Universidad de las Ciencias Informáticas

Facultad 2

Desarrollo de mecanismos de seguridad para el sistema Gestor de Recursos de Hardware y Software (GRHS)

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

Autores: Jorge Bárbaro Piñeiro Cruz

Javier Ricardo Ponce Pérez

Tutores: MSc. Ramón Alexander Anglada Martínez

Ing. Jenny de la Rosa Pasteur

La Habana, junio de 2016

DECLARACIÓN DE AUTORÍA

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

_____	_____
Jorge Bárbaro Piñeiro Cruz	Javier Ricardo Ponce Pérez
Firma del Autor	**Firma del Autor**
_____	_____
MSc. Ramón Alexander Anglada Martínez	Ing. Jenny de la Rosa Pasteur
Firma del Tutor	**Firma del Tutor**

"La productividad nunca es un accidente. Siempre es el resultado de un compromiso con la excelencia, una planificación inteligente y un esfuerzo enfocado"

Paul J. Meyer

AGRADECIMIENTOS

Jorge:

A mis padres, por el cariño y el amor que me han brindado toda la vida, por ser mis principales educadores, por ser el motivo de inspiración para llevar toda esta hazaña adelante y poder siempre contar con ellos.

A mi hermano, que tanto me ha aconsejado y sin su apoyo nada de esto fuera realidad, por ser mi ejemplo y guía ante la vida.

A toda mi familia, por brindarme su amor y apoyo incondicional.

A mi novia, por el apoyo constante y su paciencia.

A mi compañero de tesis, por su dedicación a la realización del trabajo.

A mis tutores, por su consagración y apoyo para que este trabajo resultara lo mejor posible.

Al tribunal y el oponente, por sus consejos que ayudaron a fortalecer el trabajo.

Al equipo de trabajo del proyecto GRHS, por su tiempo y ayuda.

A mis profesores, de toda la vida, por siempre educarme.

A mis compañeros del aula, con los cuales he recorrido estos cinco años, por su constancia.

Al movimiento ACM-ICPC, donde me hice de grandes amigos y sólidos conocimientos.

A mis amigos, los del barrio, los que crecieron conmigo, los de la escuela, por su presencia y apoyo.

AGRADECIMIENTOS

Javier:

Quiero agradecer a todos los que de una forma u otra ayudaron en la realización de este trabajo, muchas gracias. En especial a mis padres y mi hermano dado que son mi todo, gracias.

A mi compañero de tesis Jorge que siempre estuvo conmigo, cuando no había mucho que hacer, cuando no podíamos respirar del cúmulo, en general por ser el capitán que impidió que el barco se hundiera, gracias.

De forma especial a mis amigos que están aquí, que han pasado el día a día conmigo, que han amanecido conmigo, que me han cedido el puesto en la mesa, a todos gracias.

También a aquellos que no han podido graduarse como Israel, Raydel, Dayron, Gerardo, Ian, Harold, a todos gracias.

A mis tutores que han tenido que trabajar horas extras con este dúo tan cabezota, a esos que nos aconsejaron en cada momento y con los que tanto discutimos, por todo, gracias.

Al tribunal y oponente que han logrado que este trabajo tenga la calidad requerida a ellos, gracias.

DEDICATORIA

Jorge:

A mis padres que son mi inspiración y mis guías durante toda mi vida, les dedico este trabajo.

A mi hermano por ser sencillamente eso “mi hermano”.

A todos los que confiaron en mí de una forma u otra y sabían que era posible que llegara hasta aquí.

Javier:

Le dedico el trabajo a mi familia, principalmente a mis padres y hermano ellos que siempre me han apoyado y sacrificado, son los que me han querido más y son la razón de que haya podido graduarme, son mi fuerza mi voluntad y mi razón de ser, son los que se merecen este trabajo, solo espero que puedan mirarme y sentirse orgullosos del hijo y hermano que han criado, así como de orgulloso me siento yo tenerlos como padres y hermano.

Resumen

El presente trabajo describe la implementación de mecanismos de seguridad preventivos para mitigar amenazas de tipo suplantación, manipulación de datos y denegación de servicios presentes en el sistema Gestor de Recursos de Hardware y Software (GRHS) del centro de Telemática (TLM) en la Universidad de las Ciencias Informáticas (UCI). La solución propuesta se enmarca en la mitigación de las amenazas previamente definidas teniendo en cuenta el desarrollo de soluciones específicas para cada una, propiciando un mayor acople de la solución al sistema.

Como mecanismos de seguridad para la protección del código fuente escrito en Python se seleccionaron la compilación y el empaquetamiento; para la protección de la base de datos SQLite y los archivos de configuración, el cifrado de datos; para prevenir la suplantación de identidad y la manipulación de datos durante el proceso de comunicación, los certificados digitales y el protocolo SRP (Secure Remote Password); para contrarrestar la denegación de servicio y aumentar la capacidad de procesamiento del sistema, el establecimiento del protocolo de comunicación asíncrono AMQP(Advanced Message Queuing Protocol) y la implementación del procesamiento distribuido en el sistema.

Para llevar a cabo el desarrollo de la solución se definieron un conjunto de herramientas y tecnologías conjuntamente con la metodología de desarrollo software XP (Extreme Programming). Para validar el correcto funcionamiento de la solución se realizaron pruebas unitarias y de aceptación obteniéndose resultados satisfactorios.

Palabras clave: amenazas, cifrado, compilación, empaquetamiento, mecanismos, mitigación, seguridad.

Índice

Índice de Ilustraciones

Índice de Tablas

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) han tenido un auge en su desarrollo durante las últimas décadas. En este contexto el empleo de Sistemas Informáticos es clave para cualquier esfera de la sociedad. En el ámbito empresarial, su uso se ha incrementado considerablemente debido, en gran medida, al hecho de cada vez contar con un mayor volumen de información a gestionar.

La importancia de la información administrada en estos sistemas ha despertado gran interés en toda una gama de intrusos cibernéticos, que se han aprovechado de numerosos problemas de seguridad existentes en las aplicaciones, para llevar a cabo toda una serie de ataques como denegación de servicio, autenticación, suplantación, entre otros; es entonces de vital importancia disminuir las posibles vulnerabilidades que pudiesen existir en estos sistemas.

En la Universidad de las Ciencias Informáticas (UCI), la Dirección de Seguridad Informática (DSI) tiene entre sus funciones contribuir a garantizar el uso correcto de las tecnologías de la información. Para ello, dentro de los controles de seguridad informática que realiza, está el inventario de hardware y software, una medida recomendada internacionalmente [1]. La herramienta informática que utiliza para ello es el sistema Gestor de Recursos de Hardware y Software (GRHS).

GRHS es una plataforma desarrollada en Python que permite realizar el inventario de hardware y software de activos informáticos en redes de computadoras que tengan instalados sistemas operativos GNU/Linux o Windows. GRHS cuenta con un conjunto de colectores capaces de detectar cambios en la configuración de los equipos [2].

Esta plataforma cuenta con tres aplicaciones: *gclient,* que se encarga de la gestión, detección de cambios y notificaciones de los inventarios de las computadoras; *gserver,* responsable de la recepción y el almacenamiento de los inventarios; y el envío de configuraciones a los colectores (gclient), y *gadmin,* aplicación web donde se puede consultar la información de los inventarios y realizar las configuraciones del sistema [2].

El sistema GRHS posee vulnerabilidades de seguridad, esto puede comprometer toda la red de computadoras donde se encuentre instalado. Una de las principales

vulnerabilidades detectadas consiste en que el cliente cuando se instala en las estaciones de trabajo tiene pocos mecanismos para proteger su código fuente, por lo que este puede ser estudiado, modificado y luego un atacante podría comprometer la seguridad de todo el sistema GRHS.

Los archivos de configuración de la aplicación también carecen de protección. En este caso, el atacante podría modificar parámetros de los mismos dejando a la aplicación inutilizable total o parcialmente. Un elemento más a destacar es que el cliente manipula una base de datos local que puede ser consultada y modificada, para enviar información errónea hacia el servidor.

El protocolo utilizado para establecer la comunicación del cliente con el servidor no posee mecanismos para garantizar la autenticidad de los clientes que envían información hacia el servidor; esto trae consigo la posibilidad de que los clientes puedan ser suplantados y la información obtenida por el servidor sea errónea. A esto se añade el hecho de que el servidor procesa las peticiones de manera síncrona, lo cual provoca, en caso de tener una gran cantidad de peticiones a procesar, que no pueda ser accedido por varios clientes, incurriendo en una denegación de servicio.

De acuerdo con la situación problemática antes expuesta, en este trabajo se plantea como **problema de la investigación:** la insuficiencia de mecanismos de seguridad en el sistema Gestor de Recursos de Hardware y Software.

Como resultado del análisis del propio problema se definió como **objeto de estudio:** los mecanismos de seguridad para mitigar amenazas en aplicaciones informáticas; teniendo como **objetivo general:** desarrollar mecanismos de seguridad que mitiguen amenazas de tipo suplantación, manipulación de datos y denegación de servicios en el sistema Gestor de Recursos de Hardware y Software.

Se define para la investigación el siguiente **campo de acción:** los mecanismos de seguridad preventivos para mitigar amenazas de tipo suplantación, manipulación de datos y denegación de servicios.

Como **idea a defender** en la presente investigación se propone que: con la implementación de los mecanismos de seguridad propuestos se mitigarán las

amenazas de tipo suplantación, manipulación de datos y denegación de servicios en el sistema Gestor de Recursos de Hardware y Software.

Para dar cumplimiento al objetivo general, se planearon y ejecutaron las siguientes **tareas de investigación.**

1. Revisión de los principales conceptos, técnicas y mecanismos de seguridad con el objetivo de establecer un marco teórico para la investigación.
2. Análisis del GRHS en cuanto a la seguridad, con la finalidad de obtener la información necesaria para la posterior implementación de la solución.
3. Análisis de la metodología y herramientas de desarrollo de software a utilizar, para el desarrollo de la solución.
4. Selección de los mecanismos de seguridad que contribuyan a mitigar las amenazas de tipo suplantación, manipulación de datos y denegación de servicios que presenta el sistema GRHS, para su posterior implementación.
5. Análisis de las pruebas de software propuestas por la metodología de desarrollo de software seleccionada para comprobar el correcto funcionamiento de la solución.

Para llevar a cabo la investigación propuesta se han utilizado los siguientes métodos científicos de investigación:

Métodos teóricos

- **Analítico-Sintético:** permitió analizar los conocimientos generales sobre los mecanismos de seguridad, estableciendo la extracción de los elementos más significativos, arrojando ideas y conclusiones mucho más prácticas y concretas.
- **Inductivo-deductivo:** se utilizó para establecer una forma de razonamiento acerca de las particularidades de los mecanismos de seguridad, para reflejar sus puntos comunes.
- **Modelación:** Permitió modelar la realidad de la investigación, los principales elementos que la componen y su funcionamiento, lo cual ayudó a descubrir y estudiar nuevas cualidades y relaciones del objeto de estudio.

Métodos empíricos

- **Observación:** Se utilizó para percibir a partir de la situación real que se investiga, cómo se desarrollan groso modo los procesos que constituyen el objeto de estudio.
- **Entrevista:** Se aplicó una entrevista informal a varios de los desarrolladores del sistema para determinar las políticas de seguridad existentes. Las preguntas fueron referentes a los métodos de autenticación y arquitectura del sistema.

El presente trabajo se estructura en tres capítulos, los cuales son:

CAPÍTULO I: FUNDAMENTOS TEÓRICOS

Se abordan los elementos teóricos que sirven de base para la realización de toda la investigación; también se proporciona una descripción de las herramientas y metodología de desarrollo de software a utilizar para dar solución al problema planteado.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

Se brinda una explicación de la solución propuesta, enfatizando las características del sistema a desarrollar. Se detallan los artefactos generados durante las fases de exploración, planificación y diseño de la solución.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS

Se detallan los artefactos generados durante las fases de implementación y pruebas del sistema, validándose la solución mediante el uso de pruebas unitarias y de aceptación.

CAPÍTULO I: FUNDAMENTOS TEÓRICOS

En el presente capítulo se abordan conceptos relacionados con la seguridad informática, se identifican las amenazas que presenta el cliente de inventario, se presentan técnicas para la protección del código fuente, archivos de configuración y base de datos (SQLite). También se hace referencia a la comunicación asíncrona, el procesamiento distribuido y técnicas para verificar la autenticidad de la información. Además, se presentan las tecnologías, herramientas y la metodología de software empleada en el desarrollo de la solución.

1.1 Conceptos relacionados con la seguridad informática

En el marco del presente trabajo se consideran equivalentes los términos “seguridad de la información”, “seguridad informática” y “seguridad de las tecnologías de la información (TI)”, tal y como se plantea en la Resolución 127/2007 del Ministerio de la Informática y las Comunicaciones de Cuba (MIC) [3], la cual constituye el “Reglamento de seguridad para las tecnologías de la información” en el país.

