Universidad de las Ciencias Informáticas

Facultad 5

Título: Mecanismos de seguridad para el Middleware del SCADA "Guardián del ALBA".

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

Autores: Rosalbis Guibert Nápoles

Yusniel Cárdenas del Valle

Tutor: Ing. Maikel Pérez Javier

Co-Tutor: Ing. Ana Silvia Tellería Martínez

Ciudad de la Habana, Junio 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yusniel Cárdenas del Valle	**Rosalbis Guibert Nápoles**	**Ing. Maikel Pérez Javier**
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

Datos de Contacto

Ing. Maikel Pérez Javier

Profesor Instructor con dos años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI) y dos años de experiencia en la producción de software, específicamente en el desarrollo del Middleware del Sistema Supervisor de Procesos Automatizados (SCADA).

"Para ser exitoso no tienes que hacer cosas extraordinarias. Haz cosas ordinarias, extraordinariamente bien".

"El camino hacia el conocimiento está lleno de dudas; el camino hacia la ignorancia está lleno de certezas".

Agradecimientos

A mis padres por haberme dado la oportunidad de vivir cada uno de estos momentos, que serán los más importantes de mi vida.
A mis tíos y tías de la Habana, que son tantos que no podría mencionarlos a todos, por haberme apoyado durante tanto tiempo y por haber cumplido tan bien su papel de tutores, dándome apoyo y consejos.
A mis amistades, a Ana, Danelys, Isle la loca, a Faby, inclusive a Mario, porque sus críticas me hicieron mejor cada día. A Rislaidy por apoyarnos con la tesis.
A mis compañeros de grupo, cada uno de los grupos por los que pasé, y las chicas de mis cuartos…que son muchas también.
A Yusniel, mi compañero en este trabajo…viste!! que todo se puede…jaja
A mis tutores, por su apoyo.
A la escuela por los inolvidables momentos que viví aquí…
A todos los que me han apoyado y han hecho que mis días aquí se tornen más fáciles.
Gracias

Rosalbis

A mis padres Victoria Gardenia y Julio Roberto por su constante amor, apoyo y preocupación.
A mis dos hermanas por el simple hecho de existir y llenar de alegría mi corazón.
A Dora, a Juanito, a Yasminda, a Estelbina, a Gladys, a Abdiel y a "Mamaita", por ser un apoyo en todo momento.
A mi familia por darme amor y apoyo.
A mi tutor Maikel por ser amigo y apoyarme en momentos difíciles.
A mi hermano y amigo Rislaidy por ayudarme a llegar hasta el final de un sueño y poder seguir adelante.
A mi compañera de tesis por su preocupación y persistencia en el trabajo.
A mi amigo Julio por ayudarme siempre que lo necesitaba.
A mi novia Mavis por hacerme feliz, por ser paciente, por darme apoyo en la toma de decisiones, por ser una amiga.
A mis compañeros de aula.

A la UCI por haberme dado la posibilidad de hacer realidad un sueño…

Yusniel

Dedicatoria

A la memoria de mis abuelos, que me apoyaron aun sin estar aquí conmigo.
A mis padres
A mi hermana, mi mejor amiga
A mi hermanito, chiquito solo por edad
A todos...
Rosalbis

A mis padres Gardenia y Julio, por su amor y confianza,
A mis abuelos que ya no están,
A mis especiales amigos Rislaidy, Julio, Maithe, Nery y Annelis.

Yusniel

Resumen

Entre los principales proyectos productivos de la Universidad de las Ciencias Informáticas (UCI) se encuentra el Proyecto SCADA, ahora llamado "Guardián del ALBA", con el objetivo de desarrollar un sistema para la Supervisión, Control y Adquisición de Datos (SCADA). Este proyecto surge a partir de la necesidad de la empresa Petróleos de Venezuela - Sociedad Anónima (PDVSA) de automatizar y gestionar sus procesos industriales orientado a la producción de petróleo y gas. El proyecto está compuesto por diferentes líneas de trabajo. Dentro de estas líneas se encuentra la línea de Middleware. Esta línea desarrolla un software que se sitúa entre las capas de aplicación y transporte y su principal función es la de establecer los métodos de comunicación entre las aplicaciones en ejecución desde cualquier punto del sistema, garantizando que la localización de cada una de ellas sea transparente a las demás. Este software no cuenta con mecanismos de seguridad para garantizar la protección de la información que por él transita. Para dar solución a la polémica existente surge la siguiente interrogante: ¿cómo dotar al Middleware del "Guardián del ALBA" de mecanismos que lo hagan un sistema de comunicación seguro? Para dar solución a la interrogante planteada anteriormente, es necesaria la incorporación de mecanismos de seguridad que permitan obtener un sistema de comunicación seguro entre los módulos del SCADA.

Como resultado se obtiene un middleware con los mecanismos de seguridad necesarios para lograr una comunicación segura entre los módulos que componen el Guardián del ALBA.

PALABRAS CLAVE

Middleware, SCADA, Mecanismos de Seguridad.

Índice

Introducción

No fue hasta finales de los años 80 que las personas comenzaron a tomar en serio el tema de la seguridad en redes de computadoras de propósito general. Mientras que por una parte Internet iba creciendo rápidamente con redes importantes que se ligaban a ella, por otra, el auge de la informática unido a factores menos técnicos, iba produciendo un aumento espectacular en el número de piratas informáticos.

“El 22 de noviembre de 1988 Robert T. Morris protagonizó el primer gran incidente de la seguridad informática: uno de sus programas se convirtió en el famoso *worm* o gusano de Internet” *(Facultad de Ciencias Exactas y Naturales y Agrimensuras, 2001)*. Los miles de computadores conectados a la red se vieron inutilizados durante días, y las pérdidas fueron estimadas en millones de dólares.

Desde ese momento, el tema de la seguridad en temas referentes a las redes y sistemas distribuidos ha sido un elemento a tener muy en cuenta por cualquier responsable o administrador de sistemas informáticos.

En los últimos años el tema de la seguridad en las redes de computadoras se ha tornado un asunto de primera importancia dado el incremento de las prestaciones de estas. Los "incidentes de seguridad" que se reportan continuamente crecen cada vez a un ritmo más acelerado, a la par de la masificación de Internet y de la complejidad del software que se desarrolla. Una red mal configurada o con la protección física inadecuada puede suponer un riesgo para la seguridad.

Cada día se hace patente la preocupación por los temas relativos a la seguridad. Hoy cualquier aprendiz de pirata puede conectarse a un servidor web, descargar un par de programas y ejecutarlos contra un servidor desprotegido. Con un poco de suerte, esa misma persona, puede conseguir un control total sobre un servidor de varios miles de dólares, probablemente desde su computadora con Windows 98 y sin saber nada de él.

Algo que atenta contra la seguridad es la complejidad de los sistemas. Hoy en día se desarrollan sistemas de redes más complejos y por tanto más vulnerables en cuanto a su seguridad. Un sistema complejo es más difícil de asegurar y proporciona mayor cantidad de “puertas” a los atacantes.

Actualmente los sistemas distribuidos representan un nuevo reto en el desarrollo de software. Estos sistemas han supuesto una importante revolución en el paradigma de la computación distribuida, ahora que los roles de cliente y servidor están desapareciendo. El nuevo escenario consiste en sistemas en los cuales todos los elementos de la red se consideran iguales y, en la mayoría de los casos, los mecanismos de comunicación no están basados en una infraestructura previa, sino en redes que se forman dinámicamente. Los sistemas distribuidos son sistema muy complejos, y desde el punto de vista de la seguridad, son extremadamente vulnerables a una

amplia variedad de ataques.

El tráfico de información entre los diferentes componentes de estos sistemas obliga el uso de una tecnología de comunicación que garantice el eficiente intercambio de información entre los procesos en el menor tiempo posible.

Una de las claves para el éxito de los sistemas distribuidos es facilitar la tarea de implementación de servicios mediante la abstracción de las fragilidades de este, permitiendo al diseñador y al programador olvidarse de ellas, reduciendo la complejidad de las aplicaciones y un abaratamiento de los costes mediante un middleware. Este software debería ocultar la complejidad de la infraestructura subyacente mientras proporciona interfaces abiertas para el desarrollo de aplicaciones a terceros.

El *Middleware* es un software que se sitúa entre las capas de aplicaciones y de transporte y su principal función es la de establecer los métodos de comunicación entre las aplicaciones en ejecución desde cualquier punto del sistema, garantizando que la localización de cada una de ellas sea transparente a las demás. Facilita la computación distribuida, mediante conexión de múltiples aplicaciones para crear una aplicación mayor sobre la red.

El middleware ofrece un conjunto de servicios que hacen posible el funcionamiento de los sistemas en general. Estos servicios pueden convertirse en puntos de ataque para aquellos usuarios o sistemas que intenten acceder a las prestaciones que ofrece.

En nuestro país, en el año 2002 y por idea del Comandante en Jefe Fidel Castro, nace la Universidad de las Ciencias Informáticas (UCI) con la misión de formar profesionales altamente calificados en la rama de la informática comprometidos con la patria, para la producción de software y la informatización de la sociedad cubana.

La UCI cuenta con un modelo de formación donde se vinculan la docencia, la investigación y la producción de software, mediante la vinculación de los estudiantes a proyectos productivos y la especialización de las facultades en perfiles productivos. La facultad 5 desarrolla software relacionados con Entornos Virtuales y Automática.

Entre los principales proyectos productivos que desarrolla esta facultad está el SCADA “Guardián del ALBA” que surge a partir de la necesidad de la empresa Petróleos de Venezuela - Sociedad Anónima (PDVSA) de automatizar y gestionar sus procesos industriales. Este proyecto está divido en diferentes líneas de trabajo: Drivers, Gráfico y Diseño, Base Datos Histórica, Base Datos en Tiempo Real, Monitoreo, Reportes, Seguridad y Middleware, además de otros que simultáneamente se desarrollan en Venezuela.

El middleware como línea de trabajo tiene la función de garantizar un método de comunicación

entre los módulos del SCADA que se encuentren ejecutándose en cualquier punto del sistema, haciendo sus localizaciones transparente a los demás. Para el intercambio de información entre los módulos del sistema se identifican consumidores, proveedores, se crean proxy, canales de transmisión, se guardan referencias a objetos para hacer llamadas a procedimientos remotos, etcétera. En el instante en que se dispone a transmitir o recibir información se crean brechas de seguridad que comprometen o impiden estas acciones. Estas brechas pueden ser aprovechadas por cualquier sistema o agente externo, dando la posibilidad de leer o modificar los datos que se envíen de un módulo a otro. Otra característica notable de los sistemas distribuidos y del middleware en particular es la interoperabilidad, una vía más pragmática que otras, como el intento de hacer compatibles todos los sistemas a nivel de código fuente, centrándose exclusivamente en añadir nuevas capas de middleware que intentan darle a todos los sistemas un aspecto y unas funciones similares, o buscar la forma de hacer que todos los sistemas sean intercambiables. Este servicio presenta debilidades en cuanto se refiere a la seguridad que deben ser tratadas para no ver afectado el funcionamiento del sistema.

Este software no cuenta con mecanismos de seguridad para garantizar la protección de la información que por él transita. El middleware puede ser blanco fácil para los ataques de terceros provocando un mal funcionamiento del sistema. Estos ataques se pueden manifestar como: lectura de información no autorizada, accesos a recursos restringidos, desvío de información que puede comprometer la estabilidad del sistema, entre otros.

Para dar solución a la **situación problémica** existente, que involucre todas las respuestas posibles para obtener un middleware seguro, se plantea el siguiente **Problema Científico:** ¿Cómo dotar al Middleware del “Guardián del ALBA” de mecanismos de seguridad que lo hagan un sistema de comunicación seguro?.

El **objeto de estudio** de la investigación es: Mecanismos de seguridad y el **campo de acción** está dirigido a los Mecanismos de seguridad en el middleware del “Guardián del ALBA”.

Para dar solucionar siguiente problema científico, nos plantemos como **objetivo general** incorporar mecanismos de seguridad al Middleware actual que permitan obtener un sistema de comunicación seguro entre los módulos del sistema.

Para poder cumplir con el objetivo general se plantean las siguientes **tareas de investigación**:

- Realizar un análisis de los mecanismos de seguridad existentes.
- Describir los protocolos que implementan los mecanismos de seguridad y estén relacionados con la tecnología TAO (The ACE ORB).
- Realizar un análisis de los mecanismos de seguridad que brinda TAO.

- Identificar las vulnerabilidades de seguridad existentes en el Middleware actual.
- Definir los mecanismos de seguridad que serán incorporados, dependiendo de las vulnerabilidades encontradas.
- Incorporar en el Middleware actual una solución que disminuya las brechas de seguridad existentes.

Para el desarrollo de estas tareas se aplican una serie de **métodos científicos**, tanto teóricos como empíricos:

Teóricos:

Histórico-Lógico: Durante la revisión de las bibliografías utilizadas para la conceptualización de los diferentes temas.

Análisis-Síntesis: Durante la definición de las funcionalidades y caracterización de los mecanismos necesarios en el Middleware.

Inducción-Deducción: Durante la revisión y justificación de los mecanismos a utilizar en el middleware.

Empíricos:

Entrevistas: A los asesores del proyecto y a los líderes de la línea de Middleware por ser los de mayor experiencia en el desarrollo de de este tipo de software.

El trabajo esta estructurado de la siguiente forma:

Capítulo I: Fundamentación Teórica

En este capítulo se hace referencia a la concepción general del sistema. Se definen los conceptos de Sistemas Distribuidos, los sistemas SCADA, los middlewares como parte de los sistemas SCADA y la Seguridad como parte importante de cualquier sistema.

Capítulo II: Descripción de los mecanismos de seguridad a utilizar.

En este capítulo se hace referencia a los mecanismos de seguridad y algunos de los protocolos que implementen estos mecanismos. Al final se le agrega las herramientas que se utilizaron para desarrollar el trabajo en general.

Capítulo III: Descripción y Aplicación de la solución propuesta.

En este capítulo se definen las brechas de seguridad que presenta el middleware y la solución del problema.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En el presente capítulo se describen los conceptos de Sistemas Distribuidos, SCADA, Middleware y Seguridad. Se mencionan características y detalles de los mismos como parte del conocimiento básico que se debe tener para comprender el problema que se desea resolver.

1.2 Sistemas Distribuidos

"El desarrollo de los sistemas distribuidos vino de la mano de las redes locales de alta velocidad a principios de 1970" *(Rojo, 2003).* Recientemente, la disponibilidad de computadoras personales de altas prestaciones, estaciones de trabajo y ordenadores servidores ha depuesto a los ordenadores centralizados multiusuario. Esta tendencia se ha acelerado por el desarrollo de software para sistemas distribuidos, diseñados para soportar aplicaciones distribuidas. Este software permite a los ordenadores coordinar sus actividades y compartir los recursos del sistema (hardware, software y datos).

1.2.1 Definiciones

"Un sistema distribuido se define como una colección de computadores autónomos conectados por una red, y con el software distribuido adecuado para que el sistema sea visto por los usuarios como una única entidad capaz de proporcionar facilidades de computación" *(Rojo, 2003).*

Otra forma de hacer referencia a los sistemas distribuidos seria: Una colección de elementos de cómputo autónomos que se encuentran físicamente separados y no comparten una memoria común, se comunican entre sí a través del intercambio de mensajes utilizando un medio de comunicación. *(Instituto Tecnológico de Colima, 2004)*.

La computación distribuida es el conjunto de modelos, técnicas y herramientas computacionales que ayudan a resolver los problemas planteados por los sistemas distribuidos en cuanto a la generación, el procesamiento y la utilización de la información requerida para su funcionamiento.

Los sistemas distribuidos se implementan en diversas plataformas hardware, desde unas pocas estaciones de trabajo conectadas por una red de área local hasta Internet, o una colección de

redes de área local y de área extensa interconectadas, que pueden llegar a enlazar millones de ordenadores.

Las aplicaciones de los sistemas distribuidos varían desde la provisión de capacidad de cómputo a grupos de usuarios, hasta sistemas bancarios, comunicaciones multimedia y abarcan prácticamente todas las aplicaciones comerciales y técnicas de los ordenadores.

Los requisitos de dichas aplicaciones incluyen un alto nivel de fiabilidad, seguridad contra interferencias externas y privacidad de la información que el sistema mantiene. Se deben proveer accesos concurrentes a bases de datos por parte de muchos usuarios, garantizar tiempos de respuesta, proveer puntos de acceso al servicio que están distribuidos geográficamente, potencial para el crecimiento del sistema para acomodar la expansión del negocio y un marco para la integración de sistema usados por diferentes compañías y organizaciones de usuarios.

1.2.2 Características claves de los sistemas distribuidos

"Son seis las características principales responsables de la utilidad de los sistemas distribuidos. Se trata de compartición de recursos, apertura (openness), concurrencia, escalabilidad, tolerancia a fallos y transparencia" *(Rojo, 2003).*

En las siguientes líneas se aborda cada una de ellas de forma resumida: *(Rojo, 2003).*

- Compartición de Recursos

  En un sistema distribuido pueden compartirse, desde componentes hardware (discos, impresoras y otros dispositivos) hasta elementos software (ficheros, ventanas, bases de datos y otros objetos de datos).Los recursos en un sistema distribuido están físicamente encapsulados en una de las computadoras y sólo pueden ser accedidos por otras computadoras mediante las comunicaciones (la red).

- Apertura

  Un sistema informático es abierto si el sistema puede ser extendido de diversas maneras. Un sistema puede ser abierto o cerrado con respecto a extensiones hardware (añadir periféricos, memoria o interfaces de comunicación, etcétera) o con respecto a las extensiones software (añadir características al sistema operativo, protocolos de comunicación y servicios de compartición de recursos, etcétera).

- Concurrencia

  Cuando existen varios procesos en una única maquina decimos que se están ejecutando concurrentemente. Si el ordenador esta equipado con un único procesador central, la concurrencia tiene lugar entrelazando la ejecución de los distintos procesos. Si la

computadora tiene N procesadores, entonces se pueden estar ejecutando estrictamente a la vez hasta N procesos.

- Escalabilidad

  Los sistemas distribuidos operan de manera efectiva y eficiente a muchas escalas diferentes. La escala más pequeña consiste en dos estaciones de trabajo y un servidor de ficheros, mientras que un sistema distribuido construido alrededor de una red de área local simple podría contener varios cientos de estaciones de trabajo, varios servidores de ficheros, servidores de impresión y otros servidores de propósito específico.

- Transparencia

  La transparencia se define como la ocultación al usuario y al programador de aplicaciones la separación de los componentes en un sistema distribuido, de manera que el sistema se percibe como un todo, en vez de una colección de componentes independientes. La transparencia ejerce una gran influencia en el diseño del software de sistema.

  Existen ocho tipos de transparencia, las dos más importantes son las transparencias de acceso y de localización; su presencia o ausencia afecta fuertemente a la utilización de los recursos distribuidos. A menudo se las denomina a ambas transparencias de red.

  La característica clave en un sistema distribuido es la transparencia y en una red computacional, es la interoperabilidad, condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos. El interés de la seguridad en los sistemas distribuidos se centra en los problemas que surgen de estos requerimientos, la interoperabilidad y la transparencia.

Los sistemas distribuidos presentan mayor cantidad de brechas de seguridad que cualquier otro sistema, ya que tiene varios puntos vulnerables para ser atacado. El uso de redes de computadoras y de sistemas distribuidos se justifica si las medidas de seguridad pueden ser adecuadamente soportadas por el sistema o la red.

Debido a las características, dimensiones y complejidades inherentes de la supervisión de los procesos industriales, la mayoría de los sistemas SCADA en funcionamiento constituyen sistemas distribuidos.

1.3 SCADA

SCADA proviene de las siglas "Supervisory Control And Data Adquisition", es decir: adquisición de datos, control y supervisión.

Un sistema SCADA incluye hardware para señales de entrada y salida, controladores, interfaz hombre-máquina, redes, comunicaciones, base de datos entre otros softwares.

