ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Maestría En Seguridad Informática Aplicada

"IMPLEMENTACIÓN DE UNA ARQUITECTURA DE ASEGURAMIENTO DE ALMACENES DE CLAVES PARA USO DE APLICACIONES EMPRESARIALES EN UNA EMPRESA DE TECNOLOGÍA DE CONSUMO MASIVO"

EXAMEN DE GRADO (COMPLEXIVO)

Previo a la obtención del título de:

MAGÍSTER EN SEGURIDAD INFOMÁTICA APLICADA

JIMMY JAVIER GOVEA VILLAO

GUAYAQUIL – ECUADOR

AÑO: 2016

AGRADECIMIENTO

Mi agradecimiento a mi esposa y a mis hijos por ser el motor en cada paso que como familia damos y a mis padres por la esencia de mi formación

DEDICATORIA

El presente proyecto lo dedico a mis hijos Sara Eugenia y Juan Pablo.

TRIBUNAL DE SUSTENTACIÓN

_____

Mgs. Lenin Freire

DIRECTOR DEL MSIA

_____

Mgs. Karina Astudillo

PROFESOR DELEGADO

POR LA UNIDAD ACADÉMICA

_____

Mgs. Ronny Santana

PROFESOR DELEGADO

POR LA UNIDAD ACADÉMICA

RESUMEN

Este proyecto tiene como principal premisa convertirse en una barrera para el transporte y almacenamiento de la información sensible, de manera que terceros no autorizados no puedan acceder a ella. Para esto se implementó un almacén centralizado con registros clave/valor que pueda ser accedido como un servicio web.

Para cumplir con la premisa expuesta los datos que viajan entre cliente y servidor van cifrados y codificados, con cada cliente se puede manejar un algoritmo de cifrado distinto agregándole complejidad a aquel que se quiera poner en el medio a analizar el tráfico.

Tomando en cuenta que esta solución se convierte en una piedra angular para las aplicaciones empresariales que necesitan de esta información sensible, es claro que tiene que conservar una disponibilidad muy alta. Aplicar una infraestructura de alta disponibilidad para un servicio requiere de conceptos y herramientas que coadyuven la consecución del objetivo de mantener respuestas exitosas aún cuando uno o más elementos dentro de la arquitectura fallen.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS Y SIMBOLOGÍA

HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
REST	Representional State Transfer
SFTP	Secure File Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Laguage
TCP	Transmission Control Protocol
XML	Xtensible Markup Language

INTRODUCCIÓN

En los tiempos de guerra, siempre ha sido necesario el ocultar la información de destinos no deseados, podemos hacer una analogía empresarial con la competencia y con terceros mal intencionados que desean una ventaja a costas de nuestra información. Es por esto, que se debe proteger la información sensible que necesitan los sistemas de información para comunicarse e interactuar, de manera que no sea perceptible para los intrusos que puedan estar interceptando dichos datos, teniendo esto en mente fue tomando forma la idea de implementar una arquitectura que provea a las diversas capas de aplicación que dan soporte a los servicios de negocio, una forma segura de obtener información sensible de forma centralizada.

Para implementar este proyecto se hizo uso de algunas herramientas como el cifrado, los servicios web, la comunicación por sockets, alta disponibilidad y replicación de datos. Lo que se buscó desde un inicio es proteger información sensible y mantener a la empresa segura y sin temor a que sus datos sean interceptados y visibles por terceros.

CAPÍTULO 1

GENERALIDADES

En este capítulo de forma general se describe el problema que motivó la implementación de la solución y se nombran principales características de la arquitectura de aseguramiento de almacenes de claves para uso de aplicaciones empresariales.

1.1 Descripción del problema

En el sector empresarial, se incrementa día a día la necesidad de brindar más servicios al usuario final, para poder soportarla en el área de tecnologías de información por lo general se agregan más servidores a la granja que se deben comunicar entre sí; esto hace que sea necesario que los servidores tengan un mecanismo para poder compartir información, sin poner en riesgo los datos sensibles a los que se debe acceder y que serán usados en los procesos de soporte a estos servicios.