Se define como **seguridad de la información** a la preservación de la confidencialidad, integridad y disponibilidad de la información [4]. Los atributos de la información previamente mencionados poseen las siguientes definiciones [4]:

- **Confidencialidad:** propiedad que determina que la información no esté disponible ni sea revelada a individuos, entidades o procesos no autorizados.
- **Integridad:** propiedad de salvaguardar la exactitud y el estado completo de la información.
- **Disponibilidad:** propiedad que determina que la información sea accesible y utilizable siempre que sea solicitada por una entidad autorizada.

En términos de seguridad informática una **vulnerabilidad** es la debilidad de un activo o control que puede ser explotada por una o más amenazas [4], siendo una **amenaza** la potencial causa de un incidente no deseado, que puede resultar en daños a un sistema u organización [4].

Un modelo útil para clasificar amenazas es el creado por Microsoft, **STRIDE**, derivado de un acrónimo de las siguientes seis categorías de amenaza [5]:

- **Spoofing identity (Suplantación de identidad):** intento de obtener acceso a un sistema informático mediante el uso de una identidad falsa. Después de que el atacante gana con éxito el acceso como usuario legítimo o de acogida, la elevación de privilegios puede comenzar.
- **Tampering with data (Manipulación de datos):** implica la modificación maliciosa de datos. Los ejemplos incluyen cambios no autorizados realizados en los datos persistentes, como en una base de datos, así como la alteración de los datos a medida que fluye entre dos ordenadores a través de una red abierta, como Internet.
- **Repudiation (Repudio):** se asocian con los usuarios que niegan que realizan una acción sin otras partes que tengan alguna forma de demostrar lo contrario, por ejemplo, un usuario realiza una operación ilegal en un sistema que carece de la capacidad de rastrear las operaciones prohibidas.
- **Information disclosure (Divulgación de la información):** implican la exposición de la información a las personas que no se supone que tienen acceso a ella, por ejemplo, la capacidad de los usuarios para leer un archivo al que no tienen acceso dado su nivel de privilegio, o la capacidad de un intruso para leer los datos en tránsito entre dos equipos.
- **Denial of service (Denegación de servicio) (DoS):** negar el servicio a usuarios válidos, por ejemplo, al hacer un servidor Web no disponible temporalmente o inutilizable.
- **Elevation of privilege (Elevación de privilegios):** un usuario sin privilegios gana acceso privilegiado y por lo tanto tiene acceso suficiente para poner en peligro o destruir todo el sistema. Las amenazas de elevación de privilegio incluyen aquellas situaciones en que un atacante ha penetrado de manera efectiva todas las defensas del sistema y forma parte del sistema de confianza.

Teniendo en cuenta el modelo STRIDE las amenazas detectadas en el sistema GRHS son:

- **Suplantación de identidad:** existe la posibilidad de que el código de los clientes sea modificado maliciosamente al ser de fácil acceso y comprensión por estar escrito en el lenguaje de programación Python, además de que el servidor no cuenta con un mecanismo que le permita establecer la autenticidad de los clientes.
- **Manipulación de datos:** la base de datos donde se almacena la información del cliente no posee ningún mecanismo que limite su acceso, por tanto, puede ser consultada y modificada, otro elemento a destacar es que dentro de la conexión HTTPS entre el cliente y el servidor, el cliente no posee un certificado SSL propio, lo cual permite que la comunicación pueda ser descifrada.
- **Denegación de servicio:** el sistema no posee un mecanismo que le permita controlar el flujo de datos que se envían hacia el servidor.

Las amenazas expuestas anteriormente se pueden materializar en un ataque**,** formalmente definido como cualquier acción que comprometa la seguridad de la información [6]**.** Como medida para contrarrestar las amenazas se utilizan los **mecanismos de seguridad** que se definen como mecanismos diseñados para detectar, prevenir, o recuperarse de un ataque [6]. Los mecanismos de seguridad de acuerdo a su función se clasifican en:

- **Preventivos:** actúan antes de que un hecho ocurra y su función es detener agentes no deseados.
- **Detectores:** actúan antes de que un hecho ocurra y su función es revelar la presencia de agentes no deseados en algún componente del sistema. Se caracterizan por enviar un aviso y registrar la incidencia.
- **Correctivos:** actúan luego de ocurrido el hecho y su función es corregir las consecuencias.

Los mecanismos de seguridad a tener en cuenta en el presente trabajo se encuentran dentro de la clasificación de preventivos, dado que el objetivo es reducir las posibilidades de que ocurran incidentes de seguridad.

1.2 Protección de código fuente

La protección del código fuente es uno de los elementos clave para la garantizar la seguridad de las aplicaciones informáticas, dado que limita a los atacantes de conocer el funcionamiento interno de la misma. Dentro de las técnicas utilizadas en el lenguaje Python para proteger el código fuente se encuentran:

- **Empaquetamiento de una aplicación:** se define como la capacidad de agrupar la aplicación, sus dependencias y obtener un intérprete apropiado, donde el usuario final solo tenga que ejecutar la aplicación [9].
- **Ofuscación de código fuente:** en el ámbito de la informática, ofuscar hace referencia a la modificación realizada deliberadamente en el código fuente de un software o programa con la finalidad de dificultar su inteligibilidad. En este sentido, la ofuscación del software tiene el objetivo de proteger el código original como medida de seguridad ante potenciales robos o plagios [7].
- **Compilar código fuente:** traducir un lenguaje de alto nivel a código absoluto o lenguaje binario [8].

1.2.1 Empaquetamiento de aplicaciones escritas en Python

Python es un lenguaje interpretado por lo que requiere de su propio intérprete para ser ejecutado, siendo esta la principal precondición antes de poder ejecutar un archivo con extensión py o pyc. Mientras que en GNU/Linux y BSD (y por extensión OS X) la inclusión de Python en el sistema operativo estándar se ha vuelto un hecho, Microsoft Windows no lo considera así. También está el hecho de las diferentes versiones y las dependencias estándar del lenguaje. Existen tres herramientas principales en uso para realizar el empaquetado dirigido a cada uno de los principales grupos de sistemas operativos. Para Windows el py2exe, para OS X el py2app y para GNU/Linux y BSD el cx_Freeze.

1.2.2 Técnicas para la ofuscación de código escrito en Python

La forma más simple de ofuscación y una muy a menudo utilizada es la distribución de los bytecodes como única forma de la aplicación, los descompiladores de Python, en general, pueden tratar fácilmente con esta forma de ofuscación. Esta técnica no se

considerará como un elemento a tener en cuenta para el presente trabajo, realizándose un estudio de las demás técnicas.

1.2.2.1 Empaquetar la aplicación, modificaciones al intérprete

Consiste en empaquetar la aplicación como primer paso y la alteración superficial de un atributo del intérprete, como por ejemplo una cadena con la versión del intérprete. Este es un método trivial de eludir, solo se necesita el entendimiento del sistema de empaquetado utilizado para tener acceso a los archivos del intérprete. Esto también claramente no protege a la aplicación contra el análisis estático realizado por los descompiladores de Python de los archivos pyc, una vez que hayan sido accedidos dentro del empaquetado [9].

1.2.2.2 Modificar la estructura de los bytecodes

El formato de un archivo bytecode de Python (pyc/pyo) contiene una serie de elementos dentro de los cuales se encuentra el número mágico que se utiliza para determinar la versión del intérprete correcto para ejecutar el bytecode, este cambia con cada versión del intérprete. Para CPython todos los valores conocidos actualmente se pueden encontrar en los comentarios de la parte superior de import.c. Una solución sencilla para este método es cambiar el número mágico para uno de los valores estándar. Si la versión exacta del bytecode no se sabe hay un pequeño conjunto suficiente de números mágicos válidos, por lo que se puede probar con cada uno de ellos hasta obtener el válido [9].

1.2.2.3 Cifrado de bytecodes

El cifrado de los bytecodes esencialmente se puede considerar como una de las variantes más complejas de cambiar el formato de clasificación. La diferencia clave es que la comprensión de los cambios es mucho más difícil por lo que requiere el acceso a algún tipo de clave incrustada en el intérprete de Python modificado, para tener acceso a las instrucciones de montaje. Una vez más, un cambio en el formato de los bytecodes, significa que el intérprete de Python estándar es incapaz de ejecutar el bytecode, y los descompiladores tradicionales no puede realizar su tarea. Un pequeño

número de casos de cifrado de bytecodes se ha visto, sin embargo, ninguno hasta ahora ha sido descifrado para validar una prueba positiva de esta suposición [9].

1.2.2.4 Reasignación de código de operación

La reasignación de código de operación consiste en que el intérprete Python modificado toma el valor estándar para la asociación de operaciones, el cual asocia un lenguaje ensamblador nemotécnico Python con un valor único, y lo intercambia con otro. Esto significa que incluso si el bytecode está disponible, un descompilador producirá una salida de código fuente incorrecta, dado que el flujo de bytecode se interpreta con significados incorrectos. En el estándar de Python el mapeo opcode se encuentra en el módulo opcode.py, que es bastante sencillo de interpretar [9].

La técnica de reasignación de código de operación se ha visto en numerosas ocasiones y ha demostrado ser una solución eficaz. Se utiliza a menudo junto con otras técnicas descritas anteriormente [9].

1.2.2.5 Ofuscación de código fuente

Otro tipo de ofuscación que se menciona para la integridad es la ofuscación del código fuente. Este enfoque no trata de alterar los bytecodes, pero crea una fuente alternativa que tiene una funcionalidad equivalente a la original, y es mucho más compleja de seguir, por lo general mediante el uso de varios tipos de segmentación funcional e indirecta. La filosofía se basa en el hecho “ten el código, no lo entenderás” [9]. Esta es una opción muy estudiada y existen varias bibliotecas para ello tales como:

- **Pyminifier:** es un módulo escrito en lenguaje Python que permite la ofuscación, compresión y minificación de código Python. Fue creado por Dan McDougall y se distribuye bajo licencia propietaria [10].
- **Pyobfuscate:** este transforma el código fuente claro y normalmente escrito en un nuevo código fuente que es difícil de leer, haciendo cambios en los espacios, nombres, comentarios, removiendo funciones, etc. Fue creado por Justin Herman y es una herramienta libre [11].

- **Htibctobf:** originalmente esta herramienta se escribió para resolver un reto en una competición de hacking en el congreso Hack in the Box. Htibctobf ofusca el código Python modificando los AST (Árbol de Sintaxis Abstracta) [12].

1.2.3 Compilar el código fuente a lenguaje C/C++

Como su nombre lo indica la técnica se basa en compilar el código de python al lenguaje de programación C/C++, el cual después se compila a código objeto, siendo este ilegible. Las formas recomendadas para utilizar esta técnica es mediante compiladores tales como:

- **Cython:** es una nueva extensión del lenguaje Python que apunta a realizar una integración más sencilla entre C y Python. En muchos sentidos, es un híbrido de estos. Cython está basado en Pyrex, el cual ha estado en uso algunos años, pero este ha perdido parte de sus funcionalidades y optimizaciones. Cython está desarrollado activamente y muchas funcionalidades faltan por ser agregadas, teniendo una documentación no siempre correcta. [13]
- **Nuitka:** traduce Python (2.6+) en un programa C++ que entonces usa “libpython” para ejecutarlo de la misma forma que CPython lo hace. Aunque Nuitka está todavía en desarrollo este genera el código nativo más eficiente. [14].

Cada una de las técnicas para proteger el código fuente anteriormente mencionadas ofrecen una solución al problema planteado en un menor o mayor grado. Teniendo en cuenta que la compilación incrementa la rapidez con que funciona la aplicación y protege el código fuente, se opta por esta variante a emplear conjuntamente con el empaquetado, eliminando con esta última técnica el problema de la portabilidad de la aplicación.

No se tuvo en cuenta la ofuscación del código fuente dado que los ofuscadores presentados solo se centran en un solo archivo, eliminando las dependencias de estos con los demás al renombrar las clases, variables y funciones. Dado que el compilador Cython ofrece la posibilidad de compilar el código Python al lenguaje C y de utilizar el lenguaje extendido Cython, se seleccionó el mismo como herramienta a utilizar para la

compilación. En el caso del empaquetamiento se seleccionó cx_Freeze por la característica de ser multiplataforma.

1.3 Protección de base de datos SQLite

Las bases de datos son un componente esencial de cualquier sistema informático, y mucha de la información almacenada requiere que se garantice que dicha información se mantenga confidencial, incluso para los administradores de la red. Algunas de las formas para lograr esto es mediante el uso de bibliotecas tales como:

- **The SQLite Encryption Extension (SEE):** es un complemento de la versión de dominio público de SQLite que permite leer y escribir archivos de bases de datos cifrados. Cuatro diferentes algoritmos de cifrado son soportados: RC4, AES-128 en modo OFB, AES-128 en modo CCM, AES-256 en modo OFB [15].
- **SQLCipher:** SQLCipher es una biblioteca de código abierto que proporciona un cifrado transparente y seguro, AES-256 bits, de los archivos de base de datos SQLite. SQLCipher ha sido adoptado como una solución de base de datos segura por muchos productos comerciales y de código abierto, por lo que es una de las plataformas de base de datos cifrados más populares para móvil y aplicaciones de escritorio [16].
- **SQLiteCrypt:** adiciona un cifrado AES-256 transparente para SQLite. Esta biblioteca en C implementa un motor de cifrado de base de datos SQL autónomo, integrable y sin necesidad de configuración, que le permite proteger sus datos en cualquier base de datos SQLite [17].

