelo para la detección de ataques a las aplicaciones WEB e intercambio de ciberamenazas.», *Revista Telemática*, vol. 17, n.º 2, pp. 71–80, 2019.

[16] P. M. Ferreira, «Techniques and tools for osint –based threat analysis», may 31, 2018.

[17] V. Prokhorenko, K.-K. R. Choo, y H. Ashman, «Web application protection techniques: A taxonomy», *Journal of Network and Computer Applications*, vol. 60, pp. 95–112, 2016.

[18] M. Roesch, «Snort: Lightweight intrusion detection for networks.», en *Lisa*, 1999, vol. 99, pp. 229–238.

[19] «Suricata sistema de deteccion de Intrusos», *Solvetic*. https://www.solvetic.com/tutoriales/article/2131-suricata-sistema-de-deteccion-de-intrusos/ (accedido feb. 21, 2020).

[20] «AQTRONiX». http://www.aqtronix.com/?PageID=99 (accedido feb. 21, 2020).

[21] «Citrix Web App Firewall - Web Application Firewall (WAF) - Citrix», *Citrix.com*. https://www.citrix.com/products/citrix-web-app-firewall/ (accedido feb. 21, 2020).

[22] P. Wongthongtham, E. Chang, T. Dillon, y I. Sommerville, «Development of a software engineering ontology for multisite software development», *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, n.º 8, pp. 1205–1217, 2008.

[23] P. Cáceres, E. Marcos, y G. Kybele, «Procesos ágiles para el desarrollo de aplicaciones Web», *Taller de Web Engineering de las Jornadas de Ingeniería del Software y Bases de Datos de*, vol. 2001, 2001.

[24] P. Letelier y M. C. Penadés, «Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)», 2012.

[25] K. Beck, *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[26] C. J. B. Abundis, «Metodologías para desarrollar software seguro», *ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica*, n.º 3, 2013.

[27] Y. N. Benitez y N. S. Martínez, «Requisitos de Seguridad para aplicaciones web», vol. 12, p. 17, 2018.

[28] «✪ Lenguaje de programación Python【aprende a programar】». https://lenguajesdeprogramacion.net/python/ (accedido mar. 06, 2020).

[29] «React – Una biblioteca de JavaScript para construir interfaces de usuario». https://es.reactjs.org/ (accedido mar. 05, 2020).

[30] A. D. Toro y B. B. Jiménez, «Metodología para la elicitación de requisitos de sistemas software», *Informe Técnico LSI-2000-10. Facultad de Informática y Estadística Universidad de Sevilla*, 2000.

[31] F. B. Sánchez, H. M. Venegas, y J. M. Romero, «Técnicas para el levantamiento de requerimientos en el desarrollo de un sistema de información», *Pistas Educativas*, vol. 36, n.º 114, 2018.

[32] M. P. Izaurralde, «Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario», *Trabajo de especialidad, Febrero*, 2013.

[33] I. Sommerville, *Ingeniería del software*. Pearson educación, 2005.

[34] «UML_clase_05_UML_paquetes.pdf». Accedido: ago. 27, 2020. [En línea]. Disponible en: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

[35] C. Larman, *UML y Patrones*. Pearson Educación ^ eMadrid Madrid, 2003.

[36] R. B. Tabares, «Patrones grasp y anti-patrones: un enfoque orientado a objetos desde logica de programacion», *Entre Ciencia e Ingeniería*, vol. 4, n.º 8, pp. 161–173, 2010.

[37] D. Gutierrez, «Framework y componentes», *Universidad de los Andes–Venezuela*, 2010.

[38] «tema1-pruebasSistemasSoftware.pdf». Accedido: ago. 27, 2020. [En línea]. Disponible en: https://ocw.unican.es/pluginfile.php/1408/course/section/1803/tema1-pruebasSistemasSoftware.pdf.

[39] I. R. Román y J. D. Cosín, *Técnicas Cuantitativas para la Gestión en la Ingeniería del Software.* Netbiblo, 2007.