Cada proceso necesita de información esencial para poder ejecutarse correctamente, en algunos casos claves, direcciones IP, información de detalle de destinos secretos que se deben alcanzar que no pueden permanecer visibles para todos los usuarios en un archivo de configuración vulnerable que pueda ser extraído del servidor origen por alguna persona mal intencionada que obtenga acceso temporal lícito al activo de información, y que pueda ser usado posteriormente con fines maliciosos.

1.2 Solución propuesta

Se propone un servicio de consulta de datos como almacén centralizado de registros tipo clave/valor que tenga alta disponibilidad y que exponga varias tecnologías como interfaces de acceso; que cambie su comportamiento dependiendo de la dirección IP de origen del requerimiento y que pueda devolver los datos cifrados con un algoritmo diferente de acuerdo a la configuración correspondiente.

Obteniendo como principales beneficios, que cualquier tipo de programa, aplicación o proceso que necesite de información sensible para su correcto funcionamiento no ponga en riesgo su divulgación exponiéndola en el código fuente o en archivos de configuración, que podrían fácilmente estar disponibles en un software control de versiones sin mayor restricción, adicionalmente se busca que la acción de realizar un cambio de clave, sea

mucho más automatizada, y no sea una tarea que demore mucho tiempo e improductiva además de dejar a la empresa sujeta a riesgos de no cambiar un fuente que cause que uno o más procesos dejen de funcionar.

CAPÍTULO 2

CONCEPTOS APLICADOS

En el capítulo 2 vamos a revisar los principales conceptos utilizados en la arquitectura de aseguramiento de almacenes de claves para uso de aplicaciones empresariales, y justificar su uso en la solución.

2.1 Algoritmos de Cifrado

Para que la información sensible que se desea intercambiar entre sistemas de información no pueda ser entendida por entes externos que puedan interceptarla, se necesita pasar los datos por un proceso de enmascaramiento o cifrado de manera que no tenga ningún valor para quien no sea su destino original.

Los algoritmos de cifrado se pueden dividir en dos grandes grupos: **simétricos**, cuando usan la misma palabra clave para cifrar y descifrar y **asimétricos** cuando usan diferentes palabras clave, una para el proceso de cifrado y la otra para el proceso de descifrado [1].

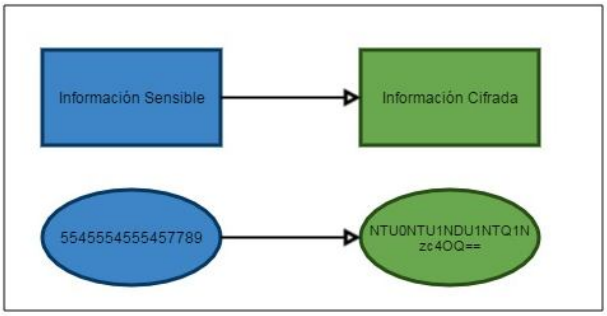

Figura 2.1 Cifrado de Información

Para la implementación de esta solución, por su naturaleza, al ser servidores administrados en su totalidad por la empresa, se optó por el uso de algoritmos simétricos, en la siguiente tabla se muestran los algoritmos más conocidos [2]:

Tabla 1 Algoritmos simétricos

Algoritmo	Tamaño de bloque	Tamaño de clave	Cantidad de claves
DES	64 bits	64 bits	1
3/DES	168 bits	64 bits	3
Blowfish	64 bits	32-448 bits	1
AES	128/192/256 bits	128/192/256 bits	1

Adicionalmente al cifrado aplicado, en esta implementación se agregó una codificación en **base64** de la cadena de bytes resultantes posterior al cifrado de los datos, de manera que la cadena final no tenga caracteres especiales [3] que puedan ocasionar algún problema en la transmisión de la información entre los entes autorizados.

2.2 Tecnologías de Interfaces de Acceso

La comunicación entre los sistemas en la arquitectura propuesta se realiza de tres formas: mediante sockets TCP, servicios web SOAP y servicios web REST, a continuación se revisará el funcionamiento básico de cada una de estas tecnologías:

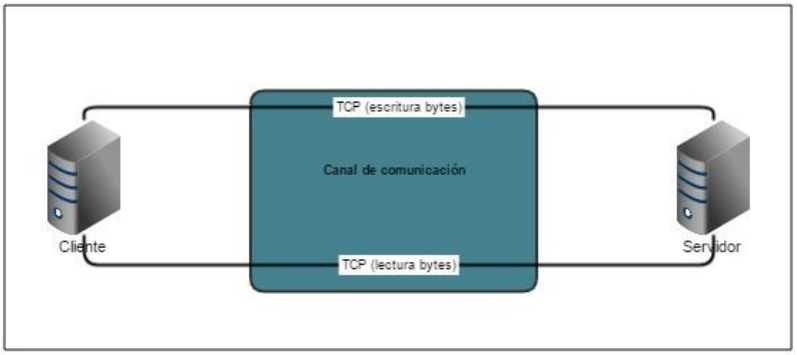

Figura 2.2 Interfaz TCP

En la interfaz **TCP** se establece un canal de comunicación, en el cual se pueden enviar y recibir datos, es responsabilidad del cliente implementar el algoritmo correspondiente adecuado para el intercambio de información que coincida con lo esperado por el servidor que previamente debió haber implementado el algoritmo de atención de requerimientos.

Es la forma de comunicación base en las primeras implementaciones de arquitecturas cliente-servidor, la complejidad real de este tipo de soluciones es que no se trata de un estándar de la industria que se pueda reutilizar fácilmente.

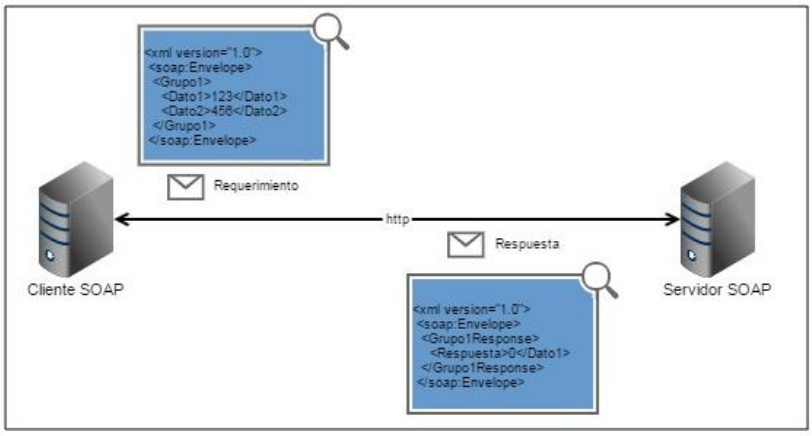

Figura 2.3 Interfaz SOAP

En la interfaz **SOAP**, los requerimientos se envían haciendo uso del protocolo http, y un mensaje **XML** que cumple con un formato de sobre estándar, lo que va dentro del sobre es la información del método a ejecutar y los parámetros correspondientes a ese método, como respuesta se obtiene otro sobre que tiene dentro la respuesta del método ejecutado. Las herramientas que coadyuvan al uso de este protocolo están en una etapa madura, tomando en cuenta la antigüedad y amplia aceptación que ha tenido en la industria esta tecnología.

La mayoría de los servicios actualmente están basados en SOAP, por este motivo ha sido considerado dentro de la implementación de este proyecto.

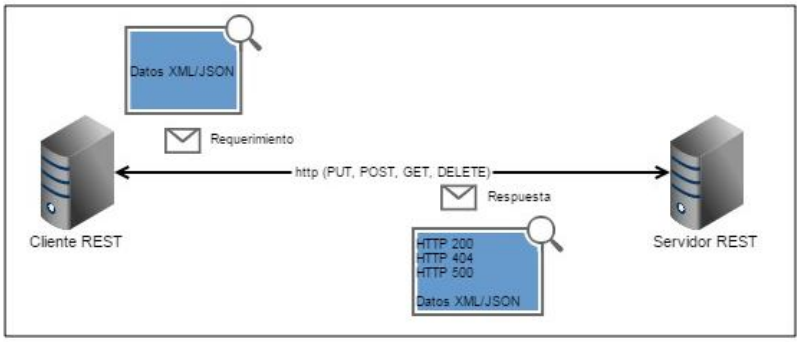

Figura 2.4 Interfaz REST

En la interfaz **REST** el intercambio de información también se realiza utilizando como transporte el protocolo **HTTP**, pero potenciando las características intrínsecas del mismo, usando esencialmente los métodos nativos del protocolo, de tal forma que el método **GET** nos sirve para obtener un recurso, como puede ser la información de un cliente, el método **PUT** nos sirve para crear un nuevo recurso, puede ser utilizado en la creación de un nuevo cliente, el método **DELETE** nos puede ayudar al momento en que deseemos dar de baja al cliente y el método **POST** lo podemos utilizar para actualizar información de clientes, entre otras funcionalidades.