Teniendo en cuenta que las soluciones SEE y SQLiteCrypt requieren la compra de licencias y que la solución ofrecida por SQLCipher posee un cifrado seguro AES-256, se seleccionó a esta última como herramienta a utilizar.

1.4 Protección de archivos de configuración

Los archivos de configuración se encuentran en formato YAML y se utilizan a lo largo de la aplicación para regir el comportamiento de la misma. Como medio para protegerlos de cualquier modificación no autorizada se selecciona el cifrado de los mismos como técnica a utilizar.

Existen dos tipos básicos de cifrado: algoritmos simétricos: (también llamados "de clave secreta") utilizan la misma clave para el cifrado y el descifrado; algoritmos asimétricos: (también llamados "de clave pública") utilizan diferentes claves para el cifrado y el descifrado. Teniendo en cuenta que el acceso a los archivos de configuración está limitado solo al cliente de inventario se utilizará un algoritmo simétrico dado que son más rápidos, consumen menos recursos y en este caso en particular se elimina una gran desventaja frente a los asimétricos: el transporte seguro de la clave.

Python posee varios módulos para trabajar con algoritmos de cifrado, tanto simétricos como asimétricos, dentro de los cuales se encuentran:

- **PyCrypto - The Python Cryptography Toolkit:** es una colección de funciones de hash seguras (como SHA256 y RIPEMD160), y varios algoritmos de cifrado (AES, DES, RSA, ElGamal, etc.) [18].
- **Cryptography:** es un módulo que ofrece algoritmos de cifrado y primitivas a los desarrolladores de Python. Incluye sistemas de cifrado simétrico, funciones resumen y funciones de derivación de claves [18].

Para realizar el cifrado se seleccionó la herramienta PyCrypto dado que su implementación es modular y el compendio de algoritmos se restringe a los comúnmente utilizados en aplicaciones de seguridad. Como algoritmo se seleccionó el simétrico AES (Advanced Encryption Standard) en modo CBC (Cipher Block Chaining) obteniendo un cifrado seguro con 256 bits de clave.

1.5 Procesamiento de datos

El procesamiento de las peticiones que envían los clientes hacia el servidor se realiza de forma síncrona, no distribuida y en conjunto con las demás enviadas por la interfaz de administración (gadmin). Esto limita la capacidad de procesamiento del servidor, siendo en algunos casos, causa de una denegación de servicio no intencionada. Dos de las herramientas utilizadas en Python para realizar procesamiento distribuido y asíncrono se detallan a continuación:

- **Gearman:** provee un marco de trabajo genérico para trabajar con otras computadoras o procesos que se adaptan mejor para hacer el trabajo. Esta biblioteca permite realizar el trabajo en paralelo, tener un equilibrio en el procesamiento, y llamar funciones entre diferentes lenguajes. Esta biblioteca puede ser usada en diversidad de aplicaciones, para sitios web de alta disponibilidad y para el transporte de eventos de réplica de bases de datos [19].
- **Celery:** permite crear tareas de trabajo asíncronas manejadas por un gestor de colas que está basado en el envío de mensajes de manera distribuida. Se focaliza en operaciones en tiempo real pero también soporta la planificación de tareas, es decir, que puede ejecutar tareas en un momento determinado o de manera periódica. Las unidades de ejecución, llamadas tareas, se ejecutan de manera concurrente en uno o más nodos de trabajo. Estas tareas pueden ejecutarse tanto asíncrona como síncronamente, siendo el sistema en general altamente configurable [20].

De los sistemas distribuidos anteriormente mencionados se selecciona Celery dado su capacidad para procesar las tareas de manera asíncrona y concurrente y su fácil integración con el marco de trabajo web Django, utilizado en el servidor.

1.6 Protocolo de comunicación

El sistema GRHS utiliza el protocolo de comunicación síncrono HTTPS (Hypertext Transfer Protocol Secure) como vía de comunicación entre los clientes y el servidor de inventario. Teniendo en cuenta que el procesamiento de la información suministrada por el cliente de inventario se realiza de manera síncrona en el servidor, es necesario cambiar el protocolo de comunicación por uno asíncrono que permita enviar la información a procesar y posteriormente obtener el resultado, sin esperar el tiempo necesario para la realización del mismo.

Un elemento más a destacar es la necesidad de contar con una herramienta que permita regular el flujo de información que envían los clientes hacia el servidor. Esta herramienta deberá actuar como un intermediario que almacene la información hasta que sea solicitada por la otra parte interesada. Dada la compatibilidad con el sistema

distribuido Celery se propone utilizar como intermediario de mensajes al servidor de mensajería Rabbitmq, el cual implementa el protocolo de comunicación asíncrono AMQP (Advanced Message Queuing Protocol), los cuales se detallan a continuación.

AMQP (Advanced Message Queuing Protocol): es un protocolo de estándar abierto que trabaja en la capa de aplicaciones del modelo OSI. Las características que definen al protocolo AMQP son la orientación a mensajes, encolamiento ("queuing"), enrutamiento (tanto punto-a-punto como publicación-subscripción), exactitud y seguridad [21].

Rabbitmq: es un intermediario de mensajes de código abierto. Este brinda a las aplicaciones una plataforma común para mandar y recibir mensajes, y a los mensajes un entorno seguro para persistir hasta que estos sean recibidos, este utiliza AMQP como protocolo de comunicación [22].

1.7 Técnicas para verificar la autenticidad de aplicaciones

Uno de los elementos esenciales dentro del proceso de comunicación es validar la autenticidad de las partes involucradas. A continuación, se detallan dos de las técnicas utilizadas para ello.

1.7.1 Certificados digitales

El Certificado Digital permite verificar la identidad de un ciudadano, garantizando que únicamente él puede acceder a su información personal, evitando suplantaciones. Los certificados digitales también son utilizados en el proceso de validación de identidad en aplicaciones informáticas. La identificación basada en un certificado digital es equivalente a la presentación del DNI electrónico en la atención presencial [23].

En otras palabras, un certificado digital es un documento digital concedido por una Autoridad Certificadora (AC) que garantiza la asociación de personas o entidades físicas o jurídicas con uno o varios elementos técnicos que vinculan la parte digital o electrónica con la física.

Dada la necesidad de implementar un protocolo de comunicación seguro, se decide usar certificados digitales generados por la Autoridad Certificadora de la UCI para

realizar procesos de autenticación del cliente de inventario, garantizando la identidad de los clientes y la seguridad de la información enviada entre estos y el servidor.

1.7.2 Contraseña Remota Segura (SRP)

El protocolo SRP realiza la autenticación remota y segura de contraseñas humano-memorizables cortas y resiste tanto a los ataques de red pasivos como activos. Debido a que SRP ofrece está combinación única de seguridad de la contraseña, la comodidad del usuario, y la ausencia de licencias restrictivas, es el protocolo más estandarizado de su tipo, y como resultado está siendo utilizado por una gran variedad de organizaciones, comerciales y de código abierto, para asegurar casi todos los tipos de tráfico de red humano-autenticado en una variedad de plataformas informáticas [24].
El protocolo SRP se selecciona como método para iniciar sesiones de comunicación entre el cliente y el servidor de inventario, dado que este proporciona un token de seguridad, el cual puede ser almacenado y consultado durante la sesión abierta

1.8 Tecnologías y herramientas para el desarrollo de la solución

1.8.1 Metodología de desarrollo de software XP (Extreme Programming)

La programación extrema se caracteriza por ser una metodología ágil que facilita a los desarrolladores expertos la respuesta a las necesidades expuestas a cambios de los clientes, incluso a finales del ciclo de vida, es un proceso ligero, ágil, flexible; está orientado hacia quien produce y usa el software. Se encuentra especialmente bien documentada con innumerables recursos en línea disponibles, comunidades libres y grupos de noticias y encontrándose una gran cantidad de proyectos realizados con dicha tecnología.

Formulada en 1999 por Kent Beck, la programación extrema es una forma ligera, eficiente, de bajo riesgo, flexible, previsible y científica de desarrollar software [25].

Para guiar el desarrollo de la solución se decide asumir la metodología XP basado en los aspectos vistos anteriormente y los que se relacionan a continuación:

- El cliente es parte del equipo de desarrollo.

- No existe un contrato tradicional al que darle cumplimiento.
- Es un proyecto considerado pequeño.
- El equipo de desarrollo es pequeño.

1.8.2 VIM

Vim es un editor de texto muy configurable construido para permitir la edición eficiente de texto. Se trata de una versión mejorada del editor vi distribuido con la mayoría de los sistemas UNIX. Vim es a menudo llamado como un "editor para programadores," y tan útil para la programación que muchos consideran que es todo un IDE. Sim embargo no es sólo para los programadores. Vim es perfecto para todo tipo de edición de texto, desde la composición de correo electrónico hasta la edición de archivos de configuración. Vim puede ser configurado para trabajar de una manera muy sencilla [26].

1.8.3 Python

El lenguaje de Python fue creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, multiplataforma y orientado a objetos. Este permite dividir el programa en módulos reutilizables desde otros programas. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). Python se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. [8]

1.8.4 Framework Django 1.4.5

Django es un framework web de alto nivel para Python que fomenta el rápido desarrollo y diseño limpio y pragmático. Construido por desarrolladores con experiencia, este encarga de gran parte de la molestia para el desarrollo Web, para que el desarrollador pueda centrarse en el desarrollo de su aplicación sin necesidad de reinventar la rueda.

Es de código abierto y libre. Django toma en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores comunes de seguridad. [27]

1.8.5 Sistema Gestor de Base de Datos (SGBD). PostgreSQL 9.1

PostgreSQL es una Gestor de Base de datos que ha sido desarrollado en varias formas desde 1977. Comenzó como un proyecto llamado Ingres en la Universidad de California en Berkeley. PostgreSQL es ampliamente considerado el más avanzado Sistema Gestor de Bases de Datos de código abierto en el mundo. Este provee características que tradicionalmente solo son vistas en productos comerciales de calibre empresarial. [28]

1.8.6 InstallJammer

Es un instalador multiplataforma con interfaz gráfica diseñado para funcionar en Windows y sobre casi todas las versiones de UNIX. InstallJammer soporta un constructor de instaladores muy potente el cual posee múltiples temas y un alto nivel de configuración. Los instaladores son construidos como archivos ejecutables para una distribución en la web y manejar la instalación de todo lo que se desee [29].

1.8.7 Jenkins

Es un motor de automatización con plugins para el desarrollo de herramientas donde su objetivo es la integración continua, las prueba automatizadas y el suministro continuo Es principalmente un conjunto de clases de Java que modelan conceptos de un sistema de construcción en línea recta. Posee interfaces y clases que modelan el código que realiza parte de la construcción tales como: SCM para el acceso al sistema de control del código fuente ANT: para realizar la construcción y MAILER: para el envío de notificaciones por email [30].

1.9 Conclusiones del capítulo

Teniendo en cuenta lo descrito en el desarrollo del capítulo se concluye que las amenazas a las que se encuentra expuesto el sistema GRHS son la suplantación de identidad, la manipulación de datos y la denegación de servicios. Para contrarrestar

estas amenazas se estipuló la construcción de mecanismos se seguridad preventivos los cuales se seleccionaron a partir de varias propuestas.

Como mecanismos para la protección del código fuente se seleccionaron la compilación y el empaquetamiento; para la protección de la base de datos y los archivos de configuración, el cifrado de datos; para prevenir la suplantación de identidad, los certificados digitales y el protocolo SRP; para contrarrestar la denegación de servicio, el protocolo de comunicación asíncrono AMQP y el procesamiento distribuido.

Para llevar a cabo el desarrollo de la solución definieron un conjunto de herramientas y tecnologías conjuntamente con la metodología de desarrollo software XP. Las herramientas y tecnologías propuestas se seleccionaron teniendo en cuenta su compatibilidad con las utilizadas para el desarrollo del sistema GRHS, propiciando mantener la línea de desarrollo.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se abordan temas relacionados con la propuesta de solución para responder a la situación problemática en cuestión, se presentan los resultados obtenidos en las fases de Exploración, Planificación y Diseño correspondientes a la metodología de desarrollo (XP) para la implementación de la solución que se propone, donde se exponen los artefactos generados durante el transcurso de las mismas.

2.1 Propuesta de solución

Teniendo en cuenta que se está en presencia de un sistema (GRHS) en explotación, con una filosofía de desarrollo previamente definida, y que las vulnerabilidades detectadas en el mismo se encuentran en diferentes ámbitos de la aplicación; la solución propuesta se basa en el desarrollo de soluciones específicas para cada vulnerabilidad con el fin de hacer posible un mayor acople con el sistema y minimizar la cantidad de cambios en el mismo.

Los principales elementos a tener en cuenta en el desarrollo de la solución son: la protección del código fuente, la protección de los archivos de configuración, la protección de la base de datos del cliente de inventario, la gestión de la configuración del cliente de inventario, la autenticidad de los datos que son enviados hacia el servidor, el procesamiento de los datos en el servidor y la generación automática de instaladores del cliente de inventario.