"SCADA no es una especificación sino una aplicación, cualquier aplicación que de un sistema tome sus datos para el control de este sistema, es un SCADA" *(Berry, 2005).*

1.3.1 Definición

Se trata de una aplicación software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etcétera.) y controlando el proceso de forma automática desde la pantalla del ordenador *(Ferrari, 2005).*

Es un sistema industrial de mediciones y control que consiste en una computadora principal o máster (generalmente llamada Estación Principal, Master Terminal Unit o MTU); una o más unidades de control obteniendo datos de campo (generalmente llamadas estaciones remotas, Remote Terminal Units, o RTU's); y una colección de software estándar y/o a medida usado para monitorear y controlar remotamente dispositivos de campo *(Berry. 2005)*.

En resumen, el SCADA es una aplicación de control de producción que se comunica con los diferentes dispositivos y controla los procesos de forma automática y pueden ser administradas desde el monitor de un computador.

Estas están conformadas por dos elementos imprescindibles: el sistema que controla, ya sea a un sistema hidráulico, una red, un sistema de iluminación, o cualquier otro proceso industrial, y una red de dispositivos inteligentes que constituyen la interfaz del primer sistema mediante sensores y controles de salida. Este segundo elemento vendría conformando el SCADA propiamente.

Como su nombre lo indica sus tres funciones principales son la adquisición, refiriéndose a recolectar, procesar y almacenar la información; la supervisión, para observar la evolución del proceso desde un monitor; y el control de los procesos para modificarlos.

Los sistemas SCADA utilizan las computadoras y tecnologías de comunicación para automatizar y controlar los procesos industriales. Son capaces de recolectar información rápidamente de una gran variedad de fuentes, y presentarla a un operador de forma sencilla de interpretar.

Además el sistema SCADA brinda información a los diferentes usuarios que interactúan con él, supervisores de proceso o mantenimiento, u otros.

1.3.2 ¿Dónde utilizar un SCADA?

Se puede utilizar el SCADA para dirigir cualquier tipo de equipamiento. El software es empleado para automatizar un proceso industrial complejo. En dicho proceso, el control manual es muy poco

práctico y difícil, ya que tiene tantos controles y accesorios que al humano le resultaría difícil de manipular y dirigir.

El SCADA es usado en:

- Administración, generación y distribución de energía eléctrica, donde se utiliza para controlar las altas y bajas del voltaje. Posibilita monitorear los circuitos que intervienen y controlar acciones a realizar con el fluido eléctrico.
- Los sistemas hidráulicos: en la manipulación del fluido y la presión.
- La construcción: proporcionando el entorno adecuado para el control y monitoreo de sistemas.
- Transito: las autoridades usan SCADA para regular la electricidad de los metros y tranvías, para automatizar las señales de tránsito, los semáforos, para localizar pistas, autobuses y trenes y para controlar el paso por puentes levadizos.
- Empresas petrolíferas y de gas: su uso permite controlar de forma eficiente la extracción de petróleo y gas, las mediciones electrónicas de los flujos de gas, el refinamiento del petróleo una ves extraído, de forma general podemos obtener la supervisión, optimización y rendimiento de todas las operaciones que se efectúan en estas empresas.

El sistema SCADA puede ser empleado por todas las industrias y proyectos de infraestructura pública o privada donde la automatización de los procesos incremente la eficiencia.

En todas las industrias se necesita gestionar múltiples factores y la interacción entre ellos. Los sistemas SCADA ofrecen la capacidad y el poder computacional para poder dirigir todas las operaciones de una industria determinada. Ejemplos de estas pudiesen ser:

- La posibilidad de detectar y corregir los problemas tan pronto como empiezan.
- Tomar medidas antes de que aparezca el problema.
- Controla procesos largos y más complejos con un pequeño, y poco especializado, personal.
- La posibilidad de mejorar los conocimientos del personal acerca del sistema.
- Se pueden colocar controles en todos los puntos críticos del proceso dirigido.
- Los operadores pueden ver tantas operaciones como muestre el monitor, y lo mejor, detalladas y en tiempo real.
- Los sistemas SCADA facilitan el trabajo sin importar cuan grande o complejo sea el

proceso industrial. Hace que se pueda hacer mucho más con menos costo, obteniendo un incremento a la rentabilidad de las empresas.

1.3.3 Ejemplos de Software SCADA

Algunos de los programas SCADA, o que incluyen SCADA como parte de ellos, son:

- **Aimax**, de Desin Instruments S.A.
- **CUBE**, Orsi España S.A.
- **FIX**, de Intellution.
- **HMI/SCADA**, de Software Horizons Inc.
- **Monitor Pro,** de Schneider Electric.
- **SCADA InTouch**, de LOGITEK.
- **SYSMAC SCS**, de Omron.
- **WinCC**, de Siemens.

Estos ejemplos de software SCADA son todos privativos que se ejecutan sobre Windows. La mayoría de ellas son aplicadas en la integración de procesos industriales. Constituyen aplicaciones configurables para su despliegue en diferentes entornos. Muchos de los productores que se dedican al desarrollo de SCADAs las diseñan e implementan para situaciones específicas de los clientes que atienden. Son pocos los softwares SCADAs que se han implementado en software libre, como un ejemplo de estos SCADAs podría citarse el SCADA Likindoy que fue desarrollado bajo la política de código abierto (Open Source), el SCADA ADDVANTAGE Pro y otros pocos. Este último soporta ambas plataformas (Linux y Windows), aunque el Linux se vea solamente en las PCs cliente.

En Cuba hasta el momento no es cuantiosa la experiencia que existe en desarrollo de softwares SCADAs. Los existentes hasta hoy han sido de pequeña complejidad y se ejecutan sobre Windows. Como ejemplo tenemos el SCADA Eros, software privativo desarrollado en 1991 por especialistas de la “Unión del níquel” de Nicaro, Holguín, que aun se mantiene activo. Está soportado en ambiente de Windows 95 y utiliza todas las facilidades del sistema, aunque se pronostica que en versiones futuras se ejecute sobre software libre.

Para lograr una comunicación rápida y eficiente entre los módulos de los SCADAs se necesitaba un software que garantizara la conectividad entre todos los sus componentes. Se requería de un software que formase un conjunto de programas distribuidos, sencillos, consistentes e integrados que facilitan las tareas de diseño, programación y gestión de aplicaciones distribuidas. Precisamente para facilitar el desarrollo de los sistemas distribuidos la Object Management Group

(OMG) aboga por el uso de sistemas abiertos, con interfaces estándar orientadas a objetos, construidos con hardware, redes, sistemas operativos y lenguajes de programación heterogéneos.

1.4 Middleware

Debido a la misma crecida de los sistemas distribuidos se hace cada vez más difícil la integración de varios procesos y la comunicación entre ellos. Por esta razón se propone una arquitectura de software para dotar a las plataformas de un conjunto extensible de servicios. Se apoyó la creación de una capa de software intermedio, alojada en el cliente, que controle las comunicaciones entre contenidos y plataforma y que aportara algunas funcionalidades necesarias, un Middleware.

No se puede decir a ciencia cierta si el middleware es imprescindible para el proyecto de una empresa. Esta respuesta depende de varios factores como la complejidad y la dimensión de la aplicación. Si la aplicación es puntual, sin capacidad de crecer, se podría obviar este componente.

Se podrían plantear muchísimos motivos para incorporar el middleware en un proyecto. “El middleware es importante dentro del marco de trabajo para permitir interactuar entre la esfera física de los sensores y el mundo lógico de la información” *(Hostalot, 2005).* El entorno realiza la manipulación de datos en base a estándares ya definidos o estándares de trabajo determinados por el propio usuario. Es esencial que la arquitectura sirva también para poder soportar la estructura actual y futura, y también que permita la comunicación entre los protocolos definidos en la misma.

1.4.1 Definiciones

“Middleware es el software de conexión que consiste en un conjunto de servicios que permiten a múltiples procesos que se ejecutan en una o más máquinas de interactuar a través de una red” *(Bray, 1997).*

El software de Middleware son aquellos componentes que sirven de apoyo y complementan el ambiente para el desarrollo y ejecución de aplicaciones *(Uruguay, 2006).*

El Middleware fue forjado en los primeros días de transición a la arquitectura Clientes/Servidor para hacer referencia al software orientado a las redes que vinculaban los clientes con los servidores. Aunque, según las definiciones, un Middleware es un software entre una aplicación y el sistema operativo o la red del sistema; un software genérico para necesidades particulares. El Middleware es un término que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores.

El middleware es un software intermedio que no pertenece a los dominios del servidor, ni a la

interfaz de usuario, ni a la lógica de la aplicación en los dominios del cliente; tampoco debe confundirse con la red física en sí, el middleware es una interfaz lógica estándar de los servicios de red.

Puede ser implementado a cualquier nivel y no es una simple pieza de software ni un servicio cualquiera que reside en la pantalla del usuario. Técnicamente puede verse como una capa que junta o sirve de mediador entre dos programas separados ya existentes. Puede ser compartido por varias aplicaciones sirviendo varios propósitos en diferentes entornos.

Se puede pensar en el middleware como un adaptador que se utiliza para conectar una nueva impresora con un computador viejo. Este adaptador conecta ambos dispositivos permitiendo la comunicación entre ellos y por tanto sus funcionalidades.

Todo tipo de Middleware tiene un mismo propósito: el permitir a múltiples computadores el hacer múltiples cosas a través de la red, o permitir a una computadora el hacer muchas cosas o algo muy complicado a través de la red.

Para el presente trabajo se definirá como Middleware la capa de red que se sitúa sobre las niveles físico y de transporte y bajo las capas de gestión y aplicación del modelo de referencia de Interconexión de Sistemas Abiertos (Open System Interconnection, OSI), el mismo tendrá como principal objetivo garantizar la comunicación segura entre los módulos que componen el SCADA, Guardián del ALBA.

1.4.2 Tipos de Middlewares

De forma general existen dos tipos de middleware *(Departamento de Control de Calidad y Auditoría Informática, 2001)*:

- *El Middleware general*: es el sustrato de la mayoría de las interacciones de Cliente/Servidor. Incluye las pilas de comunicación, directorios distribuidos, servicios de autenticación, horario de la red, llamadas a procedimientos remotos (RPC's), y servicios en cola. Incluye también las extensiones del sistema operativo de redes, como los servicios distribuidos de archivos e impresión y los productos de Middleware Orientado a Mensajes (MOM).
- *El Middleware de servicios específicos*: es necesario para cumplir un tipo particular de servicio de Cliente/Servidor; existe un middleware específico para los servidores dedicados: middleware para bases de datos, middleware para OLTP (On-Line Transactional Processing – Procesamiento de Transacciones en Línea), middleware para objetos, etcétera. El middleware es una herramienta adecuada, que no sólo es flexible y segura, sino que también protege la inversión en tecnología y permite manejar diferentes ambientes de computación.

Una clasificación más específica de los tipos de middlewares puede ser *(Monge, 2002)*:

- **Middleware base:** Es el conjunto de estándares y servicios relacionados, que sirven de soporte y base para la construcción de middleware más especializado. Los estándares de componentes permiten a los distintos tipos de productos desarrollados bajo estas especificaciones interoperar en plataformas, protocolos de comunicación y lenguajes de programación heterogéneos. Estándares como CORBA de la OMG y COM, COM+ de Microsoft, son considerados middleware base. Permiten el desarrollo de productos como los servidores de aplicaciones.
- **Middleware para procesamiento de transacciones**: Facilita la conectividad y el acceso a un gran número de usuarios con servicios de "back-end" limitados. Es importante asegurar al cliente que sus transacciones se realizan satisfactoriamente, cumpliendo adecuadamente propiedades como: atomicidad, consistencia, aislamiento, y durabilidad. Se requiere un monitor transaccional, para dar soporte a este tipo de "middleware". Con responsabilidades como el balanceo de la carga entre las peticiones del cliente, servicios de multihilo y administración de memoria. Algunos productos que proporcionan middleware para el procesamiento de transacciones son BEA's Tuxedo, Transarc's Encina y CICS de IBM.
- **Middleware de comunicación:** Un middleware de comunicación, tiene como función proporcionar un medio de comunicaciones para que las aplicaciones puedan establecer dialogo entre sí. Soportan diferentes tipos de protocolos: http (Hipertext Transfer Protocol). IIOP, protocolo para comunicar a objetos de CORBA. DCOM, protocolo para componentes ActiveX de Microsoft.
- **Middleware de base de datos:** Crea una capa transparente en el acceso a bases de datos, escondiendo toda la complejidad dada por la gran diversidad de servidores de bases de datos existentes.
- **Middleware de aplicación:** Un middleware de aplicación ofrece servicios como la ejecución, extensión, e integración de otras aplicaciones. La ejecución se da mediante protocolos como CGI (Common Gateway Interface), que permite al cliente invocar un programa del servidor mediante HTTP, el servidor ejecuta el programa y devuelve la salida del mismo al cliente. Para alcanzar la extensión de aplicaciones, se usan los API's propietarios. Por ejemplo para extender las funciones de un servidor de web; son

extensamente usados NSAPI de Netscape Server e ISAPI de Internet Server. Los servidores de aplicaciones, agrupan en un ambiente integrado: sistemas distribuidos, servicios de monitoreo transaccional, bases de datos, servicios de comunicaciones, soporte a CORBA y otros estándares. Algunos productos existentes en el mercado son WebLogic de BEA, Web Sphere de IBM ó JasmineII de Computer Associates.

También se puede ver otras clasificaciones de los middleware en función de su escalabilidad y su tolerancia a fallos. Puede existir otros tipos de clasificaciones *(Bray, 1997):*

- Remote Procedure Call (RPCs) — El cliente realiza una llamada a procedimientos que están corriendo en máquinas remotas. Pueden ser síncronos o asíncronos.
- Publish/subscribe — Este tipo de monitores middleware activan y entregan información relevante para los subscriptores.
- Message Oriented Middleware (MOM) — Los mensajes enviados al cliente se recogen y se almacenan hasta que son solicitados, mientras el cliente continúa con otros procesos.
- Object Request Broker (ORB) — Este tipo de middleware permite que los clientes envíen objetos y soliciten servicios en un sistema orientado a objetos.
- SQL-oriented Data Access — middleware entre las aplicaciones y los servidores de base de datos.

Seguidamente se muestra una breve descripción de algunos de ellos.

RPC (Llamada a Procedimiento Remoto): Es un protocolo que permite a un programa de ordenador ejecutar código en una máquina remota sin tener que preocuparse por las comunicaciones entre ambos. Las llamadas pueden ser síncronas o asíncronas. Permiten acceder a un servicio, una función en una base de datos o sistema operativo sobre una máquina remota. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tendría que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro del paradigma cliente/servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación al cliente. La idea es que la sintaxis del cliente y el servidor permanezcan igual como si estuvieran en la misma máquina.

Hay distintos tipos de RPC, muchos de ellos estandarizados. La mayoría de ellos utilizan un lenguaje de descripción de interfaz (IDL) tanto del lado del cliente como del lado del servidor que define los métodos exportados por el servidor.

Una de las importancias que posee es que manipula distintos formatos de datos. Básicamente

permite a objetos distribuidos interactuar entre sí de manera transparente, es decir, como si estuviesen en la misma máquina.

ORB (Object Request Broker - Broker de objetos distribuidos)

ORB, es un middleware del tipo base, que sirve de soporte y base para la construcción de middleware más especializado. Es un componente de software que se encarga de establecer comunicación entre los clientes y los objetos de forma transparente con respecto a la distribución. Proporciona una serie de facilidades, como la de localizar un objeto remoto dada una referencia a ese objeto, y otra es la ordenación de los parámetros y valores de retorno y de invocaciones a métodos remotos.

Desde el punto de vista del ORB, cuando en un segmento de código de una aplicación se quiere usar un servicio proporcionado por un objeto, se obtiene una referencia del objeto que provee ese servicio. Después de obtener la referencia, se puede invocar métodos en ese objeto. Una de las más importantes responsabilidades del ORB es precisamente resolver peticiones de referencias a objetos, permitiendo que se establezca comunicación entre ellos. Una vez obtenida la referencia al objeto, el siguiente paso es la ejecución del servicio, mediante el llamado a un método, el que generalmente necesita parámetros de entrada y devuelve resultados. El ORB recibe los parámetros de entrada y los traduce a un formato (on-the-wire) para transmitirlos mediante la red hasta el objeto remoto de destino, este mecanismo es conocido como “marshaling”. Haciendo uso del mecanismo llamado “unmarshaling”, se convierten los parámetros al formato que el receptor entienda. El programador utiliza el método remoto como si fuese local, esto se hace de forma transparente.

Este middleware implementa el mecanismo de direccionamiento de objetos. Dan aspecto de objetos a los proceso conectados a él. Permite conectar objetos de servicios de lenguajes muy diversos (Fortran, Cobol, C) *(Mortera, 2002).*

TAO es un ORB para tiempo real, desarrollado en la Universidad de Washington por Douglas Schmidt y su equipo. Es uno los ORB con mejores prestaciones. Permite la creación de servidores multihilos y tiene servicios para tiempo real. (*Monge, 2002*).

Publish/Subscribe (Publicador/Subscriptor)

El objetivo de dicho middleware es mantener desacoplado al publicador de un mensaje con el consumidor de este mensaje. El destino de un mensaje es un tópico, no un consumidor en particular. Se utiliza cuando un sistema quiere mandar un mensaje a todos los sistemas que están interesados en que les llegue dicho mensaje. Los mensajes generados por el publicador no están asociados a un destino en particular sino a un tópico. Los subscriptores pueden estar subscritos a uno o más tópicos.

En particular, este middleware activa y entrega información relevante a los subscriptores. Es muy útil cuando un grupo de aplicaciones quieren notificar a cada uno de las otras aplicaciones una ocurrencia en particular.

MOM (Message-Oriented Middleware - Middleware orientado a Mensaje)

Es una infraestructura Cliente/Servidor que aumenta la interoperabilidad, portabilidad y flexibilidad de una aplicación permitiendo a la aplicación ser distribuida a través de múltiples plataformas heterogéneas. Reduce la complejidad de desarrollo de aplicaciones que abarcan múltiples sistemas operativos y protocolos de red aislando a los desarrolladores de aplicaciones de los detalles de los distintos sistemas operativos e interfaces de red. Pueden verse también como aplicaciones que se extienden a través de diversas plataformas y redes.

MOM es un software que reside en ambas porciones de la arquitectura Cliente/Servidor y generalmente apoya las llamadas asíncronas entre el cliente y el servidor de aplicaciones. Las colas de mensajes proveen almacenamiento temporal cuando el programa destino está ocupado o desconectado. La mayoría de los MOM dependen de una cola de mensajes del sistema, pero hay aplicaciones que dependen de sistemas de mensajería broadcast o multicast.

Este tipo de middleware es muy apropiado para la integración funcional por consistencia de datos y procesamiento multipaso. Provee mecanismos para crear, manipular, almacenar y comunicar los mensajes, que son una combinación de datos e información de control *(Palacio, 2007).*

Sus mensajes son muy útiles para la actualización de información y la sincronización y coordinación de procesos.

1.4.3 Apoyo de CORBA al concepto de middleware.

CORBA apoya el concepto de middleware, mediante ORBs, que establecen relaciones cliente/servidor entre objetos. Los estándares de componentes permiten a los distintos tipos de productos desarrollados bajo estas especificaciones interoperar en plataformas, protocolos de comunicación y lenguajes de programación heterogéneos.

1.4.3.1 CORBA

Esta especificación CORBA es una arquitectura de objetos distribuidos que hace posible que los objetos interactúen a través de redes de computadoras utilizando plataformas heterogéneas, sistemas de comunicaciones heterogéneas y diferentes lenguajes de programación *(Castillo, 1998)*.