A diferencia de SOAP, con este protocolo no se está supeditado a enviar un sobre XML estándar, podríamos enviar cualquier recurso, siendo los más utilizados documentos **XML** no estandarizados y documentos **JSON**. La importancia de esta tecnología se da porque va desplazando SOAP, formado parte de toda nueva implementación por ser más ligera.

CAPÍTULO 3

DESARROLLO DE LA SOLUCIÓN

En el tercer capítulo, presentamos a detalle los componentes que integran la solución y como se resolvieron varios retos de seguridad que se presentaron y dieron a lugar a la ejecución de este proyecto.

3.1 Almacén Centralizado

Se implementó un almacén centralizado de datos con registros tipo clave→valor donde la clave es el argumento que se recibe por las interfaces desde los clientes, y el valor es la información sensible que deseamos proteger y que los clientes necesitan para ejecutar algún tipo de acción. El almacén centralizado pasa a ser un activo de tecnologías de información muy importante para la empresa y su operación, la indisponibilidad del mismo haría que todos los sistemas que dependan de él dejen de funcionar

correctamente. Es por este motivo que es necesario que la solución tenga **alta disponibilidad**, apalancándose de tecnologías como HaProxy que es un software de balanceo de tráfico que redirige los requerimientos a varios destinos, y mantiene el control de cuáles son los destinos que están respondiendo de forma adecuada y cuáles no, Keepalived entre el par de balanceadores, verificando con una frecuencia definida que su par se encuentre operativo, si uno de los dos se cae, el otro toma el control de la IP y se apodera de ella, este proceso es transparente para los clientes que se conectan a través de la IP flotante [4].

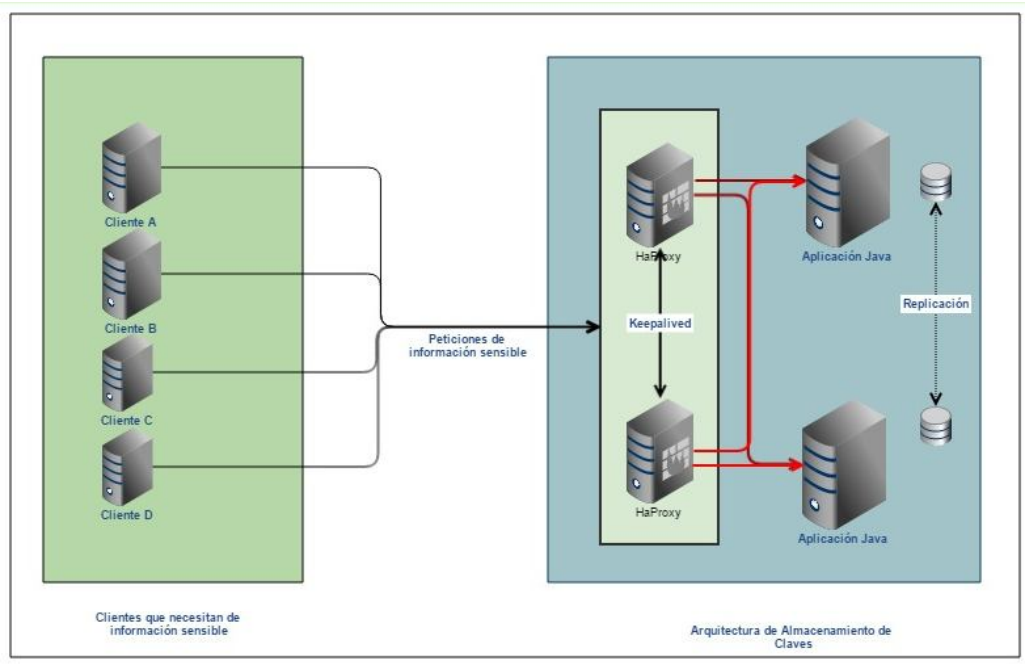

Figura 3.1 Diagrama general solución

Para garantizar la disponibilidad del repositorio existe un esquema replicación con frecuencia configurable, y la aplicación dentro de su

metadata tiene configurado un datasource con failover consiguiendo con esto que si el repositorio primario falla se tome la información del repositorio secundario, de esta manera las aplicaciones clientes operan sin inconvenientes, aún cuando elementos dentro de la arquitectura presentada dejen de funcionar.