En el caso de la protección del código fuente se plantea como propuesta la realización de un proceso que consta de dos pasos: el empaquetado del código fuente con cx_Freeze y la compilación del mismo por medio de Cython. El primer paso además de proteger el código fuente proporciona la portabilidad del mismo y la posibilidad de generar un instalador mediante InstallJammer.

Para la protección de los archivos de configuración se escoge el cifrado de los mismos como solución. Este se realiza utilizando el algoritmo simétrico AES (Advanced Encryption Standard) en modo CBC (Cipher Block Chaining) con 256 bits de clave,

mediante la librería PyCrypto, permitiendo un cifrado seguro y transparente. (Ver Ilustración 1)

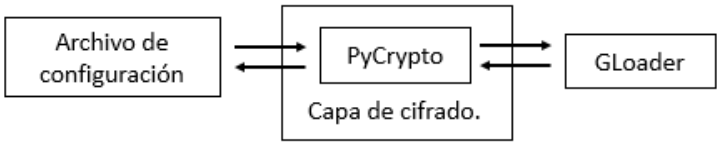


Ilustración 1. Protección de los archivos de configuración.

Para facilitar el desarrollo de los procesos de protección del código fuente y archivos de configuración se utilizará el integrador continuo Jenkins permitiendo la realización automática de los mismos. En este se programarán las tareas de empaquetamiento, compilación, cifrado de archivos de configuración y la generación automática de instaladores. (Ver Ilustración 2)

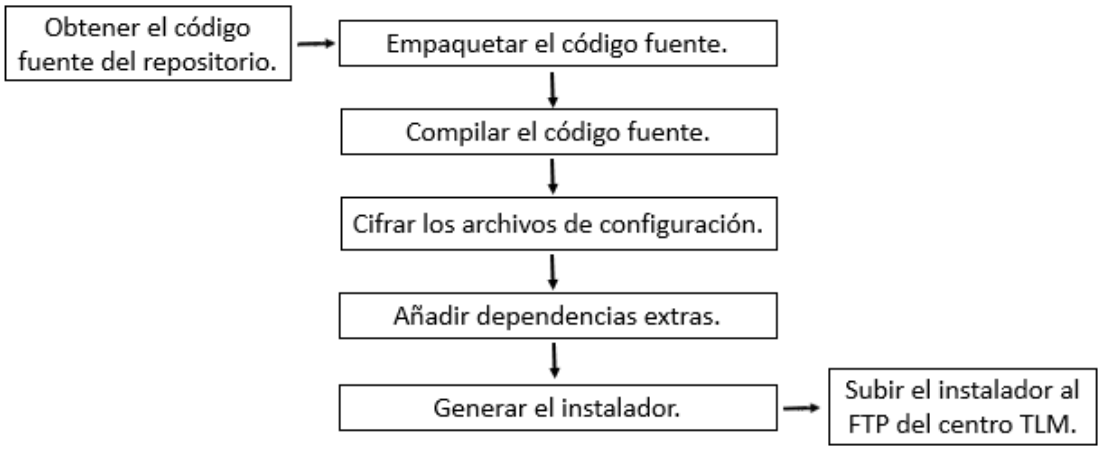


Ilustración 2. Tareas a ejecutar en el integrador continuo Jenkins.

La protección de la base de datos se gestiona mediante la biblioteca SQLCipher que proporciona el cifrado total de la misma utilizando el algoritmo propuesto para la protección de los archivos de configuración. Para su acople al sistema se utilizó el módulo Pysqlcipher; el cual se integra con Peewee, ORM (Object Relational Mapping) utilizado por el cliente de inventario. (Ver Ilustración 3)

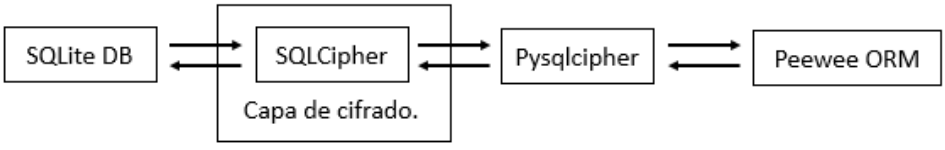


Ilustración 3. Protección de la base de datos.

Los archivos de configuración serán gestionados mediante el uso de un CLI (Command Line Interface) en el caso de los sistemas que no posean entornos de escritorio; en el caso contrario se utilizará una aplicación con GUI (Graphical User Interface) por medio de la librería PyQt; está ultima permitirá además visualizar el inventario y estará disponible como acceso directo en la barra de tareas. Se decide desarrollar estas interfaces dado que al cifrar los archivos de configuración estos no se podrán modificar manualmente como se hacía anteriormente.

Para lograr un mayor balance de carga en el procesamiento de los datos enviados por los clientes hacia el servidor estos se manejarán de forma asíncrona y distribuida haciendo uso del sistema distribuido Celery. Como fuente de almacenamiento para los datos y medio de comunicación entre las aplicaciones se utilizará el servidor de mensajería Rabbitmq, el cual implementa el protocolo AMQP (Advanced Message Queueing Protocol). (Ver Ilustración 4)

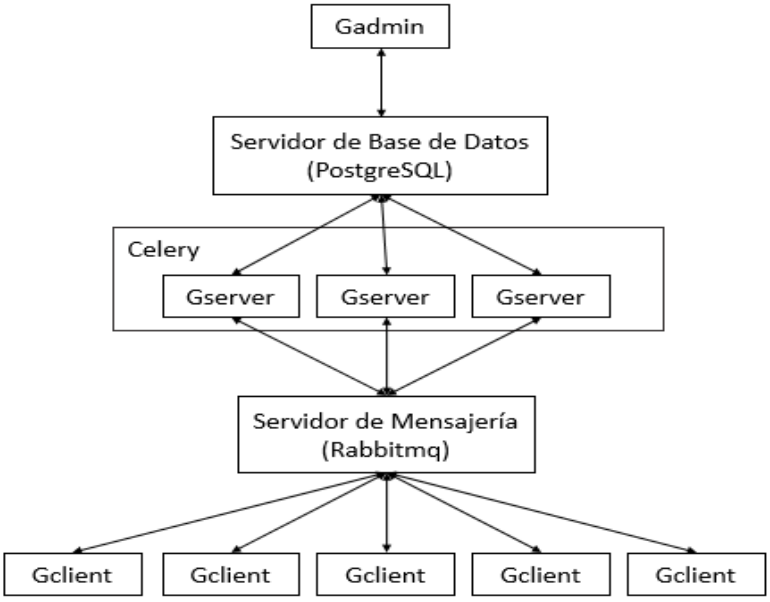


Ilustración 4. Flujo de comunicación entre cliente y servidor.

Para garantizar la seguridad de los datos que circulan entre el cliente y el servidor de inventario se utilizará una conexión SSL entre el servidor de mensajería y el sistema GRHS. Además, por medio de Celery se incorporará otro cifrado de datos a nivel de aplicación que fortalecerá la medida anterior. Los certificados SSL utilizados serán

generados por una CA (Certificate Authority) propia, permitiendo validar también su autenticidad en el momento de establecer las conexiones.

Con el objetivo de validar la autenticidad de los clientes de inventario se establecerá una negociación (handshake) entre el servidor y el cliente de inventario al inicio de cada conexión en donde se autenticarán las partes. El mismo se realizará utilizando el protocolo SRP (Secure Remote Password) el cual proveerá un token de autenticación para el posterior envío de datos, teniendo en cuenta que el token será solamente válido durante la sesión iniciada. Como medio para prever cualquier envío no autorizado de información todos los datos enviados serán interceptados y comprobados antes de procesarse.

2.2 Fases de Exploración, Planificación y Diseño de la metodología XP

2.2.1 Exploración

En esta fase, se plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.2.1.1 Historias de usuario

Una historia de usuario es una simple descripción de una funcionalidad del software por la que el cliente va a pagar. Los detalles extra son agregados cuando la historia de usuario es implementada, siendo estos procesos de requerimientos simples y sencillos.

Entre las principales características que debe tener una buena historia de usuario se encuentran: la historia debe ser entendida por el cliente (representa un concepto y no una especificación detallada), cada historia debe devolver algún valor para el cliente, el tamaño de las historias de usuario debe ser tal que se puedan construir varias de ellas en una iteración, las historias deben ser independientes unas de otras, cada historia

debe poder probarse. El tiempo de estimación es un aproximado, así como la prioridad de cada historia de usuario.

En el análisis para el desarrollo de la solución propuesta, fueron obtenidas ocho historias de usuario detalladas a continuación:

1. Proteger el código fuente del cliente de inventario.
2. Proteger la base de datos del cliente de inventario.
3. Proteger los archivos de configuración del cliente de inventario.
4. Procesar la información de forma asíncrona y distribuida.
5. Establecer el protocolo de comunicación asíncrono.
6. Establecer la seguridad de la comunicación.
7. Gestionar la configuración del cliente de inventario.
8. Generar el instalador del cliente de inventario.

A continuación, se muestra la historia de usuario Gestionar la configuración del cliente de inventario, las restantes se encuentran en el **Anexo 1**.

Historia de Usuario	
Número: 7	**Nombre:** Gestionar la configuración del cliente de inventario.
Usuario: Especialista	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 4
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite gestionar la configuración del cliente de inventario, así como visualizar el inventario realizado, utilizando un Command Line Interface (CLI) en caso de los sistemas sin entorno de escritorio y una Graphical User Interface (GUI) en caso contrario, esto se realiza para mitigar la manipulación de datos.	
Observaciones: Para modificar la configuración el usuario debe tener permisos administrativos.	

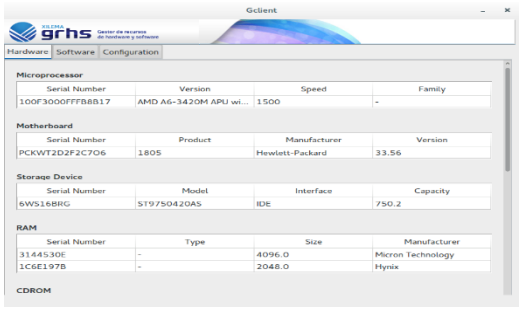
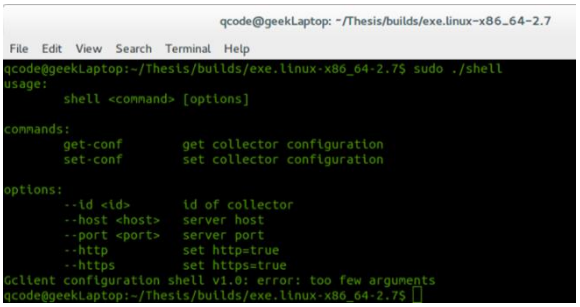
Prototipo de interfaz:
 

Tabla 1. HU-7. Gestionar la configuración del cliente de inventario.

2.2.1.2 Características no funcionales

Disponibilidad:

- El servidor de mensajes Rabbitmq ha de estar operacional durante todo el día.
- El servidor de inventario ha de estar operacional durante todo el día.

Rendimiento:

- La computadora donde se encuentre instalado el servidor de mensajería ha de tener como mínimo 128 Mb de RAM y 2 GB de espacio libre en el disco duro.

Portabilidad:

- El cliente de inventario puede ser instalado en Ubuntu 14.04, Debian 7/8, Nova 4/5 y Windows XP/7/8/10.

2.2.2 Planificación

En esta fase se establece la prioridad de cada historia de usuario, y en correspondencia, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a

una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

2.2.2.1 Prioridad por Historia de Usuario

La asignación de prioridad a las historias de usuario por el cliente es lo que decide el orden en el cual se implementarán las mismas, siempre y cuando estén de acuerdo usuario y desarrollador. Las historias de usuario con menor número de prioridad son las más importantes, quedando conformadas de la siguiente manera:

Historias de Usuario	Prioridad
Proteger el código fuente del cliente de inventario.	1
Proteger la base de datos del cliente de inventario.	2
Proteger los archivos de configuración del cliente de inventario.	2
Procesar la información de forma asíncrona y distribuida.	3
Establecer el protocolo de comunicación asíncrono.	3
Establecer la seguridad de la comunicación.	4
Gestionar la configuración del cliente de inventario.	4
Generar el instalador del cliente de inventario.	4

Tabla 2. Prioridad por Historia de Usuario.

2.2.2.2 Estimación de esfuerzo por Historia de Usuario

La siguiente tabla contiene la estimación de esfuerzo por Historia de Usuario según el orden a realizar. Esta estimación incluye todo el esfuerzo asociado a la implementación

de la HU, la misma expresa utilizando como medida el punto (máximo esfuerzo). Un punto se considera como una semana ideal de trabajo, donde se trabaje el tiempo planeado sin ningún tipo de interrupción.

Historias de Usuario	Puntos de Estimación
Proteger el código fuente del cliente de inventario.	3
Proteger la base de datos del cliente de inventario.	1
Proteger los archivos de configuración del cliente de inventario.	1
Procesar la información de forma asíncrona y distribuida.	1
Establecer el protocolo de comunicación asíncrono.	2
Establecer la seguridad de la comunicación.	1
Gestionar la configuración del cliente de inventario.	1
Generar el instalador del cliente de inventario.	1

Tabla 3. Estimación de esfuerzo por Historia de Usuario.