Los elementos más importantes de la arquitectura de interoperatividad CORBA son el Lenguaje de Definición de Interfaces (Interface Definition Language, IDL), las traducciones de IDL a lenguajes de programación como Java, C++, etcétera y una infraestructura de distribución de

objetos Object Request Broker (ORB).

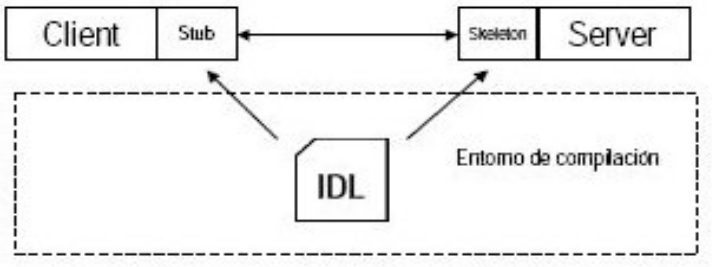

Fig.1.1 Esquema de interacción entre diferentes interfaces a través del IDL.

CORBA enfatiza en la transparencia y flexibilidad, el cliente no se entera de la ubicación del objeto servidor, y mediante IDL logra que las interfaces puedan ser implementadas en más de 30 lenguajes de programación. CORBA establece estándares para la comunicación de objetos a través de procedimientos/métodos remotos. En las plataformas heterogéneas las arquitecturas que la conforman pueden ser sistemas MacOS, OS/2, Microsoft Windows, Unix y Linux de diferentes fabricantes. Dentro de la heterogeneidad también se incluyen los sistemas de comunicaciones, protocolos de comunicación como TCP/IP. Se pueden crear objetos en lenguajes de programación diferentes como Java y C++, en plataformas distintas. Estos objetos podrán interactuar entre ellos sin ningún tipo de problema.

CORBA surge como resultado del consenso entre los miembros de la OMG. Define la infraestructura para la arquitectura Object Management Arquitecture (OMA) de OMG, especificando los estándares necesarios para la invocación de métodos sobre objetos en entornos heterogéneos.

CORBA es una especificación normativa que esta basada en cinco aspectos de los sistemas de objetos distribuidos *(Castillo, 1998)*:

1. Lenguaje de definición de interfaces IDL (Interface Definition Language), traducciones de IDL a lenguajes de programación como Java, C++, etcétera y una infraestructura de distribución de objetos ORB (Object Request Broker).
2. Servicios CORBA (CorbaServices), que complementan a los objetos que sirven para la construcción de aplicaciones. La cantidad de servicios crece permanentemente con el propósito de añadir nuevas capacidades a los sistemas desarrollados con CORBA. Estos servicios cubren aspectos como el de eventos, nombrado, de persistencia, de transacciones, etcétera.
3. Facilidades (CorbaFacilities), estas especificaciones cubren servicios de alto nivel, como los interfaces de usuario, los documentos compuestos, la administración de sistemas y

redes, etcétera.

4. Interfaces de Dominio (CorbaDomains), que proveen funcionalidad para usuarios finales en áreas de interés particular, como la medicina, contabilidad, etcétera.
5. Protocolo General Inter-ORB GIOP (General Inter-ORB Protocol), que define los mensajes y el empaquetado de los datos que se transmiten entre los objetos. Define implementaciones, de ese protocolo, sobre diferentes protocolos de transporte.

Ventajas de CORBA

CORBA organiza los servicios a través de los interfaces IDL. Logra independencia de la plataforma y del lenguaje de desarrollo, permitiendo el desarrollo en entornos heterogéneos. Informa a clientes y servidores de caídas de los canales de comunicación. Integra software heredado definiendo las interfaces IDL para ponerlo disponible en CORBA. Brinda servicios adicionales para encontrar objetos y servicios dentro de entornos distribuidos, como la definición de entornos de trabajo para diferentes disciplinas.

Servicios de apoyo a CORBA

Para proporcionar la potencia y flexibilidad existen los servicios de apoyo a CORBA. En la actualidad sólo algunos están disponibles en las implementaciones de CORBA*(Monge, 2002):*

- Servicio de nombres. Relación entre un objeto y su nombre.
- Seguridad en las comunicaciones.
- Transacciones.
- Gestión de eventos. Mensajes asíncronos entre los objetos.
- Persistencia. Maneja el almacenamiento de objetos.
- Control del ciclo de vida de los objetos. Creación, copia, traslado y destrucción de objetos.
- Control de concurrencia. Múltiples accesos a objetos.
- Externalización. Protocolos y convenciones para externalizar objetos.
- Relaciones. Creación y mantenimiento de relaciones entre objetos.
- Peticiones: Consultas de datos.
- Licencias: Control de permisos y políticas de seguridad.
- Propiedades.
- Servicio de tiempo.
- Comercio: Búsqueda dinámica de objetos.
- Recolección de objetos.

1.4.4 Funcionalidades claves del Middleware.

La principal función del middleware sería el de mediador entre las partes de una aplicación o entre

aplicaciones. El tema de la arquitectura juega un papel central en el diseño del middleware. La arquitectura esta relacionada con la organización, la estructura y la comunicación entre las partes.

El Middleware esta capacitado para proporcionar *(Departamento de Control de Calidad y Auditoría Informática, 2001)*:

- Independencia entre el cliente y el servidor. No necesitan saber comunicarse entre ellos, sino cómo comunicarse con el módulo de middleware.
- Traducción de la información de una aplicación y el paso de dicha información a otra: acepta consultas y datos recuperándolos de la aplicación cliente, los transmite y envía la respuesta de regreso. También genera los códigos de error.
- Control de las comunicaciones: da a la red las características adecuadas de desempeño, confiabilidad, transparencia y administración.

El middleware tiene varias funcionalidades, pero las claves de forma genérica serían *(Hostalot, 2005):*

- **Gestión de dispositivos**: Los middleware pueden en su mayoría controlar cualquier tipo de hardware, desde lectores a dispositivos actuales (entrada/salida). Puede controlar el funcionamiento y actualización de los dispositivos y avisar en caso de presentarse algún problema.
- **Procesamiento de datos**: Funciona como filtro de los datos recolectados para evitar lecturas múltiples y evitar sobrecargo de los sistemas. Puede configurar alertas y añadir información antes de pasarlas a los sistemas de gestión empresarial.
- **Conectar la información con las aplicaciones de negocio:** Los middlewares tienen herramientas que incluyen Programas de Interfase de Aplicación (APIs) utilizadas por muchas empresas como puentes para conectar los datos con su software de negocios.

También existen otras funcionalidades un poco más generales pero no menos importantes como:

- Distribución oculta: El hecho de que una aplicación es usualmente desarrollado de muchas partes interconectadas que corren en lugares distribuidos.
- Oculta la heterogeneidad de los componentes hardware, sistemas operativos y protocolos de comunicación.
- Suministrar una serie de servicios comunes para efectuar funciones de propósito general, para evitar esfuerzos repetidos y para facilitar la colaboración entre aplicaciones.

La complejidad del Middleware no esta en el “qué” hace sino “cómo”, ante todas las posibles situaciones de la computación distribuida. Las empresas que usan middleware, direccionan este

producto según las particularidades o énfasis que sus aplicaciones tienen. Es decir, existen productos capaces de llegar a gestionar con los lectores tan bien que pueden controlar la interferencia en la comunicación que pueda llegar a darse. Y así también pueden hallarse algunos que pueden llegar a integrarse completamente con cualquier programa software comercial de forma transparente.

Las funcionalidades fundamentales del middleware se resumen en: concertar cómo los módulos de una aplicación se interrelacionan y funcionan en coordinación haciendo posible, superar las dificultades de comunicación que existen entre las distintas partes del sistema. Además de contribuir a la creación de los sistemas distribuidos brindando velocidad y robustez al proceso al ofrecer servicios configurables y posibilidad de integración con componentes desarrollados por distintos proveedores de tecnologías.

Procura la integración de dispositivos computacionales geográficamente independientes en función de obtener un sistema único. Por otra parte, para desarrollar sistemas basados en componentes, utiliza programación orientada a objetos y la encapsulación de patrones de interacción exitosos, en frameworks reusables para obtener una reducción en las complejidades del software.

El Middleware esta estrechamente relacionado con el tipo de aplicación con el que se quiere integrar la tecnología. Las aplicaciones básicas necesitan un Middleware poco sofisticado que implemente solo las funcionalidades indispensables requeridas. Incluso se podría obviar el Middleware si la aplicación es muy simple. Sin embargo si se habla de aplicaciones complejas, la introducción de esta capa intermedia o middleware, es indispensable.

1.4.5 Servicios

Aporta una serie más funcional de APIs que los sistemas operativos y los servicios de red, que le permiten a una aplicación a *(Bray ,1997):*

- Ubicarse de forma invisible en la red, permitiendo la interacción con otras aplicaciones o servicios.
- Ser independiente de los servicios de red.
- Ser fiable y disponible.
- Capacidad escalable sin perder funciones.

El objetivo principal de los servicios de middleware es ayudar a resolver muchos problemas de solicitud de conexión y de interoperabilidad *(Bray ,1997).*

La gran cantidad de servicios con las que cuenta el middleware es un obstáculo para su uso. Para

mantener su entorno de trabajo con fácil manejabilidad, los desarrolladores deben escoger un pequeño número de servicios que satisfagan sus necesidades de funcionalidad y cobertura de la plataforma.

Existen varios tipos de servicios *(Gall, 2002)*:

- **Servicios de sistemas distribuidos.**
  - Comunicación critica, programa a programa y administración de datos.
  - Este tipo de servicios incluye RPCs, MOMs y ORBs.
- **Servicios de aplicaciones disponibles.**
  - Acceso a servicios distribuidos y las redes subyacentes.
  - Este tipo de servicios incluye transacción de monitores procesados y servicios de base de datos como Lenguaje de Estructura de Consultas (Structured Query Language, SQL).
- **Servicios de administración Middleware.**
  - Qué aplicación y funciones de sistemas deben ser constantemente monitoreada para asegurarse de una funcionalidad óptima del entorno.

1.4.6 Aplicaciones del Middleware

"El Middleware es el método más común para generar datos para códigos de barra desde aplicaciones Oracle" *(Zebra technologies, 2004)*, además de que puede tomar la forma de un software de diseño de etiquetas, aplicaciones de servidor de impresión, o un software de gestión de documentos.

Permiten la sincronización entre sistemas de gestión utilizados para una empresa y las bases de datos en Internet, Intranet y/o Extranet, tanto para comunicar proceso de negocio que proviene del Internet, como para el uso compartido de información de una empresa.

Facilita la conectividad y el acceso a un largo número de usuarios con servicios de back-end limitados. Debe, por tanto, asegurar al cliente que sus transacciones se realizan adecuadamente.

Proporcionan el medio de comunicación para que las aplicaciones puedan conversar entre sí. Pueden estar basados en distintos tipos de protocolos. Enmascara las complejidades de acceso a la base de datos, escondiendo los detalles de implementación (SQL nativo).

Permite el arranque, extensión e integración de otras aplicaciones. Para el arranque de aplicaciones se tiene, en su forma más tradicional: Interfaz de entrada común (Common Gateway Interface, CGI) un protocolo que permite al cliente invocar un programa del servidor, mediante

protocolo de transferencia de hipertexto (HyperText Transfer Protocol,http), y al servidor ejecutar una aplicación propia y devolver su salida al cliente.

La extensión de aplicaciones se logra, en muchas ocasiones, mediante API's propietarios, los cuales permiten extender las funciones de un servidor de Web. Por último, permiten agrupar en un ambiente integrado, los servicios de monitoreo transaccional; bases de datos; servicios de comunicaciones; así como soporte a una CORBA y otros estándares; sistemas distribuidos; y a sistemas legados.

Existen Middlewares, como el caso de Servicio de Distribución de Datos (Data Distribution Service, DDS), que las aplicaciones para sus comunicaciones están enteramente desacopladas. En particular, las aplicaciones no necesitan información sobre el resto de aplicaciones participantes, incluyendo su existencia o localización. ¨El software es capaz de gestionar automáticamente todos los aspectos de la entrega de mensajes, incluyendo determinar quién debe recibir los mensajes, dónde están localizados los destinatarios, y qué ocurre si los mensajes no pueden ser entregados *(eProsima, 2007.)*

Esto es posible porque permite al usuario especificar parámetros de Calidad de Servicio (Quality of Service, QoS) para configurar los mecanismos de descubrimiento automático y de especificar el comportamiento deseado en el envío y recepción de mensajes. Estos mecanismos se configuran inicialmente y no requieren un esfuerzo posterior por parte del usuario. De esta forma, intercambiando mensajes de una forma completamente anónima, el middleware simplifica enormemente el diseño de aplicaciones distribuidas y conduce a programas modulares y bien estructurados.

Mediante la utilización de este software de red, los clientes pueden integrar rápidamente aplicaciones con distintas fuentes de información y ampliar los procesos que se centran en el cliente, en toda la empresa y más allá de ella, a la vez que se reducen los riesgos relacionados con el cumplimiento y la seguridad. Además, esto mejora la posibilidad de las empresas de adaptar procesos y servicios de negocios para satisfacer los requisitos de los clientes que cambian constantemente. Al unir las aplicaciones con el Middleware, las empresas podrán obtener una visión de sus clientes, delimitarlos y atenderlos mejor.

1.4.7 Middleware para el SCADA

En los sistemas SCADA como en todo sistema distribuido se comparten recursos y la lógica de los sistemas de control y supervisión se encuentra dispersa entre sus distintos componentes.

La comunicación desempeña un rol crucial entre los diversos módulos del sistema. La abstracción de dicha comunicación posee distintos enfoques. El enfoque del middleware es uno de ellas. El objetivo es introducir una capa de software intermedia entre la aplicación y la red que proporcione

una abstracción de las comunicaciones y la transferencia de datos. Se hace del middleware un conector de dos aplicaciones literalmente independientes y que brinda la abstracción necesaria para facilitar la programación y enmascaramiento de la heterogeneidad de las redes, el hardware y el sistema operativo.

Entre los mayores desafíos del software de comunicación en un sistema SCADA se encuentran la tolerancia a fallas y la realización de operaciones en tiempo real. Estas propiedades no dependen únicamente del middleware, sino también del sistema en su totalidad, tanto de las plataformas software como hardware.

A continuación se realiza una breve descripción del funcionamiento del Guardián del ALBA en general y del papel que juega el middleware en el SCADA. Inicialmente se recolectan los datos que envían los dispositivos de campo (PLC) y se van guardando en Base de Datos de Tiempo Real (RTDB) para mantener actualizada constantemente la información del sistema, este módulo RTDB se comunica con el Middleware a través de interfaces de comunicación que este brinda. Los restantes módulos Base Datos Histórico (HDB), Seguridad, Configuración y Interfaz Hombre-Maquina (HMI) se comunican de igual forma con el Middleware para intercambiar información entre todos los módulos que componen el SCADA, de modo que toda la comunicación se hace por medio del Middleware.

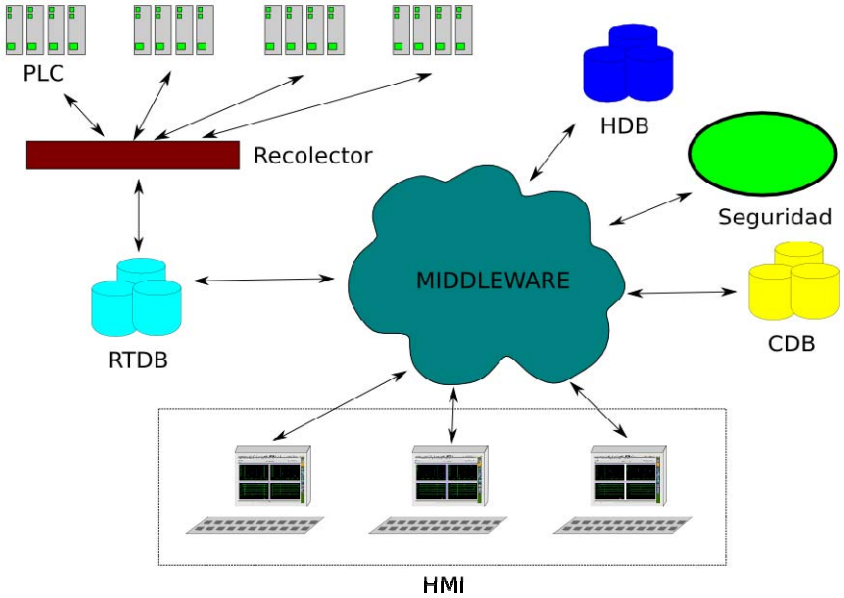

Fig.1.2 Middleware del Guardián del ALBA.

1.5 Seguridad

Cuando se cita la palabra seguridad viene a la mente de algunos la imagen de una armada, o espías en ambientes de misterio. Otros lo comparan con una organización de seguridad nacional. Pero estas personas están muy lejos de saber el verdadero significado de seguridad en el mundo de la información tecnológica.

La seguridad de la información en las tecnologías intenta resguardar las acciones de las empresas y asegurar el intercambio de moneda, de información y evitar los accesos no autorizados. Se sabe que nada es absoluto pero se puede prevenir un desastre o al menos minimizar las pérdidas de cualquier empresa para poder restaurar los negocios a tiempo.

Muchos piensan en la seguridad como un proceso que protege de cualquier acción los datos. Sin embargo la seguridad es igual de importante en otras acciones como el permitir el acceso a los datos a cierto personal, y el compartir datos de forma abierta o no entre otras.

1.5.1 ¿Por qué seguridad?

Las empresas son cada vez más dependientes de su sistema de información para sus negocios. El compromiso del sistema en términos de pérdidas, inexactitud de la información o el acceso de competidores puede ser extremadamente costoso para la empresa.

Muchas organizaciones ofrecen servicios seguros mediante sus sistemas de comunicación, la efectividad de tales servicios requiere el acceso a recursos críticos del sistema de información de la empresa (archivos, dispositivos de almacenamiento, líneas telefónicas). Dichos recursos deben ser protegidos contra el uso indiscriminado y malicioso por parte de usuarios no deseados.

Si un sistema de comunicación es vulnerable a estos tipos de ataques, el riesgo de pérdida de datos es importante. Este riesgo potencial de seguridad aumenta junto con el nivel de dependencia en tecnología de información, lo que requiere el uso de sistemas de seguridades más confiables y robustas.

Las redes de comunicación son riesgosas por tres razones:

- Existen muchos puntos vulnerables desde donde puede ser lanzado un ataque;
- El perímetro físico del sistema de comunicación se ha extendido, existiendo mensajes de entrada y salida, manteniendo contacto con todos los otros sistemas conectados a la red;
- Las redes ofrecen múltiples servicios de conexión, cada uno con un punto de acceso propio. Cada uno de estos requiere una protección adecuada contra intrusos y cada una ofrece una complejidad y dificultad propia.

En el mundo actual existe la necesidad apremiante de tomar medidas de seguridad para garantizar la privacidad, integridad y disponibilidad de recursos en los sistemas distribuidos. Los ataques contra la seguridad toman las formas de escuchas, suplantación, modificación y denegación de servicio. Los diseñadores de sistemas distribuidos seguros deben tratar con interfaces de servicio desprotegido y redes inseguras en un entorno donde se supone que los atacantes tienen conocimiento sobre los algoritmos empleados para desplegar los recursos computacionales.

Las brechas en la seguridad, son cada vez más frecuentes y variadas. Esto se debe a veces al indebido abuso del sistema, haciendo que los usuarios accidentalmente ganen accesos no autorizados a la información. Aunque a veces existen ataques maliciosos contra la información almacenada.

1.5.2 Amenazas deliberadas a la seguridad de la información

"Se entiende por amenaza una condición del entorno del sistema de información (persona, máquina, suceso o idea) que, dada una oportunidad, podría dar lugar a que se produjese una violación de la seguridad (confidencialidad, integridad, disponibilidad o uso legítimo)" *(Marañon, 2000).*

¿Las políticas de seguridad y el análisis de riesgos identifican las amenazas que han de ser contrarrestadas?, depende del diseñador del sistema de seguridad especificar los servicios y mecanismos necesarios para contrarrestar dichas amenazas o ataques mal intencionados.