3.2 Integración con administrador de claves

Uno de los principales casos de uso para esta solución es el de recuperación de claves para acceder a servicios remotos, por ejemplo para realizar una transferencia de archivos mediante SFTP, o iniciar una sesión hacia una base de datos mediante un cliente SQL.

En estos casos se recibe como parámetros clave=**usuario**, retornando como resultado la contraseña, si se trata de otro servidor el destino se especifica de la siguiente manera: clave=**ip.usuario** obteniendo como resultado la contraseña del destino que se desea alcanzar.

Existe un componente que se encarga de la interacción con los sistemas externos, particularmente como parte de la solución se implementó la integración con el administrador de claves de código abierto KeePass, utilizando el API Openkeepass [5], de manera que al realizar un cambio de claves, el cambio se aplica automáticamente en tres lugares:

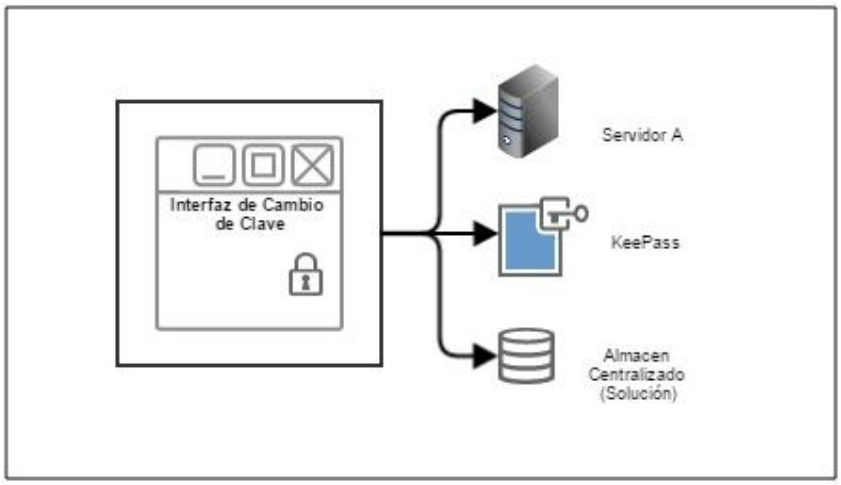

Figura 3.2 Administración de Claves

- El servidor donde se desea cambiar la clave
- En el archivo que contiene la base de datos de claves
- El almacenamiento centralizado de la solución

3.3 Cifrado de datos

Todos la comunicación entre cliente y servidor es cifrada, se puede configurar un algoritmo de cifrado distinto por cada IP de un cliente registrado. El servidor realiza validación de IP de origen, en caso de que la IP del origen no exista no se responderá la petición. De lado del cliente, antes de enviar la petición se toma el valor de la clave a buscar, se lo cifra con el algoritmo correspondiente, posteriormente se lo codifica en base64 y se envía la petición al servidor usando las distintas interfaces de acceso.