2.2.2.3 Plan de iteraciones

Luego de identificar las HU y la estimación del esfuerzo por cada una de ellas, se procede a realizar el plan de iteraciones, en el cual estarán contenidas las HU en el orden a realizar por cada iteración según su orden de relevancia, así como el total de semanas que durarán cada una de estas. Teniendo en cuenta el riesgo para desarrollar cada una de las historias de usuario, el tamaño del equipo de desarrollo, así como otros factores subjetivos, se decidió dividir el desarrollo en cuatro iteraciones.

Este plan tiene como objetivo mostrar la duración de cada iteración, así como el orden en que serán implementadas las historias de usuario en cada una de las mismas.

Iteración		
Número: 1	**H.U:** 1	**Duración total:** 3 semanas
Descripción: Se protege el código fuente del cliente de inventario por medio de la compilación y el empaquetamiento del mismo.		

Tabla 4. Plan de iteraciones. Iteración 1.

Iteración		
Número: 2	**H.U:** 2, 3	**Duración total:** 2 semanas
Descripción: Se desarrollan las funcionalidades que permiten proteger los archivos de configuración y la base de datos del cliente de inventario.		

Tabla 5. Plan de iteraciones. Iteración 2.

Iteración		
Número: 3	**H.U:** 4, 5	**Duración total:** 3 semanas
Descripción: Se establece el protocolo de comunicación asíncrono en el sistema GRHS. Hecho que implica la modificación del flujo de comunicación existente entre el cliente y servidor de inventario.		

Tabla 6. Plan de iteraciones. Iteración 3.

Iteración		
Número: 4	**H.U:** 6, 7, 8	**Duración total:** 3 semanas
Descripción: Se configuran los certificados digitales y el protocolo SRP para la autenticación de la información, se desarrollan las interfaces para gestionar la configuración del cliente de inventario y se genera el instalador del cliente de inventario.		

Tabla 7. Plan de iteraciones. Iteración 4.

2.2.2.4 Plan de entregas

El plan de entregas es el cronograma de entregas que establece que historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. En este plan se concentran las funcionalidades referentes a un mismo tema en módulos, esto permite un mayor entendimiento en la fase de implementación. Tiene como objetivo definir el número de liberaciones que se realizarán en el transcurso del proyecto y las iteraciones que se requieren para desarrollar cada una.

Módulos	HU que abarca
Código	Proteger el código fuente del cliente de inventario.
	Proteger la base de datos del cliente de inventario.
	Proteger los archivos de configuración del cliente de inventario.
Comunicación	Procesar la información de forma asíncrona y distribuida.
	Establecer el protocolo de comunicación asíncrono.
Configuración	Establecer la seguridad de la comunicación.
	Gestionar la configuración del cliente de inventario.
	Generar el instalador del cliente de inventario.

Tabla 8. Plan de entregas. HU por módulos.

	Final 1ra iteración	Final 2da iteración	Final 3ra iteración	Final 4ta iteración
Módulos	Final 1ra semana de marzo	Final 3ra semana de marzo	Final 2da semana de abril	Final 5ta semana de abril
Código	0.4	1.0	1.0	1.0
Comunicación			0.6	1.0
Configuración		0.4	0.7	1.0

Tabla 9. Plan de entregas. Porciento de realización.

2.2.3 Diseño

Durante el diseño de la solución, la máxima simplicidad posible es la clave para el éxito de XP. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra deben ser evitados en todo momento. El diseño adecuado para el software es aquel que supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos. Se debe tener en cuenta que un diseño

complejo siempre tarda más en desarrollarse que uno simple, y que siempre es más fácil añadir complejidad a un diseño simple que quitarla de uno complejo.

2.2.3.1 Patrones de arquitectura

Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. Para el desarrollo de la solución se utilizó el patrón de arquitectura Inversión de Control mediante la biblioteca PyDI.

Inversión de Control (IoC) conceptualiza que un servicio dependiente no debe tener código ligado a el de cualquiera de las clases de implementación o métodos para la localización de los servicios de dependencia. Este dispone que todos los servicios de dependencias de un servicio dependiente están propiamente conectados y que su ciclo de vida sea administrado [31].

La biblioteca PyDI mediante la clase ContainerPluginManager rige el comportamiento del cliente de inventario, esta inicializa los servicios, inyecta las dependencias necesarias para el funcionamiento de cada uno, administra los archivos de configuración e inserta las referencias dinámicas de los servicios.

2.2.3.2 Patrones de diseño

Los patrones de diseño son una solución reutilizable de problemas recurrentes que ocurren durante el desarrollo del software [32]. Para el diseño de la solución se hizo uso de los Patrones Generales de Software para Asignar Responsabilidades (GRASP), así como de los patrones Gang of Four o Pandilla de los Cuatro (GoF).

Los patrones GRASP utilizados fueron:

- Alta cohesión: en cada clase solo se implementaron las funcionalidades que le corresponden.
- Bajo acoplamiento: cada clase implementada solo se comunica con un número relativamente pequeño de clases.
- Creador: las clases implementadas que tienen la responsabilidad de crear objetos contienen toda la información necesaria para construir los mismos.

- Experto: cada clase implementada mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas.

Se utilizaron como patrones GoF:

- Decorator (Decorador): se utiliza para crear las tareas de Celery en el servidor de inventario, agregando de forma dinámica las opciones del Celery a los métodos implementados.
- Singleton (Solitario): se utiliza para crear una sola instancia de la clase AMQPManager por medio de la biblioteca PyDI, proporcionando un punto de acceso global a la misma.
- Observer (Observador): se utiliza en el protocolo de comunicación AMQP que implementa el servidor de mensajería, por lo que el sistema GRHS hace uso del mismo para establecer la comunicación, este se conoce también por el nombre publisher-suscriber.

Otros patrones utilizados:

- Dependency Injection (Inyección de dependencias): se utiliza en la biblioteca PyDI para inyectar las dependencias entre servicios en el cliente de inventario, por tanto, el servicio de comunicación hace uso de este.

2.2.3.3 Tarjetas CRC

Estas tarjetas pueden ser también denominadas: tarjetas de clase-responsabilidad-colaboración. Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores.

Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realiza, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

Tarjeta CRC #1	
Clase: AMQPManager	
Responsabilidad	**Colaboración**

▪ Administrar el flujo de comunicación del cliente con el servidor de inventario.	▪ Task ▪ Celery

Tabla 10. Tarjeta CRC para la clase AMQPManager.

Tarjeta CRC #2	
Clase: Task	
Responsabilidad	**Colaboración**
▪ Almacenar el contenido de una tarea. ▪ Administrar el envío de una tarea para su ejecución y la obtención del resultado.	▪ Ninguno

Tabla 11. Tarjeta CRC para la clase Task.

Tarjeta CRC #3	
Clase: GLoader	
Responsabilidad	**Colaboración**
▪ Realizar el cifrado y descifrado de los archivos de configuración.	▪ AES

Tabla 12. Tarjeta CRC para la clase Loader.

Tarjeta CRC #4	
Clase: GskinTrayIcon	
Responsabilidad	**Colaboración**
▪ Crear el acceso directo en la barra de tareas. ▪ Iniciar la vista principal de la GUI.	▪ Main

Tabla 13. Tarjeta CRC para la clase GskinTrayIcon.

Tarjeta CRC #5	
Clase: Main	
Responsabilidad	**Colaboración**

▪ Mostrar los datos del inventario realizado. ▪ Administrar la configuración del cliente de inventario.	▪ Ninguno

Tabla 14. Tarjeta CRC para la clase Main.

Tarjeta CRC #6	
Clase: AuthToken	
Responsabilidad	**Colaboración**
▪ Almacenar la información relacionada con los tokens de autenticación.	▪ Model

Tabla 15. Tarjeta CRC para la clase AuthToken.

2.3 Conclusiones del capítulo

La propuesta de solución presentada se enmarca en la mitigación de las amenazas previamente definidas teniendo en cuenta el desarrollo de soluciones específicas para cada vulnerabilidad detectada posibilitando un mayor acople de la solución al sistema GRHS, reduciendo la cantidad de cambios en el mismo. Dentro de la propuesta de solución se planteó la separación lógica de los componentes del sistema GRHS permitiendo un mayor entendimiento de la misma.

A partir de la propuesta de solución se obtuvieron ocho HU durante la fase de exploración de la metodología XP. En la fase de planificación se determinaron las prioridades de las mismas, siendo la protección del código fuente la de mayor prioridad y la que más esfuerzo requiere para su realización. El desarrollo de la solución se planificó para cuatro iteraciones con una duración aproximada de once semanas.

Durante la fase de diseño se determinó la utilización del patrón decorador el cual se utilizó para la creación de las tareas distribuidas. La realización de las tarjetas CRC permitió conocer las relaciones entre las clases destacando la clase AMQPManager como principal en el proceso de comunicación.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

En este capítulo se presentan los resultados obtenidos en las fases de Implementación y Pruebas correspondientes a la metodología de desarrollo XP. Se detallan las cuatro iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiendo las tareas generadas por cada historia de usuario, así como las pruebas unitarias y de aceptación efectuadas sobre la solución.

3.2 Desarrollo por Iteraciones

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

3.2.1 Tareas de ingeniería

Como se ha mencionado anteriormente las historias de usuario se encuentran compuestas por distintas tareas. Para brindar información acerca de las mismas, se utilizan las denominadas “Tareas de Ingeniería”. Si bien tampoco existe una plantilla específica para las mismas se recomiendan que contengan la siguiente información: el número de tarea, la historia de usuario a la que hacen referencia, el nombre de la tarea, el tipo de tarea (si es de desarrollo, de corrección, de mejora, o algún otro tipo de tarea específico), los puntos estimados, el programador responsable de ella, fecha de inicio, fecha de fin y una breve descripción de la tarea (en que consiste la misma).

3.2.2 Pruebas

Las pruebas son una actividad esencial en la ingeniería de software. En los términos más simples, equivale a la observación de la ejecución de un software para validar si se comporta como se espera e identificar posibles fallos. Estas se utilizan ampliamente en la industria del software para garantizar la calidad: de hecho, mediante el examen directamente del software en ejecución se proporciona una retroalimentación realista de su comportamiento [34].

Durante el desarrollo de cada una de las iteraciones del presente proyecto se realizaron pruebas de caja blanca (pruebas unitarias) y de caja negra (pruebas de aceptación). Las pruebas unitarias realizadas al cliente de inventario se ejecutaron de forma automática utilizando la herramienta PyUnit.

3.3 Iteración 1

Durante esta iteración se abordan las historias de usuario de mayor prioridad, dándosele un primer nivel de seguridad al sistema protegiendo el código fuente mediante la compilación y el empaquetado del mismo.

3.3.1 Tareas de Ingeniería

Historias de usuario	Tareas de ingeniería
Proteger el código fuente del cliente de inventario.	Compilar el código fuente del cliente de inventario.
	Empaquetar el código fuente del cliente de inventario.

Tabla 16. Tareas de ingeniería para la iteración 1.

A continuación, se muestran las tareas de ingeniería correspondientes con la HU-Proteger el código fuente del cliente de inventario:

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. y Nombre):** 1- Proteger el código fuente del cliente de inventario.

Nombre de la tarea: Compilar el código fuente del cliente de inventario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha de inicio: - 14/02	**Fecha de fin:** - 24/02
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite compilar el código fuente escrito en Python utilizando la herramienta Cython la cual usa el lenguaje C como intermediario.	

Tabla 17. HU-1. Tarea de ingeniería 1. Compilar el código fuente del cliente de inventario.

Número: 2	**Historia de usuario (No. y Nombre):** 1- Proteger el código fuente del cliente de inventario.
Nombre de la tarea: Empaquetar el código fuente del cliente de inventario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.5
Fecha de inicio: - 25/02	**Fecha de fin:** - 06/03
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite empaquetar el código fuente del cliente de inventario utilizando la herramienta cx_Freeze.	

Tabla 18. HU-1. Tarea de ingeniería 2. Empaquetar el código fuente del cliente de inventario.

3.3.2 Pruebas

3.3.2.1 Pruebas unitarias

En la presente iteración se desarrollaron 4 pruebas unitarias arrojando los problemas que a continuación se muestran, estos fueron solucionados posteriormente.

- Mezcla de espacios y tabulaciones durante la compilación.
- Variables no inicializadas durante la compilación.
- Referencia inválida a dependencias durante la compilación.
- No se encontró el fichero varsall.bat durante la compilación.
- No se encontró la biblioteca “libxml.so” durante el empaquetamiento.
- Incompatibilidad de versiones de bibliotecas (x86-x64) durante el empaquetamiento.

```
12          def testPackaging(self):
13              status = os.system('python setup.py build')
14              self.assertEqual(status, 0)
15
16          def testCompilation(self):
17              status = os.system('python compiler.py')
18              self.assertEqual(status, 0)
```

Ilustración 5. Iteración 1. Muestra de los casos de pruebas unitarias aplicados.

```
qcode@geekLaptop:~/grhs/gclient$ python tests.py
....
----------------------------------------------------------------------
Ran 4 tests in 347.628s

OK
qcode@geekLaptop:~/grhs/gclient$
```

Ilustración 6. Iteración 1. Resultado de los casos de pruebas unitarias aplicados.