Las cuatro categorías generales de amenazas o ataques son las siguientes:

- **Interrupción**: un recurso del sistema es destruido o se vuelve no disponible. Este es un ataque contra la disponibilidad.
- **Intercepción**: una entidad no autorizada consigue acceso a un recurso. Este es un ataque contra la confidencialidad. La entidad no autorizada podría ser una persona, un programa o un ordenador.
- **Modificación**: una entidad no autorizada no sólo consigue acceder a un recurso, sino que es capaz de manipularlo. Este es un ataque contra la integridad.
- **Fabricación**: una entidad no autorizada inserta objetos falsificados en el sistema. Este es un ataque contra la autenticidad.

Estos ataques se pueden asimismo clasificar de forma útil en términos de:

- **Ataques pasivos**: Son muy difíciles de detectar, ya que no provocan ninguna alteración de los datos.

- **Ataques activos:** Aquellos donde el atacante no altera la comunicación, sino que únicamente la escucha o monitoriza, para obtener información que está siendo transmitida.

Se presentan a continuación una lista con los tipos de ataques que más se realizan en la actualidad, explicando brevemente en qué consiste cada uno de ellos y qué debilidades son las que aprovecha.

- **Sniffing:** este ataque consiste en escuchar los datos que atraviesan la red, sin interferir con la conexión a la que corresponden. Se utiliza principalmente para obtener contraseñas, y en algunos casos para obtener información confidencial.
- **Spoofing:** es el nombre que se le da a los intentos del atacante por ganar el acceso a un sistema haciéndose pasar por otro que dispone de los privilegios suficientes para realizar la conexión.

El ataque que más se suele utilizar sobre conexiones de Protocolo de Control de Transmisiones (Transmission Control Protocol, TCP) es el conocido como adivinación del número de secuencia. “Se basa en la idea de que si un atacante puede predecir el número inicial de secuencia de la conexión TCP generado por la máquina destino, entonces el atacante puede adoptar la identidad de máquina "confiada" ” *(Colectivo de autores, 1997).*

- **Hijacking:** consiste en robar una conexión después de que el usuario ha superado con éxito el proceso de identificación ante el sistema. El ordenador desde el que se lanza el ataque ha de estar en alguna de las dos redes extremo de la conexión, o al menos en la ruta entre ambas.
- **Explotar bugs del software:** aprovechan errores del software. A la mayor parte del software se le ha añadido la seguridad demasiado tarde, cuando ya no era posible rediseñarlo todo. “Además, muchos programas corren con demasiados privilegios, lo que les convierte en objetivo de los hackers, que únicamente han de hacerse con una copia del software a explotar y someterlo a una batería de pruebas para detectar alguna debilidad que puedan aprovechar.” *(Colectivo de autores, 1997).*
- **Confianza transitiva:** en sistemas Unix existen los conceptos de confianza entre hosts y entre usuarios. “Se dice que un usuario sobre un sistema es confiado para otro sistema cuando ese usuario, desde el primer sistema, puede establecer un conexión al segundo sin necesidad de dar una contraseña” *(Colectivo de autores, 1997).* Así, cualquier atacante que tome el control de una máquina, probablemente podrá conectarse a otras gracias a la confianza entre las estaciones de trabajo y/o usuarios.

- **Ataques dirigidos por datos:** son ataques que tienen lugar en modo diferido, sin la participación activa por parte del atacante en el momento en el que se producen. El atacante se limita a hacer llegar a la víctima una serie de datos que al ser interpretados ejecutarán el ataque propiamente dicho.

- **Denegación de servicios:** estos ataques no buscan ninguna información contenida en las máquinas atacadas ni conseguir acceso a ellas. Únicamente van encaminados a impedir que sus usuarios legítimos puedan usarlas. El caso más típico es el mail bombing: envío de cantidades ingentes de correo a la máquina atacada hasta saturarla.
  Es casi imposible evitar todos los ataques de denegación de servicio, aunque lo más importante es configurar los servicios para que si uno de ellos es inundado, el resto permanezca funcionando mientras se encuentra y soluciona el problema.

- **Código móvil:** puede incluir código que accede o modifica los recursos que están legítimamente disponibles para los procesos del host pero no para el creador del código.

- **Enrutamiento fuente:** los paquetes del Protocolo de Internet (Internet protocol, IP) admiten opcionalmente el enrutamiento fuente, con el que la persona que inicia la conexión TCP puede especificar una ruta explícita hacia él. La máquina destino debe usar la inversa de esa ruta como ruta de retorno, tenga o no sentido, lo que significa que un atacante puede hacerse pasar por cualquier máquina en la que el destino confíe (obligando a que la ruta hacia la máquina real pase por la del atacante). Dado que el enrutamiento fuente es raramente usado, la forma más fácil de defenderse contra esto es deshabilitarlo en el Router.

- **ICMP Redirect and Destination unreachable:** muchos mensajes sobre el Protocolo de Mensajes de Control de Internet (Internet Control Message Protocol, ICMP) recibidos en una estación de trabajo son específicos a una conexión particular o son disparados por un paquete enviado por esa estación de trabajo. La intención es limitar el alcance de los cambios dictados por ICMP.

  Desafortunadamente las viejas implementaciones de ICMP no usan esta información extra, y cuando llega uno de esos mensajes, todas las conexiones entre el par de hosts que intervienen en la conexión que propició el mensaje se ven afectadas. Además, con la opción *redirect*, alguien puede alterar la ruta a un destino para que las conexiones en las que esté interesado pasen por su máquina, de forma que pueda intervenirlas. Los mensajes redirect deben obedecerlos sólo los hosts, no los Routers, y sólo cuando estos provengan de un Router de una red directamente conectada.

- **Tempest**: el barrido de los electrones por las pantallas de los ordenadores emana unas señales que pueden captarse incluso a varios kilómetros de distancia. La tecnología tempest es capaz de reconstruir, a partir de las señales captadas, la imagen mostrada en la pantalla que las provocó. Esta tecnología es todavía excesivamente cara, de modo que de momento no es problema a tener en cuenta.

1.5.3 ¿Qué es seguridad?

La protección de sistemas de información contra acceso no autorizado a modificar la información, ya sea en almacenamiento, en proceso, o en transito, y contra la denegación de servicios a usuarios autorizados o la posibilidad de proporcionar servicios a usuarios no autorizados, incluyendo medidas necesarias para detectar, documentar y contar cada una de las trazas creadas. *(Huerta, 2002).*

Un sistema seguro se puede entender como una característica de cualquier sistema que nos indique que está libre de todo peligro, daño o riesgo, y que es, en cierta manera, infalible. La seguridad es omnipresente, afectando a muchos componentes del sistema, incluyendo algunos que no están directamente relacionados con el mismo.

Cuando se habla de seguridad en una red, la idea es proteger a los sistemas de ataques o intrusos que intenten tomar provecho de los recursos disponibles a través de la red, y no de impedir que atraviesen los puntos de acceso; éste es el medio por el cual se logra el objetivo final (comunicarse).

Los valores de una empresa u organización deben ser protegidos contra las amenazas que se perciban. El importe de protección que la institución este dispuesta a pagar depende del valor de las acciones a asegurar y del tipo de ataque que les acontece.

Todos los dispositivos de una red son capaces de soportar e implementar los servicios de seguridad necesarios para proveer protección a los sistemas (inclusive ellos mismos). El lugar (dispositivo) en la red donde se instalen los distintos servicios de seguridad determina la flexibilidad y granularidad de la protección. Un ataque puede provenir desde cualquier sitio de la red, tanto interna como externa por lo que los sistemas finales deberían protegerse por sí mismos de los intrusos.

Los tres elementos principales a proteger en cualquier sistema informático son el software (programas lógicos que hacen funcional al hardware, tanto sistemas operativos como aplicaciones), el hardware (elementos físicos de un sistema informático, como CPUs, terminales, cableado, medios de almacenamiento secundario como CD-ROMs y los datos (conjunto de información lógica que manejan el software y el hardware). Habitualmente los datos constituyen el principal elemento de los tres a proteger, ya que es el más amenazado y seguramente el más

difícil de recuperar.

1.5.4 Servicios de seguridad

Para hacer frente a las amenazas a la seguridad del sistema se definen una serie de servicios para proteger los sistemas de proceso de datos y de transferencia de información de una organización. Estos servicios hacen uso de uno o varios mecanismos de seguridad.

Una clasificación útil de los servicios de seguridad es la siguiente *(Marañon, 2000)*:

- **Confidencialidad**: requiere que la información sea accesible únicamente por las entidades autorizadas.
- **Autenticación**: requiere una identificación correcta del origen del mensaje, asegurando que la entidad no es falsa.
- **Integridad**: requiere que la información sólo pueda ser modificada por las entidades autorizadas.
- **No repudio**: ofrece protección a un usuario frente a otro que niegue posteriormente que en realidad se realizó cierta comunicación.
- **Control de acceso:** requiere que el acceso a los recursos (información, capacidad de cálculo, nodos de comunicaciones, entidades físicas, etcétera.) sea controlado y limitado por el sistema destino frente a usos no autorizados o manipulación.
- **Disponibilidad**: requiere que los recursos del sistema informático estén disponibles a las entidades autorizadas cuando los necesiten

No existe un único mecanismo capaz de proveer todos los servicios anteriormente citados, pero la mayoría de ellos hacen uso de técnicas criptográficas basadas en el cifrado de la información.

1.5.5 Criptografía: Una breve introducción

“La criptografía se define como la rama inicial de las Matemáticas y en la actualidad también de la Informática y la Telemática, que hace uso de métodos y técnicas con el objeto principal de cifrar, y por tanto proteger, un mensaje o archivo por medio de un algoritmo, usando una o más claves” *(Aguirre, 2006).*

Esto dará lugar a diferentes tipos de sistemas de cifra, denominados criptosistemas, que permiten asegurar al menos tres de los cuatro aspectos básicos de la seguridad informática: la confidencialidad o secreto del mensaje, la integridad del mensaje y autenticidad del emisor, así como el no repudio mutuo entre emisor (cliente) y receptor (servidor).

En la Jerga de la criptografía, la información original que debe protegerse se denomina texto en claro. El cifrado es el proceso de convertir el texto plano en un galimatías ilegible, denominado texto cifrado o criptograma. Por lo general, la aplicación concreta del algoritmo de cifrado se basa en la existencia de una clave: información secreta que adapta el algoritmo de cifrado para cada uso distinto *(Gómez, 2007)*.

Fig 1.3: Modelo de Criptografía *(Ferney, 2007).*

Las dos técnicas más sencillas de cifrado, en la criptografía clásica, son la sustitución técnica que supone el cambio de significado de los elementos básicos del mensaje, letras, dígitos o los símbolos, y la transposición, que supone una reordenación de los mismos; la gran mayoría de las cifras clásicas son combinaciones de estas dos operaciones básicas.

1.5.5.1 Sistemas de cifrado

- *Simétrico*

  Los sistemas de cifrado simétrico o de clave privada, son aquellos que utilizan la misma clave para cifrar y descifrar un documento. Uno de los algoritmos de encriptación de llave privada más conocido es el DES (Data Encryption Standard).

  Aunque este proceso de encriptación es muy útil, tiene limitaciones. Debido a que este proceso utiliza la misma llave para los procesos de encriptación y desencriptación, si las partes involucradas en la transmisión de datos seguros no han tenido contacto previo es un problema acordar la clave. Por tanto se tiene que buscar también un canal de comunicación que sea seguro para el intercambio de la clave. Es importante que dicha clave sea muy difícil de adivinar ya que hoy en día los ordenadores pueden adivinar claves muy rápidamente. Los sistemas de clave secreta son muy rápidos pero carecen de lo anterior. Otro problema es la distribución y almacenamiento seguro de las claves cuando crece el número de personas con las que es necesario comunicarse, ya que el número de llaves crece con el cuadrado del número de personas con el que se debe mantener una comunicación segura.

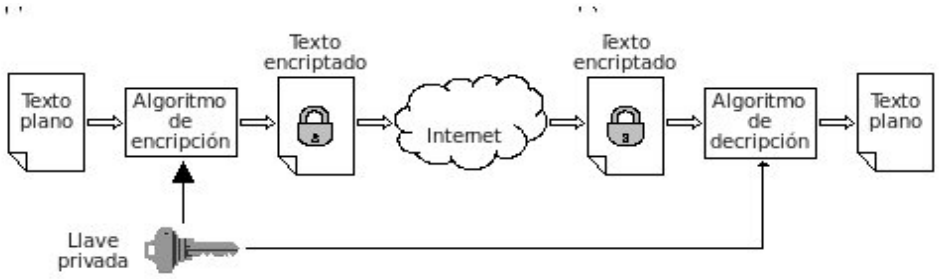

Fig. 1.4: Modelo del sistema de encriptación de llave privada *(Ferney, 2007).*

- *Asimétrico*

  También son llamados sistemas de cifrado de clave pública. Este sistema de cifrado usa dos claves diferentes. Una es la clave pública y se puede enviar a cualquier persona y otra que se llama clave privada, que debe guardarse para que nadie tenga acceso a ella Estas llaves están matemáticamente relacionadas de manera que la información encriptada con una de ellas, solo puede ser desencriptada con la otra. Estas claves tienen la propiedad de que a partir del conocimiento de una de ellas no puede determinarse la otra. El algoritmo de encriptación de llave pública más conocido es el RSA (por las iníciales de sus inventores Rivest, Shamir y Adleman).

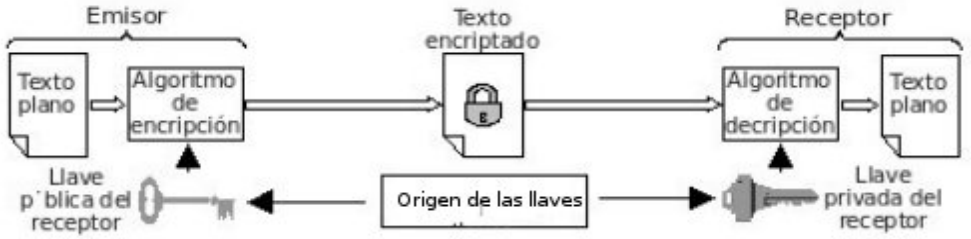

Fig. 1.5: Modelo del sistema de encriptación de llave pública *(Ferney, 2007).*

Para enviar un mensaje, el remitente usa la clave pública del destinatario para cifrar el mensaje. Una vez que lo ha cifrado, solamente con la clave privada del destinatario se puede descifrar, ni siquiera el que ha cifrado el mensaje puede volver a descifrarlo. Por ello, se puede dar a conocer perfectamente la clave pública para que todo aquel que se quiera comunicar con el destinatario lo pueda hacer.

No tiene el problema de distribución de llaves que tiene el sistema de encriptación de llave privada. El proceso de encriptación de llave pública es mucho más lento que el de llave privada y no es apropiado para encriptar grandes volúmenes de datos, aunque tienen fácil

intercambio de clave y cuentan con firma digital. Este sistema es útil para los casos en los que no se conoce a la persona con la cual se va establecer una comunicación segura o no se comparten llaves comunes con esa persona. Este es el caso de la mayoría de las transacciones de comercio electrónico.

Estos dos métodos se pueden combinar para garantizar la privacidad, autenticidad e integridad de los mensajes.

- *Híbridos*

Es el sistema de cifrado que usa tanto los sistemas de clave simétrica como el de clave asimétrica. Funciona mediante el cifrado de clave pública para compartir una clave para el cifrado simétrico. En cada mensaje, la clave simétrica utilizada es diferente por lo que si un atacante pudiera descubrir la clave simétrica, solo le valdría para ese mensaje y no para los restantes. Tanto PGP (Pretty Good Privacy, en español, *privacidad bastante buena*) como GnuPG (GNU Privace Guard) usan sistemas de cifrado híbridos" *(Marti, 2006)*. La clave simétrica es cifrada con la clave pública, y el mensaje saliente es cifrado con la clave simétrica, todo combinado automáticamente en un sólo paquete. El destinatario usa su clave privada para descifrar la clave simétrica y acto seguido usa la clave simétrica para descifrar el mensaje.

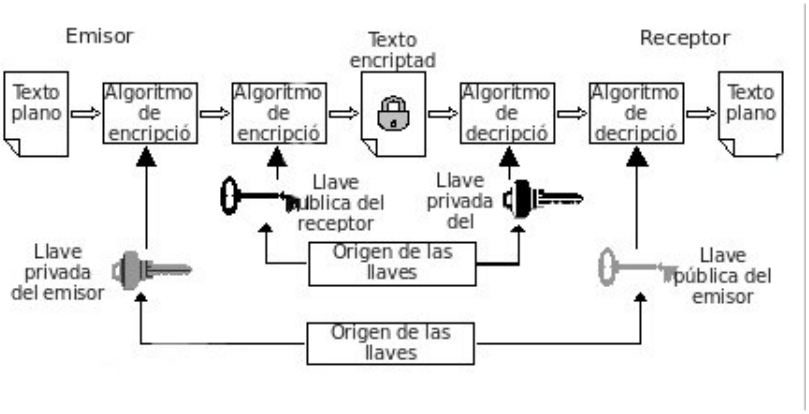

Fig. 1.6: Modelo del sistema de encriptación hibrido *(Ferney, 2007).*

1.5.5.2 Simétricos vs. Asimétricos

Sistemas	Ventajas	Desventajas
Simétricos	Muchos algoritmos: IDEA, CAST, 3DES, RC4, Blowfish, RC5 y 6, AES, etc. Implementación eficiente en software y hardware Rápidos o muy rápidos	Necesario secreto previo compartido. Comunicación segura de claves (procedimiento y canal) Número de claves crece con el cuadrado del número de comunicantes.
Asimétricos	Implementación eficiente en software y, algo menos, en hardware. No es necesario secreto previo compartido. Número de claves crece linealmente con número de comunicantes.	Pocos algoritmos RSA, ElGamal, Curvas Elípticas Muy lentos y costosos computacionalmente.

Tabla 1.1 Comparación entre criptosistemas simétricos y asimétricos.

La mejor solución es la combinación de ambos métodos para obtener las ventajas de ambos:

- la seguridad del criptosistema de clave pública.
- la rapidez de los criptosistemas de clave secreta.

La necesidad de proteger los datos de los atacantes ha traído consigo la búsqueda de métodos y mecanismos de seguridad capaces de mantener la integridad y la privacidad de la información. Otros individuos u organizaciones han desarrollado recursos que conjugan ambos mundos: el físico y el digital. Esto nace de la necesidad de compartir recursos entre varios usuarios que se encuentren distantes el uno del otro. En el mundo físico, las organizaciones adoptan políticas de seguridad para poder compartir recursos dentro de unos límites especificados. Las políticas de seguridad se hacen cumplir con la ayuda de los mecanismos de seguridad.

La protección de una red contra amenazas no puede ser lograda por una sola tecnología o servicio por lo que es necesario diseñar una estrategia balanceada que permita cubrir los puntos débiles en la seguridad de una red.

Capítulo 2 Descripción de los mecanismos de seguridad a utilizar.

2.1 Introducción

La creciente evolución de las redes de comunicaciones, así como su extensión y demanda a nivel mundial interconectando recursos informáticos de todo tipo hace necesario la implantación de servicios de seguridad para proteger la información en tránsito por dichas redes. Los servicios de seguridad utilizan los mecanismos implementados en una arquitectura de seguridad. Un mecanismo de seguridad implica un intercambio de unidades de datos entre entidades pares y la realización de una serie de funciones aplicadas a las unidades de datos (cifrado, cálculo de código de redundancia, y otros). Este intercambio entre entidades pares que tiene por objeto implementar un servicio de seguridad (un auténtico protocolo) lo denominan protocolo de seguridad.