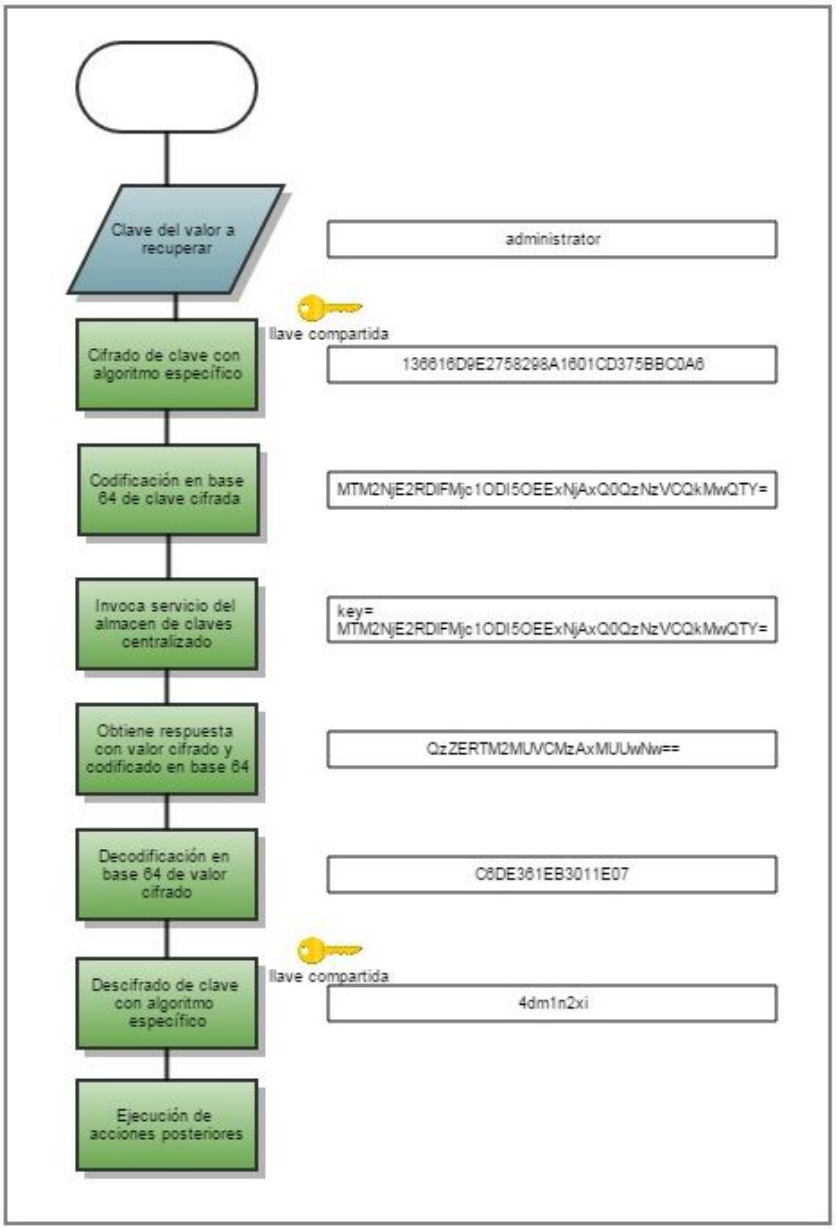

Figura 3.3 Flujo de eventos en el cliente

El cliente toma la respuesta del servidor y aplica el proceso inverso, una vez que obtiene el valor correspondiente, pasa la posta al proceso llamante.