3.3.2.2 Pruebas de aceptación

Las pruebas de aceptación realizadas resultaron satisfactorias. A continuación, se describen los casos de prueba utilizados.

Caso de Prueba de Aceptación
Nombre: Compilar el código fuente del cliente de inventario.
Historia de Usuario: Proteger el código fuente del cliente de inventario.
Descripción: Se ejecuta el script para la compilación del código fuente del cliente de inventario, comprobando que este se encuentre compilado e ilegible al terminar.
Precondiciones: -
Pasos de ejecución: • Situarse desde la terminal en el directorio a compilar. • Ejecutar el script de compilación.

Tabla 19. Caso de Prueba de Aceptación. Compilar el código fuente del cliente de inventario.

Caso de Prueba de Aceptación
Nombre: Empaquetar el código fuente del cliente de inventario.

Historia de Usuario: Proteger el código fuente del cliente de inventario.
Descripción: Se ejecuta el script para el empaquetamiento del código fuente del cliente de inventario, comprobando la portabilidad del mismo al terminar.
Precondiciones: -
Pasos de ejecución: • Situarse desde la terminal en el directorio a compilar. • Ejecutar el script de empaquetamiento.

Tabla 20. Caso de Prueba de Aceptación. Empaquetar el código fuente del cliente de inventario.

3.4 Iteración 2

Durante esta iteración se desarrollan las funcionalidades para cifrar y descifrar la base de datos y los archivos de configuración del cliente de inventario.

3.4.1 Tareas de Ingeniería

Historias de usuario	Tareas de ingeniería
Proteger la base de datos del cliente de inventario.	Cifrar y descifrar la base de datos del cliente de inventario.
Proteger los archivos de configuración del cliente de inventario.	Cifrar y descifrar los archivos de configuración del cliente de inventario.

Tabla 21. Tareas de ingeniería para la iteración 2.

A continuación, se muestra la tarea de ingeniería correspondiente con la HU-Proteger la base de datos del cliente de inventario:

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. y Nombre):** 2- Proteger la base de datos del cliente de inventario.
Nombre de la tarea: Cifrar y descifrar la base de datos del cliente de inventario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha de inicio: - 06/03	**Fecha de fin:** - 13/03
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite realizar el cifrado y descifrado de la base de datos SQLite del cliente de inventario utilizando la biblioteca SQLCipher por intermedio de la dependencia Pysqlcipher.	

Tabla 22HU-2. Tarea de ingeniería 1. Cifrar y descifrar la base de datos del cliente de inventario.

A continuación, se muestra la tarea de ingeniería correspondiente con la HU-Proteger los archivos de configuración del cliente de inventario:

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. y Nombre):** 3- Proteger los archivos de configuración del cliente de inventario.
Nombre de la tarea: Cifrar y descifrar los archivos de configuración del cliente de inventario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: - 13/03	**Fecha de fin:** - 20/03
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite realizar el cifrado y descifrado de los archivos de configuración del cliente de inventario utilizando el algoritmo simétrico AES, en modo CBC con 256 bits de clave, por medio de la biblioteca Pycrypto.	

Tabla 23. HU-2. Tarea de ingeniería 1. Cifrar y descifrar los archivos de configuración del cliente de inventario.

3.4.2 Pruebas

3.4.2.1 Pruebas unitarias

En la presente iteración se desarrollaron 6 pruebas unitarias arrojando los problemas que a continuación se muestran, estos fueron solucionados posteriormente.

- La biblioteca PyDI no reconoce los archivos de configuración.
- La modificación de los archivos de configuración transforma la estructura del formato YAML.

- La versión de la biblioteca Peewee es incompatible con las consultas realizadas en el código del cliente de inventario.

```
20          def testEncryptedDatabase(self):
21              database = 'database.db'
22              os.remove(database)
23              db_ciphered = sqlcipher.connect(database)
24              db_ciphered.executescript('pragma key="secret"; pragma kdf_iter=64000;')
25              db_ciphered.execute('create table gclient (name text primary key);')
26              db_plain = sqlite3.connect(database)
27              self.assertRaises(Exception, db_plain.execute, 'select * from gclient;')
28
29          def testEncryptedFile(self):
30              in_text = 'Top Secret'
31              name = 'file.enc'
32              gloader.save(data=in_text, out_filename=name, yml=False, ciphered=True)
33              out_text = open(name, 'rb').read()
34              self.assertNotEquals(in_text, out_text)
35              out_text = gloader.load(in_filename=name, yml=False, ciphered=True)
36              self.assertEquals(in_text, out_text)
```

Ilustración 7. Iteración 2. Muestra de los casos de pruebas unitarias aplicados.

```
qcode@geekLaptop:~/grhs/gclient$ python tests.py
......
----------------------------------------------------------------------
Ran 6 tests in 1.424s

OK
qcode@geekLaptop:~/grhs/gclient$
```

Ilustración 8. Iteración 2. Resultado de los casos de pruebas unitarias aplicados.

3.4.2.2 Pruebas de aceptación

Las pruebas de aceptación realizadas resultaron satisfactorias. A continuación, se describen los casos de prueba utilizados.

Caso de Prueba de Aceptación
Nombre: Cifrar la base de datos del cliente de inventario.
Historia de Usuario: Proteger la base de datos del cliente de inventario.
Descripción: Se realiza el cifrado de la base de datos SQLite del cliente de inventario y se verifica que la información no pueda ser consultada de forma estándar.
Precondiciones: -
Pasos de ejecución:

• Iniciar el cliente de inventario.

Tabla 24. Caso de Prueba de Aceptación. Cifrar la base de datos del cliente de inventario.

Caso de Prueba de Aceptación
Nombre: Cifrar los archivos de configuración del cliente de inventario.
Historia de Usuario: Proteger los archivos de configuración del cliente de inventario.
Descripción: Se ejecuta el script para el cifrado de los archivos de configuración del cliente de inventario, comprobando que se encuentren ilegibles al terminar.
Precondiciones: -
Pasos de ejecución: • Situarse desde la terminal en el directorio con los archivos de configuración a cifrar. • Ejecutar el script para cifrar los archivos de configuración.

Tabla 25. Caso de Prueba de Aceptación. Cifrar los archivos de configuración del cliente de inventario.

3.5 Iteración 3

Durante esta iteración se establece el procesamiento distribuido en el servidor de inventario y el protocolo de comunicación AMQP para el envío de datos en el sistema GRHS.

3.5.1 Tareas de Ingeniería

Historias de usuario	Tareas de ingeniería
Procesar los datos de forma asíncrona y distribuida.	Enviar y recibir datos de forma asíncrona
	Procesar los datos forma distribuida
Establecer el protocolo de comunicación asíncrono.	Obtener nomencladores y configuraciones
	Obtener el inventario
	Enviar traza de inventario e incidencia
	Obtener actualización.

Tabla 26. Tareas de ingeniería para la iteración 3.

A continuación, se muestran las tareas de ingeniería correspondientes con la HU-Procesar los datos de forma asíncrona y distribuida**:**

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. y Nombre):** 4- Procesar los datos de forma asíncrona y distribuida
Nombre de la tarea: Enviar y recibir datos de forma asíncrona	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 20/03	**Fecha de fin:** - 23/03
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite enviar y recibir datos entre el cliente y el servidor de inventario de forma asíncrona utilizando el protocolo AMQP por medio de la biblioteca Celery.	

Tabla 27. HU-4. Tarea de ingeniería 1. Enviar y recibir datos de forma asíncrona.

Tarea de Ingeniería	
Número: 2	**Historia de usuario (No. y Nombre):** 4- Procesar los datos de forma asíncrona y distribuida
Nombre de la tarea: Procesar los datos forma distribuida.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 23/03	**Fecha de fin:** - 27/03
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite procesar los datos enviados por los clientes de inventario de manera distribuida en el servidor utilizando la herramienta Celery.	

Tabla 28. HU-4. Tarea de ingeniería 2. Procesar los datos forma distribuida.

A continuación, se muestran las tareas de ingeniería correspondientes con la HU-Establecer el protocolo de comunicación asíncrono:

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. y Nombre):** 5- Establecer el protocolo de comunicación asíncrono

Nombre de la tarea: Obtener nomencladores y configuraciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 27/03	**Fecha de fin:** - 30/03
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite obtener del servidor de inventario los nomencladores y las configuraciones del cliente.	

Tabla 29. HU-5. Tarea de ingeniería 1. Obtener nomencladores y configuraciones.

Tarea de Ingeniería	
Número: 2	**Historia de usuario (No. y Nombre):** 5- Establecer el protocolo de comunicación asíncrono.
Nombre de la tarea: Obtener el inventario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 30/03	**Fecha de fin:** - 03/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite obtener el inventario del cliente en el servidor en caso de no existir este se crea.	

Tabla 30. HU-5. Tarea de ingeniería 2. Obtener el inventario.

Tarea de Ingeniería	
Número: 3	**Historia de usuario (No. y Nombre):** 5- Establecer el protocolo de comunicación asíncrono.
Nombre de la tarea: Enviar traza de inventario e incidencia	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 03/04	**Fecha de fin:** - 06/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite enviar las trazas del inventario e incidencias hacia el servidor.	

Tabla 31. HU-5. Tarea de ingeniería 3. Enviar traza de inventario e incidencia.

Tarea de Ingeniería	
Número: 4	**Historia de usuario (No. y Nombre):** 5- Establecer el protocolo de comunicación asíncrono.
Nombre de la tarea: Obtener actualización	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 06/04	**Fecha de fin:** - 10/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite obtener del servidor en caso de existir la actualización del cliente.	

Tabla 32. HU-5. Tarea de ingeniería 4. Obtener actualización.

3.5.2 Pruebas

3.5.2.1 Pruebas unitarias

En la presente iteración se desarrollaron 12 pruebas unitarias arrojando los problemas que a continuación se muestran, estos fueron solucionados posteriormente.

- Los procesos del monitoreo no se comunican con el servidor.
- Las incidencias no muestran la localización del agente.
- La dirección IP del agente es incorrecta.
- La instancia de Celery no se comunica vía SSL con el servidor Rabbitmq.
- La base de datos del cliente de inventario se bloquea.
- Error en el tipo de codificación al serializar la actualización del cliente de inventario para su descarga.

```
38          def testGetConfiguration(self):
39              app = Celery('amqp://guest:guest@127.0.0.1:5672//')
40              task = app.send_task('configuration.actionlevel')
41              result = task.get()
42              self.assertNotEqual(result, None)
43
44          def testGetNomenclator(self):
45              app = Celery('amqp://guest:guest@127.0.0.1:5672//')
46              task = app.send_task('nomenclator.inventorytype')
47              result = task.get()
48              self.assertNotEqual(result, None)
```

Ilustración 9. Iteración 3. Muestra de los casos de pruebas unitarias aplicados.

```
qcode@geekLaptop:~/grhs/gclient$ python tests.py
............
----------------------------------------------------------------------
Ran 12 tests in 4.849s

OK
qcode@geekLaptop:~/grhs/gclient$
```

Ilustración 10. Iteración 3. Resultado de los casos de pruebas unitarias aplicados.

3.5.2.2 Pruebas de aceptación

Las pruebas de aceptación realizadas resultaron satisfactorias. A continuación, se describen los casos de prueba utilizados.

Caso de Prueba de Aceptación
Nombre: Procesar la información de forma distribuida.
Historia de Usuario: Procesar la información de forma asíncrona y distribuida.
Descripción: Se ejecutan instancias del servidor de inventario verificándose que trabajaban de forma distribuida.
Precondiciones: Tener activo del servidor de mensajería Rabbitmq
Pasos de ejecución: • Ejecutar varias instancias del gserver. • Ejecutar varios clientes de inventario.

Tabla 33. Caso de Prueba de Aceptación. Procesar la información de forma distribuida.

Caso de Prueba de Aceptación
Nombre: Comunicación asíncrona.
Historia de Usuario: Establecer el protocolo de comunicación asíncrono.
Descripción: Se ejecuta un cliente de inventario, se verifica que envíe los datos y reciba la respuesta de forma asíncrona.
Precondiciones: Tener activo del servidor de mensajería Rabbitmq.
Pasos de ejecución: • Iniciar el servidor de inventario.

• Iniciar el cliente de inventario.

Tabla 34. Caso de Prueba de Aceptación. Comunicación asíncrona.

3.6 Iteración 4

En esta iteración se configura la seguridad en la comunicación entre el cliente y el servidor de inventario, se gestiona la configuración del cliente de inventario y se genera el instalador del mismo. Al culminar esta iteración, se consta con la solución lista para su puesta en funcionamiento.

3.6.1 Tareas de Ingeniería

Historias de usuario	Tareas de ingeniería
Establecer la seguridad de la comunicación.	Establecer el método de autenticación para la comunicación.
	Configurar el uso de los certificados digitales.
Gestionar la configuración del cliente de inventario.	Autenticar usuario administrador.
	Modificar la configuración del cliente de inventario mediante una CLI (Command Line Interface).
	Modificar la configuración mediante del cliente de inventario mediante una GUI (Graphical User Interface)
Generar el instalador del cliente de inventario.	Generar los prototipos del instalador del cliente de inventario
	Automatizar el proceso de generación del instalador del cliente de inventario.