En este Capítulo se hace un análisis de los mecanismos más empleados en la protección de la seguridad de un sistema. Se mencionan algunos protocolos que implementan dichos mecanismos en el momento de ser implementados, así como también se puntualizan las herramientas a utilizar en el desarrollo del trabajo.

2.2 Violación a la seguridad o privacidad

En Internet, los mensajes son transmitidos a través de un número de intermediarios antes de llegar a su destino, los intermediarios son generalmente Routers, cualquiera de los cuales puede copiar, modificar o borrar los mensajes. Es por esta razón que no se debe asumir que las comunicaciones de datos son privadas a menos que se utilice un mecanismo de encriptación para protegerlas.

El “romper” las claves de los sistemas de encriptación actuales es muy difícil, por lo que es poco probable que se pueda comprometer la seguridad de las transacciones que utilizan estos sistemas.

2.3 Violación a la integridad de los datos

Las fallas en la integridad de los datos que se envían a través de las redes son casi siempre accidentales, aunque existe la posibilidad de que los datos sean alterados intencionalmente para hacer algún daño. Un ataque que modifique datos, como por ejemplo, la alteración de un comando transmitido puede tener gran impacto en el sistema. Existen mecanismos para detectar si los datos han sido modificados, pero al igual que con los métodos de encriptación, no se han implantado tan ampliamente como deberían.

2.4 Negación de servicio

Los ataques de negación de servicio han sido denominados como las peores amenazas a la seguridad. Un ataque de negación de servicio tiene como objetivo hacer que cierto servicio no esté disponible para el público. El potencial que tiene este tipo de ataques para causar daños o pérdidas aumenta cada día. Defenderse de este tipo de ataques es particularmente difícil, porque para realizar este tipo de ataques los atacantes se apoyan en debilidades estructurales de los protocolos de comunicación más utilizados, como el Internet Protocol (IP).

2.5 Seguridad en el transporte de los datos

El transporte de los datos a través de las redes es el aspecto del comercio electrónico que ha recibido la mayor concentración de recursos para aseverar su seguridad. El brindar seguridad a los datos en transito implica garantizar la integridad y confidencialidad de los datos, y la autenticidad tanto del emisor como del receptor, garantizando que la información no sea modificada, leída o desviada del destino programado.

La necesidad de desarrollar mecanismos de seguridad para el transporte de los datos surge de la naturaleza insegura de Internet, ya que la información en transito se puede leer, copiar, modificar o eliminar mientras se transmite por las diferentes redes. Para brindar la seguridad necesaria en el transporte de los datos se han desarrollado diferentes mecanismos como la encriptación de datos, las firmas digitales y los certificados de autenticidad, de los cuales se hablará más adelante. También se han desarrollado protocolos de comunicación seguros como el Protocolo de Capa de Conexión Segura (Secure Socket Layer, SSL, que es el estándar para la transmisión segura de datos a través de Internet) o el Protocolo de Transferencia de Hipertextos Seguro (Secure - HyperText Transfer Protocol, S-HTTP) y protocolos para garantizar la seguridad de las transacciones de pago electrónico como el Protocolo de transacciones electrónicas seguras (Secure Electronic Transaction, SET).

2.6 Mecanismos de seguridad

Para enfrentar las amenazas, como las antes descritas, y asegurar las transacciones entre los módulos del SCADA es necesario que se cumplan los servicios de Privacidad, Integridad, Autenticidad y No repudiación que brinda la seguridad.

Para cumplir con estos requerimientos se han desarrollado mecanismos o combinaciones de mecanismos que permiten asegurar las transacciones. Estos mecanismos se describen a continuación.

Los mecanismos que se utilizan para implementar las políticas de seguridad que definan las responsabilidades y reglas a seguir para evitar amenazas, o minimizar los efectos en caso de ser producida alguna de estas, se definen como mecanismos de seguridad. Son la parte más visible del sistema de seguridad y la herramienta básica para garantizar la protección de los sistemas o la propia red.

Los mecanismos de seguridad se dividen en tres grandes grupos *(Huerta, 2002):*

- **de prevención**: aquellos que aumentan la seguridad de un sistema durante el funcionamiento normal de éste, previniendo la ocurrencia de violaciones a la seguridad;
- **de detección**: aquellos que se utilizan para detectar violaciones de la seguridad o intentos de violación;
- **de recuperación**: aquellos que se aplican cuando una violación del sistema se ha detectado, para retornar éste a su funcionamiento correcto.

Vale aclarar que aunque los tres tipos de mecanismos son importantes para la seguridad del sistema, se enfatiza más en el uso de los mecanismos de prevención y detección. Evitar un ataque, detectar un intento de violación o detectar una violación exitosa inmediatamente después de ocurrida, es mucho más productivo que restaurar el estado del sistema luego de una penetración.

Es más, “si consiguiéramos un sistema sin vulnerabilidades y cuya política de seguridad se implementara mediante mecanismos de prevención de una forma completa, no necesitaríamos mecanismos de detección o recuperación” (Huerta, 2002). Aunque realmente esto es muy difícil de conseguir en la práctica.

2.6.1 Mecanismos de prevención

Los mecanismos de prevención se pueden agrupar en cuatro grandes grupos *(Huerta, 2002)*:

- **Mecanismos de autenticación e identificación:** Hacen posible identificar entidades del

sistema de forma única y ya identificadas, autenticarlas. Son los mecanismos más importantes de cualquier sistema.

- **Mecanismos de control de acceso**: Controlan todo tipo de acceso sobre el objeto que se protege. El control de acceso más habitual es Control de Acceso Discrecional (*Discretionary Access Control,* DAC), implementado por los bits *rwx* y las listas de control de acceso para cada fichero (objeto) del sistema; sin embargo, también se permiten especificar Controles de Acceso Obligatorio (Mandatory Access Control, MAC).
- **Mecanismos de separación:** Permite separar los objetos dentro de cada nivel, evitando el flujo de información entre objetos y entidades de diferentes niveles, siempre que no exista una autorización explícita del mecanismo de control de acceso. Los mecanismos de separación se dividen en cinco grandes grupos, en función de cómo separan a los objetos, el mecanismo de separación más habitual es el de separación lógica o aislamiento, implementado en algunos sistemas mediante una Base Segura de Cómputo (TCB).
- **Mecanismos de seguridad en las comunicaciones:** Para proteger la integridad y la privacidad de los datos cuando se transmiten en la red, se utilizan mecanismos basados en su mayoría en la Criptografía. Aunque cada vez se utilizan más los protocolos seguros (como SSH o Kerberos, en el caso de sistemas Unix en red), aún es frecuente encontrar conexiones que se hacen en texto plano ya no sólo entre máquinas de una misma subred, sino entre redes o subredes diferentes.

Dentro de los más habituales se pueden referenciar:

2.6.1.1 Cifrado

Garantiza que la información no es inteligible para individuos, entidades o procesos no autorizados (confidencialidad).

Consiste en transformar un texto en claro mediante un proceso de cifrado en un texto cifrado, gracias a una información secreta o clave de cifrado. Cuando se emplea la misma clave en las operaciones de cifrado y descifrado, se dice que el criptosistema es simétrico. Estos sistemas son mucho más rápidos que los de clave pública, resultando apropiados para funciones de cifrado de grandes volúmenes de datos.

Se pueden dividir en dos categoría**s:**

- **cifradores de bloque**, que cifran los datos en bloques de tamaño fijo (típicamente bloques de 64 bits). Actúan sobre bloques de caracteres y sólo cifran cada bloque una vez se ha generado,
- **cifradores en flujo**, que trabajan sobre flujos continuos de bits. Cifran el mensaje carácter

a carácter, byte a byte, a medida que se van generando.

El cifrado es hoy mucho más avanzado y complejo que las formas antiguas. Se puede utilizar en muchísimos áreas, desde asegurar secretos militares hasta mantener la característica intelectual de forma confidencial.

Hay varias formas de técnicas del cifrado, algunas más fuertes o más seguras que otras. “Las técnicas de cifrado no representan en sí mismas la seguridad de un sistema, sólo son una herramienta, que conjuntamente con las firmas digitales, los controles de acceso, los sistemas de integridad de datos o el intercambio de autenticación, permiten definir el nivel de seguridad deseado en una comunicación *(Miller, 2006)”.* Es importante entender que estos mecanismos proporcionan un marco de seguridad genérico, aplicable a cualquier comunicación, ya sea telefónica, postal o por correo electrónico.

Las técnicas de cifrado deben tener las siguientes propiedades:

- Para los usuarios autorizados debe ser relativamente sencillo cifrar y descifrar datos.
- El esquema de cifrado no depende de mantener en secreto el algoritmo, sino de un parámetro del algoritmo llamado llave de cifrado.
- Para un intruso debe ser muy difícil determinar cuál es la llave de cifrado.

Los métodos de cifrado se dividen en dos categorías *(INEI, 1997):*

- **Cifradores de substitución** (incluyendo los códigos): En un cifrado de sustitución, cada letra o grupo de letras se sustituyen por otra letra o grupo de letras de una tabla de sustitución. Un ejemplo sería el Cifrado Del César.
- **Cifradores de transposición**: A diferencia de los cifradores de sustitución, que reemplazan las letras del texto en claro por símbolos, los cifradores de transposición reordenan las letras. Se cambia la posición de los caracteres en un mensaje.

La encriptación de datos es buena para proteger los datos en transito, pero cuando los datos se almacenan, generalmente se hace en “texto plano” y es allí donde los atacantes pueden obtenerlos.

2.6.1.2 Integridad de datos

Este mecanismo implica el cifrado de una cadena comprimida de datos a transmitir, llamada generalmente valor de comprobación de integridad (Integrity Check Value o ICV). Este mensaje se envía al receptor junto con los datos ordinarios. El receptor repite la compresión y el cifrado posterior de los datos y compara el resultado obtenido con el que le llega, para verificar que los

datos no han sido modificados.

Es necesario diferenciar entre la integridad de una unidad de datos y la integridad de una secuencia de unidades de datos, ya que se utilizan distintos modelos de mecanismos de seguridad para proporcionar ambos servicios de integridad. El mecanismo de integridad de datos soporta el servicio de integridad de datos. L*a* integridad de datos se centra en la corrección, la validez y la exactitud de los datos *(Marañon, 2000)*.

El término integridad de datos se refiere a la corrección y completitud de los datos. Cuando los contenidos de una base de datos se modifican, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Pueden añadirse datos no válidos a la base de datos, tales como, un pedido que especifica un producto no existente. Pueden modificarse datos existentes tomando un valor incorrecto, por ejemplo, si se reasigna un vendedor a una oficina no existente.

Se pueden hallar muchas formas en donde la integridad de datos puede ser comprometida. El error humano, en la entrada de datos es una de las causas superiores. Otra podría ser la inestabilidad del medio de comunicaciones al transmitir los datos. El uso de software que contenga algún tipo de virus también puede comprometer la integridad de datos. También, el mal funcionamiento del hardware, algo así como desplomes del disco, es otra causa de la integridad de datos comprometida *(Rosales, 2001).*

En el mundo físico generalmente, la verificación de la integridad de los "datos" se ha hecho de forma visual. La ausencia de señales de alteración significa que los datos no han sido cambiados.

Puede alterarse fácilmente las cosas cuando se almacenaron sobre una computadora y muchas personas podrían potencialmente ganar acceso, haciendo la integridad de datos mucho más difíciles en el mundo de la red.

El uso más común en la integridad de los datos son los sellos. Para asegurar la integridad de datos, se necesita de algún modo crear un sello que no pueda alterarse y pueda ser usado luego de verificar que los datos no han sido cambiados.

Otro ejemplo de integridad de datos es auditando. Las compañías emplean auditores externos para asegurar que sus libros de contabilidad afirmen correctamente la posición fiscal de las organizaciones. La auditoría puede ser una herramienta valiosa para mantener los sistemas y la integridad de los datos también en el mundo de la red.

2.6.1.3 Firma Digital

Este mecanismo implica el cifrado, por medio de la clave secreta del emisor, de una cadena comprimida de datos que se va a transferir. La firma digital se envía junto con los datos ordinarios.

Este mensaje se procesa en el receptor, para verificar su integridad. Juega un papel esencial en el servicio de no repudio.

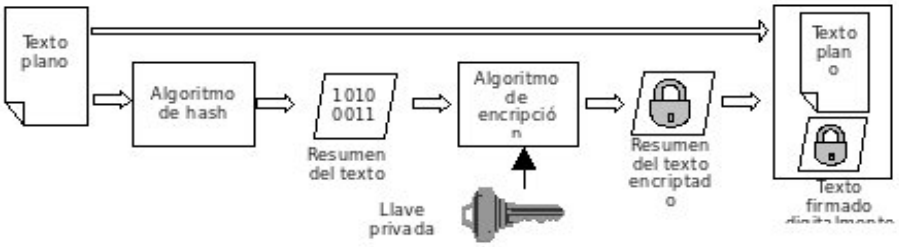

Fig. 2.1: Proceso de firma digital de un documento *(Ferney, 2007).*

El mecanismo de firma digital soporta los servicios de integridad de datos, autenticación de origen y no repudio con prueba de origen

La firma digital puede ser definida como una secuencia de datos electrónicos que se obtienen mediante la aplicación a un mensaje determinado de un algoritmo de cifrado asimétrico o de clave pública, y que equivale funcionalmente a la firma autógrafa en orden a la identificación del autor del que procede el mensaje. Desde un punto de vista material, la firma digital es una simple cadena o secuencia de caracteres que se adjunta al final del cuerpo del mensaje firmado digitalmente *(Martínez, 2002)*.

La firma digital es un procedimiento técnico que basándose en técnicas criptográficas trata de dar respuesta a las necesidades de Identidad, Integridad y No repudiación, a fin de posibilitar el tráfico comercial electrónico. El proceso de firma digital de un mensaje electrónico comprende en realidad dos procesos sucesivos: la firma del mensaje por el emisor del mismo y la verificación de la firma por el receptor del mensaje.

La firma digital se basa en la utilización combinada de dos técnicas distintas, que son la criptografía asimétrica o de clave pública para cifrar mensajes y el uso de las llamadas funciones hash o funciones resumen *(Martínez, 2002).*

Para verificar la firma digital, el receptor puede calcular el hash del mensaje, luego descripta la firma digital del mensaje utilizando la llave pública del emisor. Si la descripción tiene éxito, entonces se puede autenticar al emisor del mensaje. Si la comparación del hash realizado al mensaje coincide con el resultado de la descripción de la firma, entonces se puede asegurar que el mensaje no fue alterado.

2.6.1.4 Certificados Digitales

Un certificado digital es un archivo que identifica inequívocamente a un individuo o un sitio web y permite establecer comunicaciones seguras y confidenciales. Asocia el nombre de una entidad que participa en una transacción segura con la llave pública usada para firmar la comunicación con esa entidad en un sistema criptográfico *(Verisign, 1995)*.

Existen varios formatos para certificados digitales, los más empleados se rigen por el X.509, que especifica entre otras cosas, formatos estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación. Define el esquema para proveer servicios de autenticación a los usuarios de servicios de directorios X.500, el cual define una serie de recomendaciones para servicios de directorio. El directorio sirve como un repositorio de llaves públicas (certifican llaves públicas) *(Palacios, et al., 2000).*

Generalmente los certificados están "firmados" por una Autoridad Certificadora (CA). Los certificados digítales se presentan en tres versiones diferentes, pero la versión 3 es la que se encuentra con más frecuencia en los sistemas hoy en día. Los campos que contiene un certificado digital son los siguientes *(Carrasco, et al., 2001):*

- Versión
- Número serial
- Identificador del algoritmo de firma
- Nombre del emisor
- Periodo de validez
- Nombre del usuario
- Información de la llave pública del usuario
- Identificador único del emisor (versión 2 y 3)
- Identificador único del usuario (versión 2 y 3)
- Extensiones (versión 3)
- Firma digital de los campos anteriores

Los certificados tienen las siguientes características:

- Cualquier usuario con acceso a la llave pública de la Autoridad Certificadora puede recuperar la llave pública del usuario que fue certificado.
- Nadie más que la Autoridad Certificadora puede modificar el certificado sin ser detectado.

Como los certificados no pueden volver a crearse una vez que existan, si un usuario A tiene el certificado de otro usuario B, tiene la seguridad de que los mensajes que encripta con la llave pública de B solo pueden ser leídos por B, y los mensajes que hayan sido firmados con la clave privada de B no podrán volver a formarse.

Por la gran cantidad de usuarios, se han creado distintas Autoridades Certificadoras (CA), cada una posee un par de llaves pública y privada, con las cuales emiten sus certificados. Cada

Autoridad Certificadora tiene las llaves públicas de las otras Autoridades Certificadoras para así poder verificar la validez de los certificados que no sean emitidos por ella misma.

2.6.1.4.1 Autoridad de Certificación

"La Autoridad de Certificación (**CA**, *Certification Authority*) es la entidad encargada de firmar y revocar todos los certificados digitales. La Autoridad de Certificación es quien da validez a los certificados mediante la firma digital de estos con su clave privada *(Facultad de Informática Universidad Politécnica de Madrid, 2003)".* La confianza de los usuarios en ella es fundamental para el buen funcionamiento del servicio. Algunas autoridades de certificación pueden cobrar una cuota por su servicio mientras que otras autoridades competentes son gratuitas.

Las Autoridades de Certificación tienen la misión de emitir certificados a fin de proveer confianza sobre la autenticidad de las claves públicas de los demás o, en otros términos, que acrediten que una clave pública es realmente del usuario que se la atribuye y que todavía está vigente en el sistema.

La confianza de los usuarios en la CA es importante para el funcionamiento del servicio y justifica la filosofía de su empleo, pero no existe un procedimiento normalizado para demostrar que una CA merece dicha confianza.

La Autoridad de Certificación, por sí misma o mediante la intervención de una Autoridad de Registro (RA), verifica la identidad del solicitante de un certificado digital antes de su expedición o, en caso de certificados expedidos con la condición de revocados, elimina la revocación de los certificados al comprobar dicha identidad.
Las Autoridades Certificadoras pueden también revocar los certificados antes de que estos expiren por alguna de las siguientes razones *(Palacios, et al., 2000)*:

- Cuando se asume que la llave secreta del usuario está comprometida.
- El usuario no seguirá siendo certificado por la Autoridad Certificadora.
- Cuando se asume que el certificado emitido por la Autoridad Certificadora está comprometido.

<u>Misión de las CA.</u>

Las CA también se encargan de la gestión de los certificados firmados. Esto incluye las tareas de revocación de certificados que puede instar el titular del certificado o cualquier tercero con interés legítimo ante la CA por correo electrónico, teléfono o intervención presencial. La lista denominada CRL (Certificate Revocation List) contiene los certificados que entran en la categoría de revocados, por lo que es responsabilidad de la CA publicarla y actualizarla debidamente. Por otra parte, otra tarea que debe realizar una CA es la gestión asociada a la renovación de certificados

por caducidad o revocación.

Si la CA emite muchos certificados, corre el riesgo de que sus CRL sean de gran tamaño, lo que hace poco práctica su descarga para los terceros que confían. Por ese motivo desarrollan mecanismos alternativos de consulta de validez de los certificados, como servidores basados en el protocolo para comprobar el estado de los certificados en línea (Online Certificate Status Protocol, OCSP) y el protocolo de simple validación de certificados (Simple Certificate Validation Protocol, SCVP).

CAs Públicas y Privadas

Una CA puede ser o bien pública o bien privada. Los certificados de CA (certificados raíz) de las CAs públicas pueden o no estar instalados pero son reconocidos como entidades confiables, frecuentemente en función de la normativa del país en el que operan. Las CAs públicas emiten los certificados para la población en general (aunque a veces están focalizadas hacia algún colectivo en concreto) y además firman CAs de otras organizaciones.