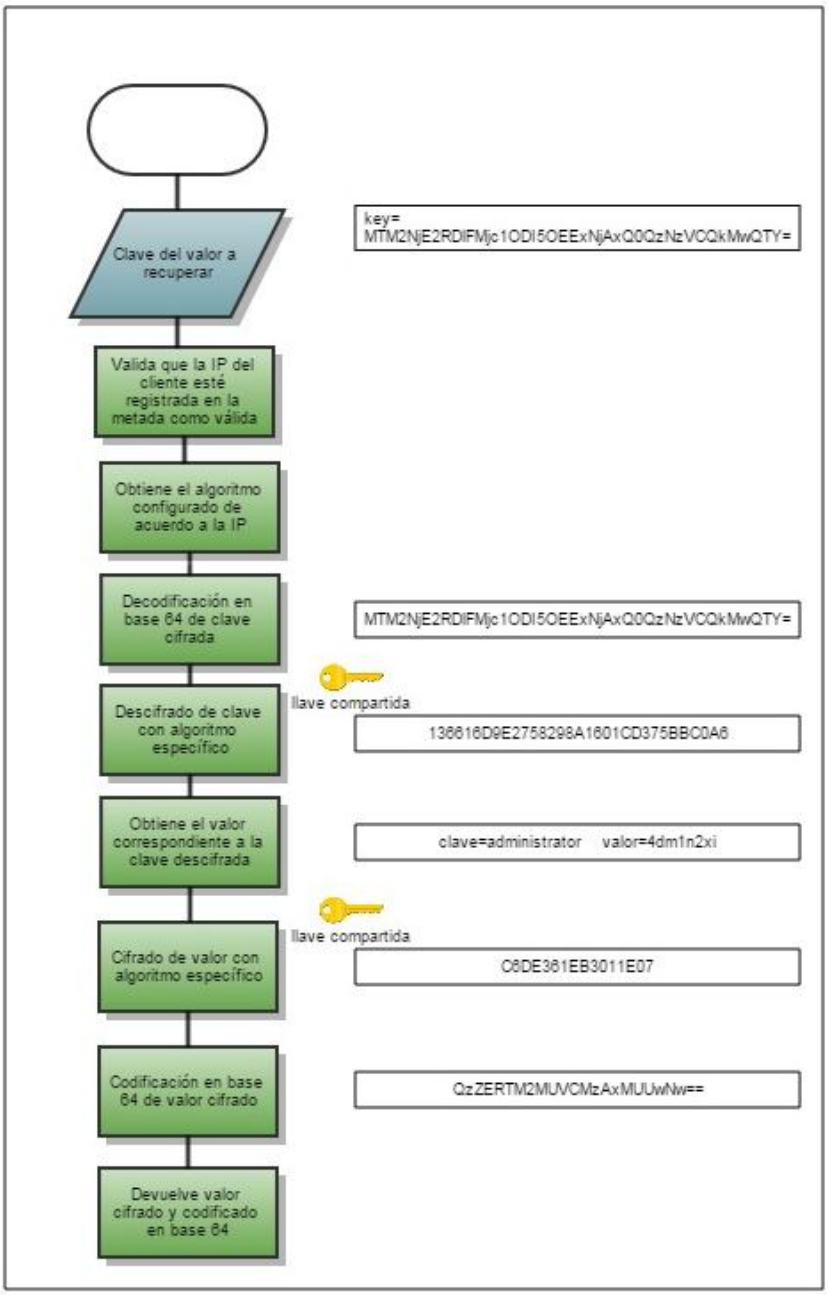

Figura 3.4 Flujo de eventos en el servidor

Las consideraciones que hay que tener en cuenta para el buen funcionamiento del esquema son las siguientes:

- El cliente y el servidor deben usar la misma llave para realizar cifrados y descifrados, si se necesita cambiar la clave, se tendrá que hacer en cliente y en servidor.
- La metadata de configuración debe coincidir, si el cliente A aplica el algoritmo de cifrado X, y el servidor en su configuración indica que para la IP del cliente A, el algoritmo de cifrado que se debe aplicar es Y, los resultados obtenidos serán inesperados.

3.4 Técnicas para proteger fuentes

Para conseguir un nivel extra de protección, en algunos fuentes muy conocidos, para tratar de minimizar las posibilidades de que se pueda estar al tanto de la forma en que trabaja el almacén de claves centralizado se optó por mecanismos de ocultamiento de información, en la siguiente tabla se puede visualizar qué tecnología se aplicó para proteger cada tipo de fuente.

Tabla 2 Mecanismos de protección de fuentes

Tipo de fuente	Mecanismo de protección
Java	Ofuscamiento
Perl	Conversión a Binario
Php	Conversión a Binario
Shell	Conversión a Binario
PL/SQL	Encriptado (Wrap Utility)

En los casos de lenguajes de programación interpretables como Bash Shell, Perl y Php, se optó por transformarlos a binarios con diversas herramientas que se encuentran en la red y otras nativas [6], para el caso de Java se optó por el ofuscamiento y hacer que su lectura sea difícil de realizar, tomando en cuenta que hoy en día hay herramientas como jdgui que hace que el descompilar de los archivos bytecode que son generados por el compilador lo pueda realizar cualquier persona sin mayor conocimiento de la tecnología [7], para el caso específico de PL/SQL se usó una herramienta propietaria de Oracle que es el PL/SQL Wrapper [8] que hace que el código no se pueda visualizar, sólo se puede ejecutar.

CAPÍTULO 4

ANÁLISIS DE RESULTADOS

En el capítulo actual se expondrán los principales resultados obtenidos con la implementación de la arquitectura de almacén de claves centralizado, se presentarán un apartado en el que explicará como la información sensible está ahora protegida de poder ser descubierta fácilmente por terceros, y daremos una mirada a la mejora en la productividad del equipo de operaciones y proyectos, gracias a la aplicación de la solución.