Tabla 35. Tareas de ingeniería para la iteración 4.

A continuación, se muestran las tareas de ingeniería correspondientes con la HU-Establecer la seguridad de la comunicación:

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. y Nombre):** 6- Establecer la

	seguridad de la comunicación
Nombre de la tarea: Establecer el método de autenticación para la comunicación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 10/04	**Fecha de fin:** - 13/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite establecer un método de autenticación para verificar la autenticidad de los clientes de inventario. Este se implementa utilizando el protocolo SRP.	

Tabla 36. HU-6. Tarea de ingeniería 1. Establecer el método de autenticación para la comunicación.

Tarea de Ingeniería	
Número: 2	**Historia de usuario (No. y Nombre):** 6- Establecer la seguridad de la comunicación
Nombre de la tarea: Configurar el uso de los certificados digitales.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 13/04	**Fecha de fin:** - 17/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite establecer una conexión segura entre el sistema GRHS y el servidor de mensajería Rabbitmq. Los certificados digitales utilizados se cargarán dinámicamente en el cliente de inventario.	

Tabla 37. HU-6. Tarea de ingeniería 2. Configurar el uso de los certificados digitales.

A continuación, se muestran las tareas de ingeniería correspondientes con la HU-Gestionar la configuración del cliente de inventario:

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. Y Nombre):** 7- Gestionar la configuración del cliente de inventario.
Nombre de la tarea: Autenticar usuario administrador.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: -17/04	**Fecha de fin:** -19/04

Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez
Descripción: Permite limitar el acceso a la gestión de la configuración solo a los usuarios administradores.

Tabla 38. HU-7. Tarea de ingeniería 1. Autenticar usuario administrador.

Tarea de Ingeniería	
Número: 2	**Historia de usuario (No. Y Nombre):** 7- Gestionar la configuración del cliente de inventario.
Nombre de la tarea: Modificar la configuración del cliente de inventario mediante una CLI.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: -19/04	**Fecha de fin:** -21/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite cambiar los parámetros de configuración del cliente de inventario mediante una CLI implementada en Python.	

Tabla 39. HU-7. Tarea de ingeniería 2. Modificar la configuración del cliente de inventario mediante una CLI.

Tarea de Ingeniería	
Número: 3	**Historia de usuario (No. Y Nombre):** 7- Gestionar la configuración del cliente de inventario.
Nombre de la tarea: Modificar la configuración del cliente de inventario mediante una GUI.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: -21/04	**Fecha de fin:** -24/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite cambiar los parámetros de configuración del cliente de inventario y visualizar el inventario realizado mediante una GUI implementada mediante la biblioteca PyQt.	

Tabla 40. HU-7. Tarea de ingeniería 3. Modificar la configuración del cliente de inventario mediante una GUI.

A continuación, se muestran las tareas de ingeniería correspondientes con la HU-Generar el instalador del cliente de inventario:

Tarea de Ingeniería	
Número: 1	**Historia de usuario (No. y Nombre):** 8- Generar el instalador del cliente de inventario.
Nombre de la tarea: Generar los prototipos del instalador del cliente de inventario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 24/04	**Fecha de fin:** - 27/04
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite generar los prototipos del instalador del cliente de inventario para varios sistemas operativos utilizando la herramienta InstallJammer.	

Tabla 41. HU-8. Tarea de ingeniería 1. Generar los prototipos del instalador del cliente de inventario.

Tarea de Ingeniería	
Número: 2	**Historia de usuario (No. y Nombre):** 8- Generar el instalador del cliente de inventario.
Nombre de la tarea: Automatizar el proceso de generación del instalador del cliente de inventario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: - 27/04	**Fecha de fin:** - 01/05
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite automatizar los procesos de compilación, empaquetado y generación de instaladores del cliente de inventario utilizando la herramienta Jenkins.	

Tabla 42. HU-8. Tarea de ingeniería 2. Automatizar el proceso de generación del instalador del cliente de inventario.

3.6.2 Pruebas

3.6.2.1 Pruebas unitarias

En la presente iteración se desarrollaron 14 pruebas unitarias arrojando los problemas que a continuación se muestran, estos fueron solucionados posteriormente.

- La longitud del token de autenticación excede el tamaño máximo asignado.

- La biblioteca PyDI no encuentra el plugin “logger”.
- La interfaz gráfica para visualizar el inventario no inicia cuando no existe la base de datos del cliente de inventario.
- El cliente de inventario no inicia como servicio cuando se instala.
- El cliente de inventario no crea el archivo de logs ni la base de datos en el lugar donde se encuentra instalado.

```
50          def testGetNomenclatorSSL(self):
51              app = Celery('amqp://guest:guest@127.0.0.1:5671//')
52              app.conf.update({'BROKER_USE_SSL': {
53                  "keyfile":  "ssl/key.pem",
54                  "certfile": "ssl/cert.pem",
55                  "ca_certs": "ssl/cacert.pem"
56              }})
57              task = app.send_task('nomenclator.inventorytype')
58              result = task.get()
59              self.assertNotEqual(result, None)
60
61          def testHandshake(self):
62              app = Celery('amqp://guest:guest@127.0.0.1:5671//')
63              app.conf.update({'BROKER_USE_SSL': {
64                  "keyfile":  "ssl/key.pem",
65                  "certfile": "ssl/cert.pem",
66                  "ca_certs": "ssl/cacert.pem"
67              }})
68              result = app.send_task('handshake', {"step": 0}).get()
69              self.assertNotEqual(result, None)
```

Ilustración 11. Iteración 4. Muestra de los casos de pruebas unitarias aplicados.

```
qcode@geekLaptop:~/grhs/gclient$ python tests.py
..............
----------------------------------------------------------------------
Ran 14 tests in 49.284s

OK
qcode@geekLaptop:~/grhs/gclient$
```

Ilustración 12. Iteración 4. Resultado de los casos de pruebas unitarias aplicados.

3.6.2.2 Pruebas de aceptación

Las pruebas de aceptación realizadas resultaron satisfactorias. A continuación, se describen los casos de prueba utilizados.

Caso de Prueba de Aceptación
Nombre: Autenticar cliente de inventario.
Historia de Usuario: Establecer la seguridad de la comunicación.
Descripción: Se inicia el cliente de inventario y se verifica que el token obtenido durante la ejecución del protocolo SRP sea válido.
Precondiciones: Tener activo del servidor de mensajería Rabbitmq.
Pasos de ejecución: • Iniciar el servidor de inventario. • Iniciar el cliente de inventario.

Tabla 43. Caso de Prueba de Aceptación. Autenticar cliente de inventario.

Caso de Prueba de Aceptación
Nombre: Editar archivo de configuración.
Historia de Usuario: Gestionar la configuración del cliente de inventario.
Descripción: Se editan los archivos de configuración usando la interfaz gráfica y se comprueba que su edición haya sido correcta.
Precondiciones:-.
Pasos de ejecución: • Iniciar la interfaz gráfica.

Tabla 44. Caso de Prueba de Aceptación. Editar archivo de configuración.

Caso de Prueba de Aceptación
Nombre: Construir el instalador utilizando la herramienta Jenkins.
Historia de Usuario: Generar el instalador del cliente de inventario.
Descripción: Se ejecuta el proyecto en el Jenkins verificando que se genera el instalador con el código fuente compilado, empaquetado y los archivos de configuración cifrados.
Precondiciones:-.
Pasos de ejecución:

• Ejecutar el proyecto en el Jenkins.

Tabla 45. Caso de Prueba de Aceptación. Construir el instalador utilizando la herramienta Jenkins.

3.7 Conclusiones del capítulo

En este capítulo se llevaron a cabo las fases de implementación y pruebas planteadas por la metodología XP. Para guiar el proceso de implementación se desarrollaron 17 tareas de ingeniería, distribuidas en 4 iteraciones, las cuales propiciaron una mayor organización y rapidez en el desarrollo del sistema. Con el objetivo de contribuir a elevar la calidad final del producto se realizaron 36 pruebas unitarias y 9 pruebas de aceptación, las cuales arrojaron 21 errores en total. Estos fueron corregidos posteriormente, validando cada una de las funcionalidades para evitar que el software con defectos llegue al cliente.

CONCLUSIONES

En el presente trabajo fueron cumplidos los objetivos planteados, así como las tareas de la investigación arribando a las siguientes conclusiones:

- Se realizó un análisis de los fundamentos teóricos referentes a la seguridad informática propiciando una base sólida para el desarrollo de la solución.
- Se realizó una evaluación de las diferentes técnicas de protección de código fuente, escrito en Python determinándose el uso de la compilación y el empaquetamiento para el desarrollo de la solución.
- Se realizó un análisis de las técnicas para proteger los archivos de configuración y la base de datos del cliente de inventario, para esto se realizó el cifrado de los mismos.
- Se identificó el protocolo de comunicación asíncrono AMQP como el más apropiado para el desarrollo de la solución.
- Se incluyó el procesamiento distribuido en el sistema GRHS propiciando un mayor balance de carga.
- Se realizó un análisis de las técnicas para verificar la autenticidad de las aplicaciones informáticas utilizándose en la solución el protocolo Secure Remote Password (SRP) y los certificados digitales.
- Se seleccionó la metodología, herramientas y tecnologías que mejor se ajustaban con el desarrollo de la solución y el equipo de trabajo.
- Se validó la solución implementada, arrojando resultados satisfactorios, dado que las deficiencias encontradas fueron solucionadas en iteraciones posteriores.

El desarrollo de esta solución constituye un aporte importante al sistema GRHS, debido a que se mitigan amenazas de seguridad que posee el sistema, brindando un producto más seguro y de mayor calidad.

RECOMENDACIONES

Se propone continuar el desarrollo de la solución con funcionalidades tales como:

- Detectar y notificar intentos de conexión de clientes de inventario inválidos.
- Gestionar los certificados digitales en el cliente de inventario.
- Realizar la autenticación de los usuarios en las interfaces basándose en los permisos del servidor de inventario.
- Reiniciar el servicio del cliente de inventario desde las interfaces.

REFERENCIAS BIBLIOGRÁFICAS

[1]. **SANS Institute.** SANS. [En línea] [Citado: Enero 26, 2016.] https://www.sans.org/critical-security-controls.

[2]. **Hernández Pérez, Julio Antonio, Ordoñes Leyva, Yoanni y Aviles Vazquez, Ernesto.** *GESTIÓN DE INCIDENCIAS EN INVENTARIOS DE RED.* La Habana, Cuba : Universidad de las Ciencias Informáticas, 2015.

[3]. **MIC.** *Resolución 127/2007* (1) *MIC. Reglamento de seguridad para las tecnologías de la información.* La Habana, Cuba : Ministerio de la informática y las comunicaciones (MIC), 2007.

[4]. **ISO/IEC.** *ISO/IEC 27000: Information security management systems - Overview and vocabulary.* International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), 2014.

[5]. **Microsoft Corporation.** The STRIDE Threat Model. *Microsoft Developer Network.* [En línea] 2005. https://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx.

[6]. **Jung, EJ.** University of San Francisco, Departament of Computer Science. *CS 686 Special Topics in CS Privacy and Security.* [En línea] 2010. http://www.cs.usfca.edu/~ejung/courses/686/lectures/02def.pdf.

[7]. **Arini Balakrishnan, Chloe Schulze.** *Code Obfuscation Literature Survey.* Computer Sciences Department : University of Wisconsin, Madison, 2005. CS701.

[8]. **Dic. ALEGSA** - Santa Fe, Argentina. [En línea] http://www.alegsa.com.ar/Dic/.

[9]. **Smith, Rich.** In memory reverse engineering for obfuscated Python bytecode. Las Vegas: Immunity Inc, 2010.

[10]. **O'Leary, Mike**. Malware and Persistence. *Cyber Operations.* s.l. : Springer, 2015, págs. 367-410.

[11]. **Herman, Justin.** pyobfuscate. Python source code obfuscator. [En línea] 29 de noviembre de 2013. https://github.com/astrand/pyobfuscate.

[12]. **Femerling, Simon Roses.** AppSec. Myths about Obfuscation and Reversing Python. [En línea] http://www.simonroses.com/2013/10/appsec-myths-about-obfuscation-and-reversing-python/.

[13]. *Using cython to speed up numerical python programs.* **Wilbers, Ilmar M, Langtangen, Hans Petter y Odegard, Asmund.** 2009, Proceedings of MekIT, Vol. 9, págs. 495-512.

[14]. *Performance of Python runtimes on a non-numeric scientific code.* **Murri, Riccardo** [15]. **SQLite** . The SQLite Encryption Extension (SEE). [En línea] http://www.hwaci.com/- sw/sqlite/ see.html.

[16]. *Stronger Password-Based Encryption Using All-or-Nothing Transforms.* **Zaverucha, Greg.** s.l. : Microsoft Research, 2015.

[17]. **Sherif, Asser.** *School of Sciences and Engineering.* The American University in Cairo. 2014. Ph.D. dissertation.