Las entidades certificadoras actuales cobran por el servicio de firma de llaves y no suele ser precisamente asequible. Por otro lado, montar una entidad certificadora oficial también resulta muy costoso ya que se demandan unas ciertas garantías que detrás del negocio hay una cierta seguridad. Por tanto, "es habitual que los administradores de pequeñas redes se creen su propio certificado para firmar sus claves. De esta forma se puede disponer de comunicaciones encriptadas sin necesidad de entidades certificadoras *(Facultad de Informática Universidad Politécnica de Madrid, 2003)".*

2.6.1.5 Intercambio de autenticación.

Estos mecanismos hacen posible identificar entidades del sistema de una forma única, y una vez identificadas, autenticarlas (comprobar que la entidad es quién dice ser). Son los mecanismos más importantes en cualquier sistema, ya que forman la base de otros que basan su funcionamiento en la identidad de las entidades que acceden a un objeto. Por ejemplo, 'A' envía un número aleatorio cifrado con la clave pública de 'B', 'B' lo descifra con su clave privada y se lo reenvía a 'A', demostrando así que es quien pretende ser. Por supuesto, hay que ser cuidadosos a la hora de diseñar estos protocolos, ya que existen ataques para destruirlos.

Existen dos grados en el mecanismo de autenticación *(López, et al., 2007):*

- **Autenticación simple**. El emisor envía su nombre distintivo y una contraseña al receptor, quien los comprueba.
- **Autenticación fuerte.** Utiliza las propiedades de los criptosistemas de clave pública. Cada usuario se identifica por un nombre distintivo y por su clave secreta. Cuando un segundo

usuario desea comprobar la autenticidad de su interlocutor, habrá de asegurarse que aquel está en posesión de su clave secreta, para lo cual deberá obtener su clave pública.

Es un mecanismo utilizado para proporcionar autenticación de usuarios, de origen de datos, de interlocutores de comunicaciones. La mayor parte se basa en técnicas criptográficas. Para prevenir el phishing en los intercambios de autenticación deberían ser mutuos de cliente a servidor y viceversa.

2.6.1.5.1 Autenticación basada en Certificados Digitales

El control de acceso mediante certificados digitales se hace utilizando el protocolo SSL, el cual en su versión 3.0 soporta la autenticación tanto del servidor como del cliente de una sesión web. Para poder implantar este mecanismo es necesario que los clientes posean certificados digitales emitidos por alguna Autoridad Certificadora. Los certificados digitales contienen los datos del cliente así como su clave pública para poder realizar la encriptación asimétrica.

Este mecanismo presenta un nivel de seguridad más alto que el que presentan los mecanismos explicados anteriormente. Además, este mecanismo se implementa de forma transparente al usuario porque el intercambio de la información de identificación lo realiza el software cliente al momento de establecer la comunicación con el servidor.

Este mecanismo no sufre de los problemas de spoofing, de contraseñas débiles o de la captura de contraseñas cuando son transmitidas. El único problema es que se le impone al usuario la responsabilidad de conseguir un certificado personal emitido por una Autoridad Certificadora. Lo difícil es convencer al usuario a que se someta al proceso de obtención del certificado *(Carrasco, y otros, 2001).*

2.6.1.6 Control de Acceso

Es el esfuerzo para que sólo aquellos usuarios autorizados accedan a los recursos del sistema o a la red, como por ejemplo mediante las contraseñas de acceso.

Es la capacidad de dejar pasar o limitar el acceso de peticiones que proviene desde un cliente cualquiera. En un marco en el que se permita la identificación de flujos, el control de acceso podría llegar aplicarse independientemente para cada uno de los flujos.

Este mecanismo se utiliza para autenticar las capacidades de una entidad, con el fin de asegurar los derechos de acceso a recursos que posee. El control de acceso se puede realizar en el origen o en un punto intermedio, y se encarga de asegurar si el emisor está autorizado a comunicar con el receptor y/o a usar los recursos de comunicación requeridos. El mecanismo de control de acceso soporta el servicio de control de acceso.

2.6.1.7 Seguridad física

Hace referencia a las medidas organizativas y físicas como encerrar bajo llave los activos físicos como instalaciones e infraestructura de red, sistemas de computación y almacenamientos de datos, para proporcionar protección de monitorización no autorizada, robo, corrupción y desastres naturales.

Aquí se incluye la tecnología Tempest que trata de limitar las emanaciones electromagnéticas de información confidencial procedentes de pantallas de computador, impresoras, procesadores, etcétera. También se incluyen los sensores de presencia perimétricos y volumétricos, los soportes y cables de acero para inmovilización de PC en las mesas, sistemas antiincendios.

2.6.1.7 Protección contra virus o código malicioso

Normalmente implementado a través de programas automáticos que monitorizan los sistemas incluyendo los servicios de correo electrónico en busca de signos de presencia o actividad de virus, troyanos gusanos, spam, código móvil (como mal-applets, mal-ActiveX), adware, spyware, grayware, etcétera. También se incluyen el hardware antivirus para virus de red en forma de security appliances específicos o multifuncionales.

2.6.1.8 Seguridad del personal

Engloba los procedimientos para asegurar que el personal de la organización pueda ser de confianza para cumplir con las políticas de seguridad.

2.6.1.9 Base de computación confiable.

Permiten establecer los componentes de un sistema de computación que son de confianza. Se incluyen elementos como el mecanismo de control de acceso utilizado para proteger dichos componentes.

2.6.1.10 Listas de control de acceso (ACL) y etiquetas de seguridad.

Son mecanismos para definir los derechos de un principal (los principales pueden arrancar sesiones – sujetos sobre un computador que accederán a objetos utilizando operaciones) para acceder a un objeto concreto. Pueden también estar basados en capacidades (o listas de capacidades) que son tokens o tickets concedidos por un servicio de seguridad en base a los derechos de acceso definidos.

2.6.1.11 Tolerancia a fallos y sistemas de recuperación.

Sistemas que se diseñan para resistir a fallos hardware y errores software utilizando tecnologías como RAID, replicación de datos, clúster de servidores y servicios SAI/UPS para alimentación

eléctrica de emergencia.

2.6.1.12 Backups.

Mecanismo esencial para recuperar la pérdida o corrupción de datos mediante copias de seguridad periódicas y la necesaria verificación de su recuperación.

2.6.1.13 Notario &ndash.

Fedatario electrónico, una TTP (Tercera parte de confianza) que puede utilizarse para proporcionar servicios a otras entidades o a un sistema, estos servicios incluyen prueba de integridad, de origen de destino, de tiempo de transmisión, de no repudio, etc.

2.6.1.14 Tráfico de relleno.

Consiste en insertar datos sin significado en un mensaje normalmente cifrado para protegerlo contra la pérdida de confidencialidad debida a ataques intencionados que utilizan el análisis de tráfico, así el atacante no sabe si se está enviando información, ni qué cantidad de datos útiles se está transmitiendo. Son más efectivos cuando el proceso esta protegido por medidas criptográficas.

2.6.1.15 Cortafuegos.

Se trata de mecanismos software y hardware para restringir el tráfico de red a través de canales bien definidos y fácilmente monitorizados utilizando técnicas como el filtrado de paquetes y los servicios proxy.

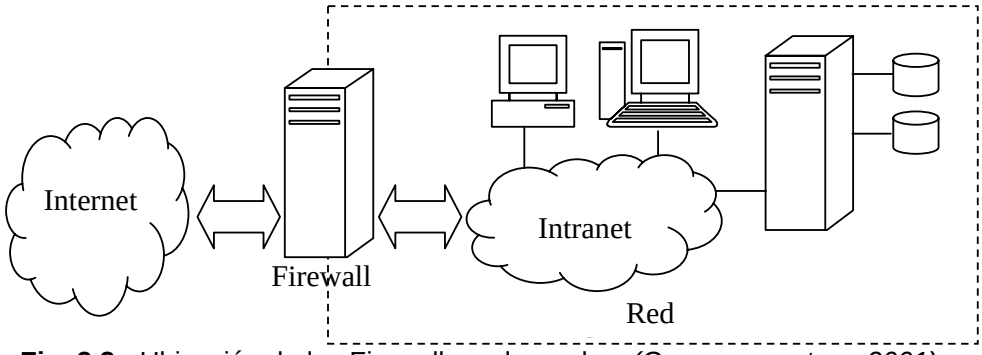

Fig. 2.2 - Ubicación de los Firewalls en las redes *(Carrasco, y otros, 2001).*

Los cortafuegos brindan protección solo contra atacantes “aficionados”, un atacante determinado puede violar la protección que brindan los firewalls aunque estén bien configurados. Las estadísticas muestran que el 75% de los firewalls pueden ser violados.

2.6.1.16 Control de encaminamiento.

Permite enviar determinada información por determinadas zonas consideradas clasificadas. Asimismo posibilita solicitar otras rutas, en caso que se detecten persistentes violaciones de integridad en una ruta determinada.

Son métodos para asegurar que tipos específicos de datos sensibles se transmitan sólo a través de enlaces seguros. Aquí se incluyen las técnicas *hop integrity* para defenderse contra ataques de denegación de servicios y el protocolo IDIP para la erradicación de intrusiones.

2.6.1.17 Unicidad.

Consiste en añadir a los datos un número de secuencia, la fecha y hora, un número aleatorio, o alguna combinación de los anteriores, que se incluyen en la firma digital o integridad de datos. "De esta forma se evitan amenazas como la reactuación o resecuenciación de mensajes*" (Marañon, 2000).*

La implementación de estos mecanismos se puede realizar de forma parcial o global, cada persona debe decidir qué requerimientos exige a las comunicaciones que mantiene con cada interlocutor.

2.6.2 Mecanismos de detección.

Es muy importante que los mecanismos de seguridad de un sistema estén diseñados e implementados de tal forma, que prevengan el acceso no autorizado a sus recursos y datos. Sin embargo prevenir los problemas de seguridad en su totalidad, es una tarea difícil debido al factor humano.

Es posible seguir minimizando los riesgos, tratando de detectar los intentos de ataques o intrusiones para poder tomar las acciones necesarias y reducir su impacto. En 1980, mientras se introducía el concepto de detección de intrusos, se definió una intrusión como la posibilidad potencial de un intento deliberado, no autorizado, de acceder información, manipularla o eliminar la funcionalidad del sistema.

Cualquier mecanismo de seguridad con este propósito puede ser considerado un sistema de detección de intrusiones (Intrusion Detection Systems, IDS) o normalmente, sistemas de detección de intrusos, pero generalmente sólo se aplica esta denominación a los sistemas automáticos y especializados en desarrollar este tipo de tareas de detección.

Generalmente existen dos grandes enfoques a la hora de clasificar a los sistemas de detección de intrusos: en función de qué sistemas vigilan, o en función de cómo lo hacen.

Si se toma la primera de estas aproximaciones los sistemas de detección de intrusos generalmente están divididos en 2 clases, los sistemas de detección de intrusos a nivel de red o NIDS y a nivel de host o HIDS. Existe una nueva tercera clase la cual ha sido denominada sistema de detección de intrusos a nivel de aplicación o ApplicationIDS. Esta última clase también ha sido denominada como "Firewall de aplicación".

Desafortunadamente, hoy en día, estos sistemas basan su funcionamiento en el reconocimiento de firmas de ataques previamente conocidos, las cuales son comparadas. En el caso de los NIDS contra el tráfico de red capturado por un sensor previamente establecido, o en el caso de los ApplicationIDS contra el tráfico de red dirigido hacia un servidor determinado, el cual generalmente es un servidor Web.

Los HIDS han basado su funcionamiento en módulos que son aplicados directamente al kernel de un sistema en particular, añadiendo nuevas características de auditoria y seguridad que evitan que un usuario local escale privilegios, teniendo acceso a recursos y procesos de forma no autorizada. Además, se han desarrollado nuevas investigaciones enfocadas al desarrollo de tecnologías adaptivas basadas en los HIDS.

En la actualidad esta división queda algo pobre, ya que cada día se avanza más en el diseño de sistemas de detección de intrusos que no tendrían cabida en ninguna de las categorías anteriores o que corresponderían a múltiples clases al mismo tiempo.

La segunda gran clasificación de los IDS se realiza en función de cómo actúan estos sistemas. Actualmente existen dos grandes técnicas de detección de intrusos, las basadas en la detección de anomalías y las basadas en la detección de usos indebidos del sistema.

En la detección de anomalías se supone que una intrusión se puede ver como una anomalía en el sistema, por lo que si se establece un perfil del comportamiento normal de los sistemas es posible detectar las intrusiones, una intrusión sería básicamente una desviación del comportamiento normal establecido. Para la detección de usos indebidos se establecen firmas para cada uno de los diferentes ataques conocidos y algunas de sus variaciones. Así, mientras que la detección de anomalías conoce lo normal y detecta lo que no lo es, la detección de usos indebidos se limita a conocer lo anormal para poderlo detectar. Es esta última técnica la que comúnmente se utiliza en los IDS actuales, la cual posee grandes limitaciones.

2.6.3 Mecanismos de recuperación.

En este grupo de mecanismos de seguridad se puede encontrar un subgrupos denominado *mecanismos de análisis forense*, cuyo objetivo no es simplemente retornar al sistema a su modo de trabajo normal, sino averiguar el alcance de la violación, las actividades de un intruso en el

sistema, y la puerta utilizada para entrar, de esta forma se previenen ataques posteriores y se detectan ataques a otros sistemas de nuestra red.

Para garantizar la seguridad se crearon diversos protocolos que emplearan los mecanismos antes expuestos.

2.7 Protocolos de Seguridad.

Un protocolo de seguridad define las reglas que gobiernan las comunicaciones, diseñadas para que el sistema pueda soportar ataques de carácter malicioso. Protegerse contra todos los ataques posibles es generalmente muy costoso, por lo que los protocolos son diseñados bajo ciertas premisas con respecto a los riesgos a los cuales el sistema está expuesto.

Un protocolo es una serie de pasos, que involucra a dos o más entidades, diseñado para realizar una tarea particular.

1. Todos los principales deben conocer los pasos del protocolo de antemano.
2. Todos deben estar de acuerdo en seguir el protocolo.
3. El protocolo no admite ambigüedades.
4. El protocolo debe ser completo, define qué hacer en cualquier circunstancia posible.
5. No debe ser posible hacer más (o aprender más) que lo que el protocolo define.

Un protocolo criptográfico es un protocolo que usa funciones criptográficas en algunos o todos los pasos.

2.7.1 SSL (Secure Socket Layer)

“El Secure Socket Layer o SSL fue desarrollado por Netscape para brindar un canal de comunicación seguro entre los servidores web y los clientes web que decidan utilizarlo. SSL es un protocolo del stack de protocolos TCP/IP *(Ghosh, 1998)”*. SSL se ubica entre la capa de transporte y la capa de aplicación, debido a que cada capa hace uso de los servicios de las capas inferiores.

SSL puede utilizarse para proteger las comunicaciones que se realizan empleando cualquiera de los protocolos de la capa de aplicación (Telnet, FTP, SMTP, etcétera), sin embargo, actualmente el SSL solo funciona con el protocolo de transferencia de hipertextos (HTTP). Asegura el canal de comunicación brindando encriptación end-to-end de los datos que se transmiten entre el servidor y el cliente. Son varios los sistemas y tecnologías que se han desarrollado para intentar implementar estos aspectos en las transacciones electrónicas, siendo sin duda SSL el más conocido y usado en la actualidad.

SSL permite la Confidencialidad y la Autentificación en las transacciones por Internet, siendo usado principalmente en aquellas transacciones en la que se intercambian datos sensibles. Es

una de las formas base para la implementación de soluciones Infraestructuras de Clave Pública (PKI).

La idea que persigue SSL es encriptar la comunicación entre servidor y cliente mediante el uso de llaves y algoritmos de encriptación.

“SSL es un sistema de protocolos de carácter general, y está basado en la aplicación conjunta de Criptografía Simétrica, Criptografía Asimétrica (de llave pública), certificados digitales y firmas digitales para conseguir un canal o medio seguro de comunicación a través de Internet” (Universidad de Venezuela Simón Bolívar, 2002).

De los sistemas criptográficos simétricos, se aprovecha la rapidez de operación, mientras que los sistemas asimétricos se usan para el intercambio seguro de las claves simétricas, consiguiendo con ello resolver el problema de la confidencialidad en la transmisión de datos.

SSL se introduce como una capa adicional, situada entre la capa de Aplicación y la capa de Transporte, sustituyendo los sockets del sistema operativo, lo que hace que sea independiente de la aplicación que lo utilice.

SSL proporciona servicios de seguridad a la pila de protocolos, encriptando los datos salientes de la capa de Aplicación antes de que estos sean segmentados en la capa de Transporte y encapsulados y enviados por las capas inferiores. Los algoritmos, longitudes de clave y funciones hash de resumen usados en SSL dependen del nivel de seguridad que se busque o se permita.

Este protocolo brinda cifrado de alto nivel de los datos intercambiados (se cifran incluso las cabeceras HTTP), autenticación del servidor (y si es necesario también del cliente) e integridad de los datos recibidos.

Cuando el cliente pide al servidor una comunicación segura, el servidor abre un puerto cifrado, gestionado por un software llamado Protocolo SSL Record, situado encima de TCP. Será el software de alto nivel, Protocolo SSL Handshake, quien utilice el Protocolo SSL Record y el puerto abierto para comunicarse de forma segura con el cliente (Universidad de Venezuela Simón Bolivar, 2002).

Para establecer una comunicación SSL es necesario que previamente el cliente y el servidor realicen un proceso de reconocimiento mutuo y de petición de conexión que recibe el nombre de apretón de manos o Handshake, que en este caso está controlado por el Protocolo SSL Handshake, que se encarga de establecer, mantener y finalizar las conexiones SSL. Durante el mismo se negocian los parámetros generales de la sesión y los particulares de cada conexión.

Para empezar a transmitir datos cifrados en necesario que cliente y servidor se pongan de acuerdo respecto a la forma común de encapsular los datos que se van a intercambiar, es decir,

qué formato de datos se va a usar en la transmisión cifrada. Esto se realiza mediante el Protocolo de Registro SSL (Protocolo SSL Record).

La tecnología basada en los protocolos SSL proporcionó grandes avances en la implantación de sistemas de comunicación seguros, que han hecho posible un crecimiento importante en las transacciones por Internet.

SSL proporciona una buena seguridad de que los datos no van a ser capturados por extraños de forma útil en el proceso de transferencia de los mismos, pero no proporciona ninguna seguridad después de finalizar la conexión.

A esto se añade que SSL sólo proporciona seguridad en la transacción cliente-servidor seguro, pero queda otra fase de la transacción, la que va desde el servidor seguro a la empresa emisora, y sobre ésta no se tiene ningún tipo de control.

SSL carece de muchos de los elementos necesarios para construir un sistema de transacciones seguras usando Internet. Para intentar atenuar estos fallos se ha intentado sacar al mercado y estandarizar otros sistemas diferentes, como SET (Transacciones Electrónicas Seguras), pero el caso es que hasta ahora ninguno de ellos ha conseguido desplazar a SSL.

2.7.1.1 TLS (Transport Layer Security)

Para intentar corregir las deficiencias observadas en SSL se buscó un nuevo protocolo que permitiera transacciones seguras por Internet, sobre todo teniendo en cuenta que SSL es propiedad de la empresa Nestcape. El resultado de esta búsqueda fue el protocolo TLS, que permite una compatibilidad total con SSL siendo un protocolo público.

TLS busca integrar en un esquema tipo SSL al sistema operativo, a nivel de la capa TCP/IP, para que el efecto "túnel" que se implementó con SSL sea realmente transparente a las aplicaciones que se están ejecutando.

A la hora de intercambiar las claves, TLS no soporta el algoritmo simétrico Fortezza, que sí es soportado por SSL. Esto es debido a la búsqueda de un código público, ya que Fortezza es de propiedad privada.

“TLS utiliza dos campos más en el MAC que SSL, lo que lo hace más seguro” (Universidad de Venezuela Simón Bolivar, 2002).

2.7.2 IIOP (Internet Inter-ORB Protocol)

Para resolver los problemas inherentes a sistemas heterogéneos y distribuidos, que dificultan la implementación de verdaderas aplicaciones empresariales, los proveedores de software están ofreciendo interfaces de programación y protocolos estándares.