4.1 Información sensible protegida

Al realizar búsquedas de código fuente en el software de versiones, o cuando se brindaba un acceso temporal autorizado a los servidores de producción a terceros, antes de la solución encontrábamos varias entradas que contenían información sensible cómo:

- Claves de acceso a servicios/servidores
- IPs de servidores de producción
- Puertos de servicios críticos

```
#Antes
usuario=user01
clave=miclave

server=192.168.1.1
puerto=7777

#Ahora
usuario=user01
clave=obtenerValorRepositorio(usuario)

server=obtenerValorRepositorio("SERVICIO_A")
puerto=obtenerValorRepositorio("PUERTO_SERVICIO_A")
```

Figura 4.1 Información sensible dentro de los fuentes

Aplicando la solución, ahora los fuentes no contienen información sensible y se elimina el riesgo que terceras personas que tengan acceso al software de control de versiones pueda visualizar información de la que no es un destinatario directo. Los nuevos activos de información son validados para que no contengan quemada información dentro de sus fuentes que podría causar un problema de seguridad mayor.

4.2 Productividad al usar la solución

Dentro de las actividades necesarias para cambiar una clave de un sistema, existían varios pasos a seguir, el cambio de la clave per sé, búsqueda en todos los sistemas locales y remotos si había algún código fuente que esté

haciendo referencia a una credencial, el cambio de esa clave quemada en todos esos fuentes, la re-compilación de los fuentes en los casos que aplicara, en muchos ocasiones se omitían fuentes por motivos de desconocimiento, que posteriormente provocaban afectaciones de servicio, todos estos pasos consumían bastante tiempo, tomando en cuenta que se hacían de forma manual, en el siguiente cuadro se muestra un ejemplo del tiempo promedio que tomaba un cambio de clave en un sistema de complejidad media, con la solución presentada en este documento ese tiempo fue reducido a lo que toma el primer paso, permitiendo a la operación realizar cambios de claves de forma inmediata minimizando los riesgos que generan los cambios manuales.

Tabla 3 Tiempos de cambios de clave antes de la solución

Tarea	Unidad	Cantidad de ítems	Horas hombre
Cambio de clave	clave	1	1
Búsqueda de fuentes	archivo fuente	30	8
Cambio en fuentes	archivo fuente	30	8
Recompilación	archivo fuente	3	8
Fuentes faltantes	archivo fuente	5	8
Total de horas			33

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Con la consecución de la arquitectura descrita en el presente proyecto la información sensible empresarial está más segura imposibilitando el acceso no autorizado a la misma.

2. Al conseguir manejar los procesos de cambio de clave en una hora y sin los riesgos asociados a los cambios manuales, los distintos actores responsables de un cambio pueden dedicarse a otras labores, consiguiendo una mayor productividad en el departamento.

Recomendaciones

1. En la medida de lo posible usar soluciones como la presentada en este proyecto, a fines de evitar que se "queme" información sensible en el código.

2. La tecnología utilizada para la creación de esta solución en su momento era la que mejor se adaptaba a lo requerido, sin embargo con los avances informáticos veloces que vivimos es importante volver al análisis para evolucionar la arquitectura de solución utilizando nuevos conceptos.

BIBLIOGRAFÍA

[1] Jason Albanese y Wes Sonnenreich, Network Security Ilustrated, Mc Graw Hill, 2004

[2] Abhay Bhargav y B.V. Kumar, Secure Java For Web Application Development, CRC Press, 2010

[3] Prabhakar Chaganti, Rich Helms, Amazon SimpleDB Developer Guide, Packt Publishing, 2010

[4] How to Setup Haproxy wiht Keepalived http://dasunhegoda.com/how-to-setup-haproxy-with-keepalived/833, fecha de consulta Diciembre del 2015

[5] Openkeepass: Java API for KeePass 2.x http://slackspace.de/articles/openkeepass-java-api-for-keepass-2-x/, fecha de consulta Diciembre del 2015

[6] Convert PHP file to binary http://stackoverflow.com/questions/1845197/convert-php-file-to-binary, fecha de consulta Diciembre del 2015

[7] Godfrey Nolan, Bulletproof Android Practical Advice for Building Secure Apps, Addison Wesley, 2014

[8] Cómo ocultar el código de PL/SQL http://www.oracle.com/technetwork/es/articles/como-ocultar-el-codigo-de-plsql-096315-esa.html, fecha de consulta Diciembre del 2015