[18]. *A Working Introduction to Crypto with PyCrypto.* **Isom, Kyle.** 2011.

[19]. **SQLite** . The SQLite Encryption Extension (SEE). [En línea] http://www.hwaci.com/- sw/sqlite/ see.html.

[20]. **Zaccone, Giancarlo.** *Python Parallel Programming Cookbook.* s.l. : Packt Publishing Ltd, 2015.

[21]. *Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand.* **Marsh, Gregory, y otros.** 2008, Ohio State University, Tech. Rep. OSU-CISRC-5/09-TR17.

[22]. **Videla, Alvaro and Williams, Jason JW.** *RabbitMQ in action.* s.l. : Manning, 2012.

[23]. **Málaga**. Certificados Digitales. [En línea] [Citado el: 7 de abril de 2016.] http://www.malaga.eu/recursos/ayto/m_gestiones/firma/debesaber/3_1.html.

[24] *The Secure Remote Password Protocol.* **Wu, Thomas D y others.** 1998. NDSS. Vol. 98, págs. 97-111.

[25] **Beck, Kent.** *Extreme Programming Explained Embrance Change.* 1999. ISBN: 0201616416.

[26]. **VIM**. Vim the editor. [En línea] http://www.vim.org/about.php.

[27] **Foundation, Django Software**. Django. *The web framework for perfectionists with deadlines.* [En línea] https://www.djangoproject.com/.

[28] **Worsley, John y Drake, Joshua D.** *Practical PostgreSQL.* s.l. : " O'Reilly Media, Inc.", 2002.

[29] **Courtney, Damon.** InstallJammer *Multiplatform Installer.* [En línea] [Citado el: 4 de febrero de 2016.] http://installjammer.com/docs/.

[30] **Kawaguchi, Kohsuke.** Jenkins. [En línea] [Citado el: 6 de febrero de 2016.] https://jenkins.io/doc/.

[31] **Huang, Yunwu y Ponzo, John J.** *Dependency injection by static code generation.* s.l.: Google Patents, #Jun.#~3 de 2014. US Patent 8,745,584.

[32] *Design patterns: Elements of reusable object-oriented software.* **Vlissides, John, et al.** 120, 1995, Reading: Addison-Wesley, Vol. 49, p. 11.

[33] **Unidad Docente de Ingeniería del Software.** Patrones del “Gang of Four”. *Facultad de Informática - Universidad Politécnica de Madrid.* 2013desi

[34] *Software testing research: Achievements, challenges, dreams.* **Bertolino, Antonia.** 2007. 2007 Future of Software Engineering. págs. 85-103.

BIBLIOGRAFÍA

1. *Embracing change with extreme programming.* **Beck, Kent.** 10, s.l. : IEEE, 1999, Computer, Vol. 32, págs. 70-77.

2. **Beck, Kent.** *Extreme programming explained: embrace change.* s.l. : addison-wesley professional, 2000.

3. *Cython: The best of both worlds.* **Behnel, Stefan, y otros.** 2, s.l. : AIP Publishing, 2011, Computing in Science \& Engineering, Vol. 13, págs. 31-39.

4. **Harwani, BM.** *Introduction to Python programming and developing GUI applications with PyQT.* s.l. : Nelson Education, 2011.

5. **Videla, Alvaro y Williams, Jason JW.** *RabbitMQ in action.* s.l. : Manning, 2012.

6. *Rapid GUI Programming with Python and Qt. Definitive Guide to PyQt.* **Summerfield, Mark.** s.l. : Prentice, 2007.

7. *Introducción a la Arquitectura de Software.* **Reynoso, Carlos Billy.** 2004, Universidad de Buenos Aires, Vol. 33.

8. **Pressman, Roger S.** *Software engineering: a practitioner's approach.* s.l. : Palgrave Macmillan, 2005.

9. **Pressman, Roger S y Troya, Jose Maria.** *Ingeniería del software.* s.l. : McGraw Hill, 1988.

10. *Auditoria informática.* **Piattini, Mario y Del Peso, Emilio.** 2001, Un enfoque práctico.

11. **McClure, Stuart, Scambray, Joel y Kurtz, George.** *Hackers 3: secretos y soluciones para la seguridad de redes.* 2002.

12. *Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand.* **Marsh, Gregory, y otros.** 2008, Ohio State University, Tech. Rep. OSU-CISRC-5/09-TR17.

13. *Securing web application code by static analysis and runtime protection.* **Huang, Yao-Wen, y otros.** 2004. Proceedings of the 13th international conference on World Wide Web. págs. 40-52.

14. **SANS Institute.** SANS. [En línea] [Citado: Enero 26, 2016.] https://www.sans.org/critical-security-controls.

15. **Hernández Pérez, Julio Antonio, Ordoñes Leyva, Yoanni y Aviles Vazquez, Ernesto.** *GESTIÓN DE INCIDENCIAS EN INVENTARIOS DE RED.* La Habana, Cuba : Universidad de las Ciencias Informáticas, 2015.

16. **MIC.** *Resolución 127/2007* (1) *MIC. Reglamento de seguridad para las tecnologías de la información.* La Habana, Cuba : Ministerio de la informática y las comunicaciones (MIC), 2007.

17. **ISO/IEC.** *ISO/IEC 27000: Information security management systems - Overview and vocabulary.* International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), 2014.

18. **Microsoft Corporation.** The STRIDE Threat Model. *Microsoft Developer Network.* [En línea] 2005. https://msdn.microsoft.com/en-US/library/ee823878(v=cs.20).aspx.

19. **Jung, EJ.** University of San Francisco, Departament of Computer Science. *CS 686 Special Topics in CS Privacy and Security.* [En línea] 2010. http://www.cs.usfca.edu/~ejung/courses/686/lectures/02def.pdf.

20. **Arini Balakrishnan, Chloe Schulze.** *Code Obfuscation Literature Survey.* Computer Sciences Department : University of Wisconsin, Madison, 2005. CS701.

21. **Dic. ALEGSA** - Santa Fe, Argentina. [En línea] http://www.alegsa.com.ar/Dic/.

22. **Smith, Rich.** In memory reverse engineering for obfuscated Python bytecode. Las Vegas: Immunity Inc, 2010.

24. **O'Leary, Mike**. Malware and Persistence. *Cyber Operations.* s.l. : Springer, 2015, págs. 367-410.

25. **Herman, Justin.** pyobfuscate. Python source code obfuscator. [En línea] 29 de noviembre de 2013. https://github.com/astrand/pyobfuscate.

26. **Femerling, Simon Roses.** AppSec. Myths about Obfuscation and Reversing Python. [En línea] http://www.simonroses.com/2013/10/appsec-myths-about-obfuscation-and-reversing-python/.

27. *Using cython to speed up numerical python programs.* **Wilbers, Ilmar M, Langtangen, Hans Petter y Odegard, Asmund.** 2009, Proceedings of MekIT, Vol. 9, págs. 495-512.

28. *Performance of Python runtimes on a non-numeric scientific code.* **Murri, Riccardo**

29. **SQLite** . The SQLite Encryption Extension (SEE). [En línea] http://www.hwaci.com/-sw/sqlite/ see.html.

30. *Stronger Password-Based Encryption Using All-or-Nothing Transforms.* **Zaverucha, Greg.** s.l. : Microsoft Research, 2015.

31. **Sherif, Asser.** *School of Sciences and Engineering.* The American University in Cairo. 2014. Ph.D. dissertation.

32. *A Working Introduction to Crypto with PyCrypto.* **Isom, Kyle.** 2011.

33. **SQLite** . The SQLite Encryption Extension (SEE). [En línea] http://www.hwaci.com/-sw/sqlite/ see.html.

34. **Zaccone, Giancarlo.** *Python Parallel Programming Cookbook.* s.l. : Packt Publishing Ltd, 2015.

35. *Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand.* **Marsh, Gregory, y otros.** 2008, Ohio State University, Tech. Rep. OSU-CISRC-5/09-TR17.

36. **Videla, Alvaro and Williams, Jason JW.** *RabbitMQ in action.* s.l. : Manning, 2012.

37. **Málaga**. Certificados Digitales. [En línea] [Citado el: 7 de abril de 2016.] http://www.malaga.eu/recursos/ayto/m_gestiones/firma/debesaber/3_1.html.

38. *The Secure Remote Password Protocol.* **Wu, Thomas D y others.** 1998. NDSS. Vol. 98, págs. 97-111.

39. **Beck, Kent.** *Extreme Programming Explained Embrance Change.* 1999. ISBN: 0201616416.

40. **VIM**. Vim the editor. [En línea] http://www.vim.org/about.php.

41. **Foundation, Django Software**. Django. *The web framework for perfectionists with deadlines.* [En línea] https://www.djangoproject.com/.

42. **Worsley, John y Drake, Joshua D.** *Practical PostgreSQL.* s.l. : " O'Reilly Media, Inc.", 2002.

42. **Courtney, Damon.** InstallJammer *Multiplatform Installer.* [En línea] [Citado el: 4 de febrero de 2016.] http://installjammer.com/docs/.

43. **Kawaguchi, Kohsuke.** Jenkins. [En línea] [Citado el: 6 de febrero de 2016.] https://jenkins.io/doc/.

44. **Huang, Yunwu y Ponzo, John J.** *Dependency injection by static code generation.* s.l.: Google Patents, #Jun.#~3 de 2014. US Patent 8,745,584.

45. *Design patterns: Elements of reusable object-oriented software.* **Vlissides, John, et al.** 120, 1995, Reading: Addison-Wesley, Vol. 49, p. 11.

46. **Unidad Docente de Ingeniería del Software.** Patrones del “Gang of Four”. *Facultad de Informática - Universidad Politécnica de Madrid.* 2013desi

47. *Software testing research: Achievements, challenges, dreams.* **Bertolino, Antonia.** 2007. 2007 Future of Software Engineering. págs. 85-103.

Anexo 1. Historias de usuario

Historia de Usuario	
Número: 1	**Nombre:** Proteger el código fuente del cliente de inventario.
Usuario: Desarrollador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 3	**Iteración Asignada:** 1
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite proteger el código fuente del cliente de inventario, así como empaquetar la aplicación. Esto se realiza para mitigar la suplantación de identidad.	
Observaciones:	

Tabla 46. HU-1. Gestionar la configuración del cliente de inventario.

Historia de Usuario	
Número: 2	**Nombre:** Proteger la base de datos del cliente de inventario.
Usuario: Desarrollador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 2
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite proteger la base de datos SQLite del cliente de inventario por medio del cifrado de la misma, para ello se utiliza la librería SQLCipher que implementa el algoritmo simétrico AES con 256 bits de clave. Esto se realiza para mitigar la manipulación de datos.	
Observaciones:	

Tabla 47. HU-2. Proteger la base de datos del cliente de inventario.

Historia de Usuario	
Número: 3	**Nombre:** Proteger los archivos de configuración del cliente de inventario.
Usuario: Desarrollador	

Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 1	**Iteración Asignada:** 2
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite proteger los archivos de configuración del cliente de inventario utilizando el cifrado de datos como técnica. Esto se llevará a cabo por medio del algoritmo simétrico AES, con 256 bits de clave, en modo CBC que implementa la librería PyCrypto. Esto se realiza para mitigar la manipulación de datos.	
Observaciones:	

Tabla 48. HU-3. Proteger los archivos de configuración del cliente de inventario.

Historia de Usuario	
Número: 4	**Nombre:** Procesar los datos de forma asíncrona y distribuida.
Usuario: Desarrollador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 1	**Iteración Asignada:** 3
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite procesar los datos enviados hacia el servidor de inventario de manera asíncrona y distribuida mediante el uso del sistema distribuido Celery. Esto se realiza para mitigar la denegación de servicios.	
Observaciones:	

Tabla 49. HU-4. Procesar los datos de forma asíncrona y distribuida.

Historia de Usuario	
Número: 5	**Nombre:** Establecer el protocolo de comunicación asíncrono.
Usuario: Desarrollador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 2	**Iteración Asignada:** 3
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	

Descripción: Permite establecer una comunicación asíncrona entre el cliente y servidor de inventario mediante el servidor de mensajería Rabbitmq. Esto se realiza para mitigar la denegación de servicios.
Observaciones:

Tabla 50. HU-5. Establecer el protocolo de comunicación asíncrono.

Historia de Usuario	
Número: 6	**Nombre:** Establecer la seguridad de la comunicación.
Usuario: Desarrollador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos Estimados: 1	**Iteración Asignada:** 4
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite autenticar los datos enviados entre el cliente y el servidor de inventario. Esto se realiza para mitigar la suplantación de identidad.	
Observaciones:	

Tabla 51. HU-6. Establecer la seguridad de la comunicación.

Historia de Usuario	
Número: 8	**Nombre:** Generar el instalador del cliente de inventario.
Usuario: Desarrollador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Puntos Estimados: 1	**Iteración Asignada:** 4
Programador responsable: Jorge Bárbaro Piñeiro Cruz y Javier Ricardo Ponce Pérez	
Descripción: Permite la generación de un instalador mediante el uso de la herramienta InstallJammer, así como la automatización de los procesos usando el integrador continuo Jenkins.	
Observaciones:	

Tabla 52. HU-8. Generar el instalador del cliente de inventario.