Originalmente los esfuerzos de la OMG (Object Management Group) se centraron en resolver un problema fundamental: “Cómo lograr que sistemas distribuidos orientados a objetos implementados en diferentes lenguajes y ejecutándose en diferentes plataformas interactúen entre ellos”. Más allá de los problemas planteados por la computación distribuida, problemas más simples como la falta de comunicación entre dos sistemas generados por compiladores de C++ distintos que corren en la misma plataforma frenaron los esfuerzos de integración no bien comenzados. Para opacar aún más el escenario, distintos lenguajes de programación ofrecen modelos de objetos distintos. Los primeros años de la OMG estuvieron dedicados a resolver los principales problemas de cableado. Como resultado se obtuvo la primera versión de CORBA que es una arquitectura estándar para sistemas de objetos distribuidos, publicado en 1991.

Como una de las propiedades más importantes que posee el middleware se podría citar a su soporte de interfaces y protocolos estándares. Lo que permitiría hacer uso de todo aquel protocolo que haya sido estandarizado con anterioridad.

Debido al importante rol que juegan una interfaz estándar en la portabilidad de aplicaciones y un protocolo estándar en la interoperabilidad entre aplicaciones, varios esfuerzos se han realizado para establecer un estándar que pueda ser reconocido por los mayores participantes en la industria del software. Algunos de ellos han alcanzado instituciones como el Instituto Nacional Americano de Estándares (American National Standard Institute, ANSI) y la Organización Internacional de Normalización (International Organization for Standardization, ISO); otros han sido propuestos por consorcios de industrias como la OFP (Open Software Foundation) y la OMG.

Como una de las metas en las especificaciones de CORBA se halla la interoperabilidad. En estas especificaciones se definieron estándares para permitir la comunicación entre implementaciones realizadas por desarrolladores diferentes. Uno de estos estándares se denomina General Inter-ORB Protocol (Protocolo General Inter.-ORB) o GIOP. Se pretendía que este GIOP pudiera ser implementado sobre cualquier capa de transporte con conexiones. La Implementación de GIOP para Internet utiliza TCP/IP y se denomina IIOP (Protocolo Inter.-ORB para Internet).

2.7.2.1 GIOP (Protocolo General Inter.-ORB)

GIOP especifica el formato de las comunicaciones CORBA sin especificar ningún transporte específico. Para eso se define IIOP, que no es más que GIOP sobre TCP/IP. CORBA lo define como un protocolo de comunicaciones entre los distintos objetos. Las reglas de comunicación entre los objetos clientes y servidor son definidas por el protocolo IIOP. El protocolo IIOP establece que toda petición de servicio del cliente deberá ser trasferido al servidor a través del ORB; y toda respuesta del servidor llegara al cliente pasando por el ORB. Cuando el cliente y una implementación de objeto están distribuidos por una red, ellos usan el protocolo GIOP/IIOP

suministrado por la arquitectura para lograr la comunicación entre ellos.

Este estándar de interoperabilidad definido por CORBA, fue desarrollado para permitir que diferentes ORBs se pudieran comunicar. La interoperabilidad de CORBA ofrece una infraestructura que hace que diferentes implementaciones de ORBs sean compatibles. Esta parte del estándar se sitúa en la capa de transporte del modelo TCP/IP.

La arquitectura de interoperabilidad entre ORBs basada en el protocolo GIOP que especifica la sintaxis de transferencia de un conjunto de mensajes estándar para la comunicación entre ORBs se basó en un protocolo de transporte orientado a la conexión. La especificación de este consiste en los siguientes elementos:

La Representación Común de Datos (CDR): es una sintaxis de transferencia, que mapea los tipos de datos de IDL en una representación de bajo nivel para su transferencia "por el hilo" entre ORBs.

La Referencia de Objetos Interoperable (IOR): define el formato de una referencia a un objeto remoto. Una IOR consiste en varios perfiles etiquetados, y sus componentes pueden llevar diversa información según se necesite. La IOR típica habitualmente contiene la versión del protocolo, la dirección del servidor y una secuencia de octetos que identifica al objeto remoto (clave del objeto).

Los formatos de mensaje definidos de GIOP: Define varios tipos de mensajes que pueden intercambiar los ORB de la parte del cliente y del servidor. Sólo dos de esos tipos de mensajes son necesarios para realizar la comunicación (Request y Reply), los otros corresponden a mensajes de control. Los mensajes se intercambian entre agentes para facilitar las peticiones de objetos, localizar implementaciones de objetos y gestionar los canales de comunicación. Los mensajes posibles son:

- *Request* se envía con el propósito de invocar el método remoto Reply se devuelven como respuesta del Request. Por lo general contiene los datos devueltos por el método remoto. En otros casos o la descripción de la excepción que fue lanzada en el lado del servidor.
- *CancelRequest* se utiliza para cancelar una petición que había sido enviada (ya no se espera una respuesta)
- *LocateRequest* se utiliza para verificar si el servidor conoce y soporta cierto objeto remoto, y en caso contrario, a qué dirección deberían enviarse las peticiones a ese objeto.
- *LocateReply* se envía desde el servidor como respuesta a LocateRequest. Si es necesario, puede contener la nueva dirección del objeto remoto que se ha movido.
- *CloseConnection* es enviado por el servidor cuando desea indicar que no proporcionará más respuestas en un futuro.

- *MessageError* se envía como respuesta a mensajes no válidos o mal formados. No se utiliza para informar de errores más allá del sistema de mensajes; tales errores deben enviarse usando el mensaje Reply.
- *Fragment* es un mensaje posterior, que es continuación del mensaje previo.

La capa de transporte para poder transmitir mensajes por el protocolo GIOP debe satisfacer las siguientes condiciones:

- Debe ser orientado a la conexión.
- Las conexiones son full-dúplex.
- Las conexiones son simétricas.

GIOP es un protocolo de comunicaciones muy eficiente, pensado para aligerar las comunicaciones lo más posible. En él, simplemente se define el formato de los mensajes, y luego, cada ORB es responsable de convertir dichos mensajes en datos para la máquina en la que se está ejecutando, permitiendo así la interacción entre distintas implementaciones de CORBA ejecutándose en máquinas distintas, con sistemas operativos distintos, e incluso, con representación de datos distinta. GIOP define todos los aspectos de interoperabilidad entre distintos ORBs, independientemente del nivel de transporte.

El objetivo del protocolo IIOP es estandarizar la forma de comunicar dos ORB’s (posiblemente de diferente fabricante) utilizando como dorsal una red TCP/IP. Todo ORB conforme a CORBA debe implementarlo o proporcionar un semipuente a él.

En particular, IIOP se usa como el protocolo de transporte. Una conexión IIOP permite llevar invocaciones en el sentido inverso bajo ciertas condiciones que no comprometan la seguridad de la conexión. Este estándar lleva bastante tiempo en el mercado y podríamos decir que tiene una difusión algo limitada debido a su alta complejidad.

2.7.3 SSLIOP (Secure Socket Layer Inter - ORB Protocol)

Esta extensión brinda encriptación a los pedidos y respuestas de CORBA, protección a los mensajes de no ser modificados por una tercera parte y autenticación entre cliente y servidor.

En el tema de la seguridad, es completamente transparente en las aplicaciones. Es posible introducir comunicaciones seguras permitiendo solicitudes entrantes que son inseguras.

En cuanto a la seguridad que brinda SSLIOP, este depende de SSL para brindar tres facetas de seguridad:

- protección de la integridad, para proteger mensajes de ser modificados.

- encriptación, proporcionando protección de que los mensajes no sean vistos por terceras partes.
- autenticación, asegurando que una o ambas partes involucradas sean los que dicen ser.

Estas tres facetas no necesitan ser previstas al mismo tiempo. La autenticación puede ser limitada a una parte (normalmente al servidor) o puede ser prevista pero sin encriptación.

Siempre es necesario la autenticación de al menos un lado de las conexiones para evitar ataques. De lo contrario el atacante podría enmascararse como usuario común y conectarse, posibilitando así que pueda interceptar y corromper los datos.

Está la opción de no autenticar ningún lado, pero no es conveniente. Siempre es bueno autenticar ambos lados (el cliente y el servidor) o autenticar solo el servidor. Pero no es posible autenticar solo el cliente, aunque, si se solicita se podría permitir esta posibilidad. Para proporcionar autenticación es necesario construir las llaves (pública y privada).

La encriptación es requerida donde la transmisión de datos sea sensible a ser dañada. En algunos casos, solo se requiere de protección de integridad de datos, previniendo el ataque de modificación de datos durante su transmisión.

2.8 TAO (The ACE ORB)

2.8.1 Preámbulo

Los middlewares son ampliamente utilizados como parte intrínseca de los sistemas computacionales, tanto industriales como corporativos o de gestión. Las necesidades y características de cada sistema, hacen que se hayan desarrollado diversos tipos de middleware con diferentes características y capacidades. Este análisis se centra en la evaluación de los más importantes y que a la vez concuerden con los objetivos trazados en el diseño del SCADA.

Existen tres modelos principales para comunicar aplicaciones en red, punto a punto, encuesta/respuesta, y publicador/suscriptor.

El Middleware del proyecto SCADA utiliza los modelos publicador/subscriptor y encuesta/respuesta, también conocido por cliente/servidor.

El modelo cliente/servidor es mayormente utilizado en el mundo informático. Un cliente se conecta directamente a un servidor y envía encuestas al mismo. El servidor procesa la información y envía la respuesta al cliente. La ventaja del cliente es muy confiable, él conoce en qué momento ha sido hecha la encuesta y es capaz de saber, en un intervalo semipreciso, cuándo la información estará a su disposición.

Las tecnologías middleware basadas en el modelo encuesta/respuesta, de mayor nivel de abstracción y más utilizadas actualmente son la siguientes, Remote Method Invocation (RMI), desarrollada por Sun Microsystem, Distributed Component Objects Model (DCOM) y sus variantes, desarrollada por Microsoft, CORBA, desarrollada por el consorcio de empresas de OMG.

Las aplicaciones publicar/suscribir, son típicamente aplicaciones distribuidas con nodos de terminales que se comunican con otros homólogos mediante el envío (publicación) y la recepción (suscripción) de datos anónimamente. Los publicadores son responsables por la entrega de los datos apropiados y el envío de estos a todos los suscriptores registrados, así como los suscriptores son los responsables por la recepción de los datos, desde el publicador apropiado y la presentación de estos a las aplicaciones de usuario que los necesiten.

La tecnología middleware empleada en el proyecto SCADA es CORBA. Actualmente es una de las opciones tecnológicas más importantes a la hora del desarrollo de sistemas software distribuido. CORBA es un estándar que establece una plataforma de desarrollo de sistemas distribuidos, facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos

El ORB es el núcleo de cada una de las implementaciones CORBA, este brinda mecanismos para ejecutar las invocaciones remotas y la comunicación entre los objetos cliente y servidor además de una serie de capacidades relacionadas con esta función.

Actualmente existen ORBs con diversos lenguajes y plataformas. Entre los ORBs que se podrían utilizar en el SCADA, que cumpla con los requisitos propuestos por la dirección del proyecto se encuentra TAO, el ORB de ACE, implementación de CORBA para tiempo real.

2.8.2 ACE+TAO

TAO es una plataforma robusta para el intercambio de datos en sistemas distribuidos ampliamente utilizado por los desarrolladores ofreciendo capacidades insustituibles en los servicios de CORBA que implementa. “Inicialmente fue pensado para tiempo real, pero ha llegado a ser un ORB de propósito general muy capaz, adaptable a cualquier tipo de sistema distribuido sea de tiempo real o no” *(Schmidt, 2007).*

ACE es un marco de trabajo (framework) orientado a objetos que implanta muchos patrones centrales para software de comunicación concurrente. Estos patrones y componentes se unen al agente de invocación de objetos ORB, para de esa manera lograr un componente robusto y eficiente de comunicaciones a bajo nivel.

TAO es una implementación en tiempo real de CORBA que proporciona eficiencia, previsibilidad, escalabilidad y calidad de servicio. Crea un tratamiento de prioridades para evaluar las peticiones

de los objetos remotos, por lo cual es ideal para emplearse en los procesos críticos del sistema. Además, los resultados en cuanto a la sobrecarga, velocidad y uso de la memoria, solo superados por ORBit, son aceptables y satisfactorios para el SCADA. Permite a los clientes invocar operaciones en objetos distribuidos sin preocuparse por la ubicación del objeto, lenguaje de programación, Sistema Operativo, protocolos de comunicación e interconectores, y hardware.

Lamentablemente, ACE+TAO es muy costoso, lento, y propenso a errores para los investigadores y empresas desarrolladores de forma independiente. Pero, se han identificado una serie de patrones y componentes frameworks concisas que pueden ser aplicables sistemáticamente para eliminar varios de esos errores y aspectos de desarrollo y mantenimiento de aplicaciones distribuidas (Schmidt, 2007).

Ambos, ACE y TAO, están soportados comercialmente por múltiples compañías que utilizan modelos de negocio de software libre. Este soporte llega en el momento en que el software libre alcanza su masa crítica, y los usuarios no solo lo aceptaban, sino también reconocían su importancia en el desarrollo de software. En particular, un creciente número de proyectos y patrocinadores aplican ACE+TAO para ayudar a reducir sus costos de desarrollo, mejorar su calidad de servicios y decrementar el tiempo de salida al mercado *(Schmidt, 2007).*

2.8.3 Mecanismos de seguridad en TAO.

TAO implementa un módulo de seguridad ofreciendo un gran número de funcionalidades. El módulo implementado por TAO se divide en tres niveles de seguridad, seguridad nivel uno (SecurityLevel1), seguridad nivel dos (SecurityLevel2) y seguridad nivel tres (SecurityLevel3). Cada uno de estos niveles tiene sus propias funcionalidades. Haciendo posible escoger el nivel de seguridad apropiado para una aplicación. Este módulo incluye las políticas de servicio que propone el ORB (Object Request Broker) junto a un gran conjunto de clases que facilitan la implementación de servicios de seguridad.

La versión actual de TAO trae parte de dicho módulo implementado. Lamentablemente aún no ha concluido la implementación de todo el conjunto de funcionalidades que este ofrecerá. En el desarrollo del Middleware no se usó este módulo para su implementación, se destinó una línea de trabajo que reparara los problemas de seguridad para el SCADA en general, dejando fuera los mecanismos a implementar para la seguridad en el Middleware.

TAO implementa parte del primer grupo de mecanismo de seguridad, *mecanismos de prevención*. En el módulo de seguridad viene incluido la implementación del protocolo SSLIOP *(Secure Socket Layer Inter - ORB Protocol).* En computación distribuida, SSLIOP es IIOP *(Internet Inter-ORB Protocol)* implementado sobre SSL *(Secure Socket Layer)*, proporcionando confidencialidad, autenticación y garantiza que los datos viajen de forma segura, usando algoritmos criptográficos

para encriptar la información que se trasmite a través de él. El protocolo SSLIOP provee seguridad en la capa de transporte para todos los pedidos de transmisión.

2.9 Algunas Herramientas

2.9.1 Sistema Operativo GNU/Linux.

Este proyecto será desarrollado a partir de las herramientas que brinda el software libre. Buscando la independencia tecnológica que este brinda al posibilitar la libertad de uso y distribución de los programas sin incurrir en litigios de licenciamiento o asuntos legales.

Distribución de Linux: Debian GNU/Linux.

Debian es la única distribución importante de GNU/Linux mantenida solamente por voluntarios, es decir, sin un enfoque comercial, esto tiene ventajas y desventajas. Se actualiza la distribución diariamente, apareciendo paquetes nuevos de software constantemente. Al mismo tiempo, existe un compromiso de calidad, no se desea distribuir software con errores.

Como desventajas tiene un mayor componente técnico que otras distribuciones. También, es posible que ciertos paquetes no estén tan actualizados como debieran, quizás porque sus desarrolladores han dejado de actualizarlos y nadie se ha hecho cargo.

Se decidió usar la distribución Debian porque es una distribución de desarrollo muy estable, por lo que los paquetes que se desarrollen en él y quieran ejecutarse utilizando cualquier distribución siempre serán estables. A diferencia de otras distribuciones tiene un magnífico soporte de estabilidad en las aplicaciones (no requieren ser compiladas en la máquina que las esté usando). Los módulos del LDAP (Lighweight Directory Access Protocol) se pueden ejecutar sin problemas permitiendo que los usuarios usen sus sesiones en cualquier máquina dentro del área de trabajo, ahorrando recursos de hardware.

2.9.2 IDE a utilizar: Eclipse

Con el objetivo de realizar el prototipo del módulo de seguridad en el Middleware se investigó acerca la herramienta Eclipse, versión 3.3.0.

El Eclipse es un Entorno Integrado de Desarrollo (IDE) multiplataforma libre para crear aplicaciones clientes de cualquier tipo.

Fue creado originalmente por International Business Machines Corporation (IBM), ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El IDE de Eclipse emplea módulos (plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos donde las funcionalidades están todas incluidas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java, como Perl o Shell Script. Por ejemplo, existe un módulo para dar soporte a C/C++. Existen módulos para añadir un poco de todo, desde Telnet hasta soporte a bases de datos.

2.9.3 Lenguaje de programación: C++

El C++ es un lenguaje de programación, diseñado a mediados de los años 1980, como extensión del lenguaje de programación C (Stroustrup, 1998).

Las principales características del C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica (templates).

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel (Stroustrup, 1998):

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución.

C++ está considerado como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, “sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su trabajo” (Stroustrup, 1998).

Capítulo 3 Descripción y Aplicación de la solución propuesta

3.1 Introducción

La aplicación de técnicas de descripción formal al campo de la seguridad debería contemplar la necesidad de especificar adecuadamente los intercambios que corresponden a los protocolos de seguridad, así como las funciones y mecanismos implicados con el objetivo de ser interpretadas en fase de implementación. Esta idea abre múltiples posibilidades en el marco de la evaluación de protocolos de seguridad.

En este capítulo se adentra en el Middleware del SCADA “Guardián del ALBA” mostrando las deficiencias que este Middleware presenta. Se ofrecen detalles de la solución que se logró. Además, se brindan algunos conceptos que son necesarios para entender la solución expuesta a lo largo del capítulo.

3.2 Brechas de seguridad

Una brecha puede ser cualquier abertura hecha en una pared o edificio. También se le puede llamar brecha a iniciar una vía nueva de hacer algo en específico. En el presente trabajo se hace referencia a brechas de seguridad como nuevas vías de ataque o aberturas creadas, dando la posibilidad de atacar de forma directa o indirecta el sistema.

El Middleware ofrece un conjunto de servicios que hacen posible el funcionamiento del SCADA en general. Dichos servicios pueden convertirse en puntos de ataque para aquellos usuarios o sistemas que intenten acceder a las prestaciones que este brinda. Las brechas de seguridad se crean cuando el sistema se dispone a transmitir o recibir información y/o hacer llamadas a procedimientos remotos.

La tecnología que se usó para implementar el middleware proporciona varios servicios necesarios para la comunicación entre procesos en tiempo real. Dentro de los más usados se encuentra el Notify_Service (Servicio de Notificación) y el NameService (Servicio de Nombres).

El Servicio de Notificación permite la creación de canales para el intercambio de información.

Mediante el canal creado la transmisión de información se hará de forma desacoplada pudiéndose enviar a través de él datos estructurados, secuenciales o de cualquier tipo (Any). Para este intercambio se identifican consumidores y proveedores que con anterioridad se conectaron al canal a través de un Proxy apropiado y comenzarán a transmitir o recibir según corresponda cuando la conexión sea establecida.

Otro de los servicios que se usa para lograr una mayor interoperabilidad es el Servicio de Nombres, que no está exento de ser atacado. En este se proporciona una representación única de un nombre en específico. Este servicio es necesario para la comunicación sincrónica y la combinación de ellas, para las interacciones donde sea necesaria la comunicación asincrónica.

Las brechas de seguridad que se identificaron para el middleware del Guardián del ALBA son las siguientes:

1. El Servicio de Notificación no cuenta con un mecanismo de autenticación de proceso para la creación de canales de comunicación, lo que provoca que agentes no autorizados puedan crear un proxy falso y asociarse a un canal de comunicación. Un agente no autorizado puede convertirse en un proveedor y podrá enviar datos falsos sobrecargando aquel consumidor que esté solicitando dicho dato. También puede darse el caso que se haga con el objetivo de lograr un mal funcionamiento del sistema haciéndole llegar información falsa.
2. El Servicio de Nombre no cuenta con un mecanismo de autenticación de proceso para la escritura/lectura de datos sobre él, lo que provoca que cualquier sistema o agente externo puede escribir, leer, o modificar la información que en él persiste.
3. Las invocaciones a procedimientos remotos (Seguridad, Configuración y de consultas a la base de datos histórica) realizadas por objetos que emplean comunicación sincrónicas no están protegidas.
4. La transmisión de datos se hace en texto plano, lo que puede traer consigo que la información que se transmite sea leída o no sea auténtica, o sea, el contenido del mensaje enviado haya sido modificado o leído en su tránsito debido a la fácil lectura e interpretación de la información que viaja a través del middleware. Puede darse el caso de ser interceptado por un agente externo al sistema con el fin de obtener o corromper la información comprometiendo la funcionalidad del sistema.

Se puede afirmar entonces que en el middleware en cuestión se violan tres de los servicios de seguridad más importantes: la autenticidad, la confidencialidad y la integridad de los datos. Con la ausencia de estos no es segura la transmisión de información.

Para proteger al sistema se realiza un análisis de las amenazas potenciales que puede sufrir, las pérdidas que podrían generar, y la probabilidad de su ocurrencia.

3.3 Modo de Transmisión

En el Middleware se usan dos arquitecturas para la transmisión de datos, la transmisión sincrónica, y la transmisión asincrónica. La sincrónica se caracteriza por ser una técnica que consiste en el envío de una trama de datos que configura un bloque de información comenzando con un conjunto de bits de sincronismo (syn) y termina con otro conjunto de bits de final de bloque (etb). En este caso, los bits de sincronismo tienen la función de sincronizar los relojes existentes tanto en el emisor como en el receptor, de forma tal que estos controlan la duración de cada bit y carácter.

Por otro lado, la transmisión asincrónica tiene lugar cuando el proceso de sincronización entre emisor y receptor se realiza en cada palabra de código transmitido. Esta sincronización se lleva a cabo a través de unos bits especiales que definen el entorno de cada código. También se establece una relación asincrónica cuando no hay ninguna relación temporal entre la estación que transmite y la que recibe. Es decir, el ritmo de presentación de la información al destino no tiene por qué coincidir con el ritmo de presentación de la información por la fuente. En estas situaciones tampoco se necesita garantizar un ancho de banda determinado, suministrando solamente el que esté en ese momento disponible.

En el middleware para lograr una trasmisión asíncrona se implemento una función que va guardando toda la información que se desea trasmitir. Existe un hilo implementado que se va encargar de ir pasando esa información al canal donde cada uno de los receptores asociados a ese canal ira recibiendo dicha información. De este modo se logra un desacoplamiento entre los receptores y los emisores que están transmitiendo o recibiendo respectivamente.

3.4 Propuesta de solución del problema.

Las propuestas de solución que se presenta fueron tomadas de las funcionalidades que brinda TAO. Se toman los servicios que nos ofrece la implementación del módulo de seguridad para un mejor desarrollo del proyecto. Todas las propuestas seleccionadas favorecen a la seguridad del Middleware aunque no todas han sido escogidas para darle la solución final al problema presentado.

3.4.1 Mecanismo de seguridad propuesto.

Parece claro que, aunque los tres tipos de mecanismos son importantes para la seguridad de un sistema, se enfatiza en el uso de mecanismos de prevención. Si se consiguiera un sistema sin vulnerabilidades y cuya política de seguridad se implementará mediante mecanismos de prevención de una forma completa, no se necesitarían mecanismos de detección o recuperación. Aunque estos son difíciles de conseguir en la práctica, serán los mecanismos de prevención, los propuestos para el presente trabajo dada su importancia y potencialidad, siendo también los más empleados en la actualidad.

3.4.2 Políticas de calidad de servicios del ORB.

El ORB es un gestor de objetos que forma el núcleo de CORBA. El mismo hace que la comunicación entre Cliente y Servidor sea independiente del lenguaje de programación que se utiliza, de la ubicación de los elementos y de los aspectos relacionados con los Sistemas Operativos. El ORB es el encargado de encontrar las implementaciones de los objetos remotos devolviendo una referencia al objeto para poder hacer uso de sus funcionalidades encaminando las llamadas a los métodos y las respuestas correspondientes. En él se implementan diversas funcionalidades que hacen a las aplicaciones CORBA interoperables. Estas funcionalidades también abarcan los temas de calidad de servicios (QoS) llegando a implementar funciones que permiten crear políticas de calidad de servicios. A través de ella se pueden definir las que se van a usar para implementaciones específicas, ya sea del lado del cliente o del lado del servidor.

La implementación de seguridad que brinda TAO define varias políticas de seguridad. Las políticas, como integridad y confidencialidad, garantizan la protección de mensajes a transmitir entre ambos lados. Haciendo uso de ellas se asegura que la transmisión de datos entre las interfaces sincrónicas se haga de forma segura. El ORB que esté en uso en cualquiera de los lados puede crear estas políticas, luego puede modificar su listado de políticas y transmitir de forma segura.

Esta propuesta trae como inconveniente que solo se puede implementar para comunicaciones sincrónicas, o sea, comunicaciones Cliente/Servidor. Este tipo de comunicación solo es utilizada con determinadas interfaces lo que trae consigo que no se pueda tomar como una solución para la seguridad del Middleware, esta solución solo garantiza la seguridad en la transmisión de información para estas interfaces. Otro de los inconvenientes y como principal problema es que afecta en gran medida a la comunicación en tiempo real, retrasando el tiempo de transmisión de información en valores significativos para la principal funcionalidad del Middleware en el SCADA. Cuando se hace uso de dichas políticas de calidad de servicio se debe tener en cuenta que hay que implementarlas tanto del lado del cliente como del lado del servidor, dato que afectaría a la

hora de crear objetos remotos para la comunicación entre los componentes implicados en el intercambio de información.

3.4.3 Usar de certificados para el intercambio de identidad.

SSL proporciona autenticación y privacidad de la información entre extremos mediante el uso de criptografía. TAO soporta comunicaciones confidenciales haciendo uso del protocolo IIOP que se implementó sobre el protocolo SSL, creando un nuevo protocolo para la transmisión de datos de forma segura, llamado SSLIOP. Usando el protocolo SSLIOP para la comunicación entre los ORB es posible asegurarse de que todas las invocaciones de métodos remotos entre los ORBs sean seguras. La implementación de IIOP sobre SSL hace permisible usar las funcionalidades que caracterizan al protocolo SSL, como son la confidencialidad, la integridad de la información, entre otras. También es posible el control de acceso usando certificados basado en X.509 para el intercambio de identidades.

La autenticación mutua requiere un despliegue de infraestructura de claves públicas para los clientes. El protocolo permite a las aplicaciones comunicarse de una forma diseñada para prevenir escuchas, la falsificación de la identidad del remitente y mantener la integridad del mensaje. Para la creación de los certificados y de las llaves se usa OPENSSL. El mismo es un proyecto de software que es desarrollado por los miembros de la comunidad de código libre y consiste en un robusto paquete de herramientas de administración y librerías relacionadas con la criptografía que suministra funciones criptográficas para encriptar la información que viaja por la red, y la generación de certificados con claves mayores de 1024 byte.

3.4.3.1 Fichero de configuración para TAO

TAO implementa SSL como un protocolo conectable que puede ser cargado de forma dinámica por el ORB. Para esto se hace uso de un fichero de configuración (svc.conf). El mismo cargará el protocolo SSLIOP desde una librería llamada TAO_SSL. El protocolo SSLIOP tiene un número de opciones que se describen a continuación.

Opciones de SSLIOP

SSLNoProtection:

Del lado de cliente esta opción exige a las peticiones el uso del protocolo estándar IIOP. Del lado del servidor usar esta opción permite peticiones al servidor para ser hechas a través del mismo protocolo. Las peticiones a través de SSL pueden ser hechas aún. Esta opción será desechada en el Nivel 2 (SecurityLevel2).

SSLCertificate FORMAT: filename:

Modifica el nombre del fichero que contiene el certificado para este proceso. El fichero puede estar en el formato PEM (Privacy Enhanced Mai) o ASN.1 (ASN1). Recordar que los certificados pueden ser señalados por una CA (Certificate Authority – Autoridad de Certificación) reconocido por el cliente.

SSLPrivateKey FORMAT: filename:

Modifica el nombre del fichero que contiene la llave privada para este proceso. La llave privada y el certificado pueden estar enlazados. Es importante que se asegure la llave privada, por defecto el OpenSSl generará ficheros con llaves privadas protegidas. La contraseña será preguntada cuando se corra una aplicación CORBA.

SSLAuthenticate which:

Controla en nivel de autenticación. Los argumentos pueden ser:

NONE, SERVER, CLIENT o SERVER_AND_CLIENT. Debido a limitaciones en el protocolo SSL CLIENT implica que el SERVER sea autenticado también.

SSLAcceptTimeout which:

Modifica la máxima cantidad de tiempo a permitir para establecer una conexión SSL/TLS pasiva. Su valor por defecto son 10 segundos.

3.5 Solución del problema

Atendiendo a las características de SSL y IIOP analizadas en el apartado anterior y que TAO implementa una combinación de estas, el protocolo SSLIOP, entonces se decide utilizar este mecanismo para lograr comunicaciones seguras entre los módulos del SCADA Guardián del ALBA.

Para poder conectarse utilizando el protocolo SSLIOP se requiere de una llave y un certificado digital que debieron ser dados por una entidad certificadora calificada, o como también es llamada, autoridad de certificación (CA).

3.5.1 Uso de certificados para el intercambio de identidad.

Teniendo en cuenta que SSL proporciona autenticación y privacidad de la información entre ambos extremos mediante el uso de criptografía y que TAO implementa un protocolo capaz de hacer uso de él se toman estas características para alcanzar la solución.

3.5.1.1 Uso de las autoridades de certificación

CA se encargará de firmar y revocar todos los certificados en el sistema. La autoridad de certificación le dará validez al certificado mediante la firma digital de este con su clave privada. Se confía en ella para el buen funcionamiento del servicio. La autoridad de certificación que se usa es gratuita, aunque esa decisión pueda variar. No obstante cualquiera que fuese la decisión tomada no complejizaría para nada la solución muestra. La autoridades de certificación emite los certificados a fin de proveer confianza sobre la autenticidad de las claves públicas de los demás o que acrediten que una clave pública es realmente del proceso que se le atribuye y que todavía está vigente en el sistema. Esta CA, por sí misma o mediante la intervención de una Autoridad de Registro (RA), verifica la identidad del solicitante de un certificado digital antes de su expedición o, en caso de certificados expedidos con la condición de revocados, elimina la revocación de los certificados al comprobar su identidad.

3.5.1.2 Uso de los certificados digitales

Los certificados digitales atestiguan que la clave pública pertenece al proceso o entidad que intenta recibir o enviar. A su vez, se evita que cualquiera pueda generar un clave distinta y puede hacerse pasar por cualquier otra persona, de no ser las informaciones que van asociadas a la clave pública, la autoridad de certificación se negará a emitir el correspondiente certificado, con lo que los demás implicados no la aceptarán como clave pública de confianza. Este certificado lo emite la misma autoridades de certificación y es un documento en el que se relacionan la identidad de su poseedor y la clave pública a la que se refiere, que han sido firmados digitalmente con la clave privada de dicha autoridad. Si un certificado es revocado se suspenderá.

Dado que la autoridad de certificación puede emitir diferentes tipos de certificados; para este trabajo se emitió un certificado de Identidad personal o digital. El certificado se emplea dentro del protocolo SSL para que las comunicaciones con el servidor se protejan con un cifrado de 128 bits.

3.5.1.3 Uso del fichero de configuración de TAO

Cargando el fichero svc.conf que nos brinda TAO conectamos el protocolo SSL con las opciones *SSLCertificate, SSLPrivateKey, SSLNoProtection o SSLAuthenticate* en dependencia de lo que se desea proteger. El ORB usará el protocolo para la transmisión de datos haciéndolo de forma segura.

Con esto se garantiza:

- El control de quien puede o no ver la información que se trasmite, también mantenemos las transacciones privadas e inviolables en el sentido de que entidades no autorizadas no puedan descifrar el contenido de los mensajes.

- La seguridad de la información almacenada o transmitida no están alteradas, se asegurara que las transmisiones de información entre los ORB no son alteradas o interferidas.
- La habilidad de determinar la identidad de las partes que se comunican.
- No es posible que un emisor de un mensaje pueda alegar que no envió una comunicación segura o que no realizo una petición.

3.6 Modo de Solución

3.6.1 Creación de la entidad certificadora

Paso 1:

yusniel@middleware03:~$ mkdir CA

yusniel@middleware03:~$ ~/CA$ ./CA.pl -newca

CA certificate filename (or enter to create) ENTER

Making CA certificate ...

Generating a 2048 bit RSA private key

.....+++

.....+++

writing new private key to './demoCA/private/cakey.pem'

Enter PEM pass phrase:middlewaresecurity

Verifying - Enter PEM pass phrase:middlewaresecurity

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [CU]:

State or Province Name (full name) [Villa Clara]:

Locality Name (eg, city) [Santa Clara]:

Organization Name (eg, company) [Universidad de las Ciencias Informáticas UCI]:

Second Organization Name (eg, company) [PDVSA]:

Organizational Unit Name (eg, section) [UCI]:

Common Name (eg, YOUR name) [Yusniel Cárdenas del Valle]:

Email Address [yusniel.cu@gmail.com]:

yusniel@middleware03:~$ ~/CA$

Paso 2:

yusniel@middleware03:~/CA$ ./CA.pl -newreq

Generating a 2048 bit RSA private key

.....+++

.....+++

writing new private key to 'newreq.pem'

Enter PEM pass phrase:

middlewaresecurity

Verifying - Enter PEM pass phrase:

middlewaresecurity

You are about to be asked to enter information that will be incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [CU]:

State or Province Name (full name) [Villa Clara]:

Locality Name (eg, city) [Santa Clara]:

Organization Name (eg, company) [Universidad de las Ciencias Informáticas UCI]:

Second Organization Name (eg, company) [PDVSA]:

Organizational Unit Name (eg, section) [UCI]:

Common Name (eg, YOUR name) [Yusniel Cárdenas del Valle]:

Email Address [yusniel.cu@gmail.com]:

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:taosecurity

Request (and private key) is in newreq.pem

yusniel@middleware03:~/CA$

Paso 3:

yusniel@middleware03:~/CA$ ./CA.pl -sign

Using configuration from openssl.cnf

Enter pass phrase for ./demoCA/private/cakey.pem:

Check that the request matches the signature

Signature ok

Certificate Details:

Serial Number:

87:78:fb:11:67:df:29:80

Validity

Not Before: Apr 15 09:23:46 2008 GMT

Not After : Apr 15 09:23:46 2009 GMT

Subject:

countryName = CU

stateOrProvinceName = Villa Clara

localityName = Santa Clara

organizationName = Universidad de las Ciencias Informáticas UCI

organizationName = PDVSA

organizationalUnitName = UCI

commonName = Yusniel Cárdenas del Valle

emailAddress = yusniel.cu@gmail.com

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

Netscape Comment:

OpenSSL Generated Certificate

X509v3 Subject Key Identifier:

C9:72:AF:BD:52:51:A2:87:01:AC:E0:09:2B:98:43:2D:AC:97:19:5A

X509v3 Authority Key Identifier:

keyid:B9:4C:FD:68:80:8E:9C:71:59:BC:A8:74:2B:6F:59:A8:B2:76:9F:23

DirName:/C=CU/ST=Villa Clara/L=Santa Clara/O=Universidad de las Ciencias Informáticas UCI/O=PDVSA/OU=UCI/CN=Yusniel Cárdenas del Valle/emailAddress=yusniel.cu@gmail.com

serial:87:78:FB:11:67:DF:29:7F

Certificate is to be certified until Apr 15 09:23:46 2009 GMT (365 days)

Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y

Write out database with 1 new entries

Data Base Updated

Signed certificate is in newcert.pem

yusniel@middleware03:~/CA$

Paso 4:

Ahora necesitamos quitarle el pass phrase porque cada vez que un cliente se vaya a conectar no debe pedirlo.

yusniel@middleware03:*~/CA$ openssl rsa -in client-key.pem -out server-key.pem*

Ya tenemos todos lo que necesitamos, nuestra entidad certificadora en .../CA/***cacert.pem*** , la llave y los certificados para cada cliente en ./***server-key.pem***, entiéndase por cliente a: un cliente , un servidor, un consumidor, un proveedor, el Servicio de Nombre, el Servicio de Notificación o el ChannelRegistryService.

3.6.2 Ejecución de los Servicios

- Configuración de las variables de entorno

- yusniel@middleware03:~/CA$ export OPENSSL_CONF=openssl.cnf
- yusniel@middleware03:~/CA$ export SSL_CERT_DIR=/home/ yusniel /CA/certs/
- yusniel@middleware03:~/CA$ export SSL_CERT_FILE=/home/ yusniel /CA/cacert.pem

Es necesario configurar estas variables en la computadora donde se va a ejecutar el Servicio de Nombres.

- Ejecución del Servicio de Nombres

- yusniel@middleware03:~$ cd CA/NS
- yusniel@middleware03:~/CA/NS$ $TAO_ROOT/orbsvcs/Naming_Service/Naming_Service -ORBEndpoint iiop://192.168.10.103/ssl_port=12345 -ORBSvcConf ns.conf

En el fichero ns.conf esta la configuración del Servicio de Nombres, 192.168.10.103 es el IP de la máquina y 12345 es el puerto por el que va a escuchar el Servicio de Nombres. Es un fichero como el que hacíamos mención con anterioridad (svc.conf).

- Ejecución del Servicio de Notificación.

- yusniel@middleware03:~$ cd CA/Notify
- yusniel@middleware03:~/CA/Notify$./$TAO_ROOT/orbsvcs/Notify_Service/Notify_Service -RBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBListenEndpoints iiop://192.168.10.103/ssl_port=9999 -ORBSvcConf notify.conf

En el fichero **notify.conf** esta la configuración del Servicio de Notificación y va a escuchar por el puerto 9999.

- Ejecución de un servidor

- yusniel@middleware03:~$ cd CA/Syncware/SecurityServerTest/Debug
- yusniel@middleware03:~/CA/Syncware/SecurityServerTest/Debug$ ./SecurityServerTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf server.conf

  En el fichero **server.conf** está la configuración del servidor.

- Ejecución de un cliente

- yusniel@middleware03:~$ cd CA/Syncware/SecurityClientTest/Debug
- yusniel@middleware03:~/CA/Syncware/SecurityClientTest/Debug$ ./SecurityClientTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf client.conf

En el fichero **client.conf** está la configuración del cliente.

3.7 Conclusiones

En el presente capítulo se describen las vulnerabilidades del Middleware en cuanto a la seguridad, se propone una vía de solución para garantizar comunicaciones seguras y fiables en el Middleware. Finalmente se obtuvo un resultado, conformada por mecanismos de seguridad, con las exigencias necesarias para incorporar al Middleware del “Guardián del ALBA”.

Conclusiones

La solución propuesta responde a la situación polémica que se originó al presentar un middleware despojado de mecanismos de seguridad necesarios para garantizar la seguridad de la información que viaja a través de él. Se cumplieron las tareas que se trazaron para la realización de la investigación. Por tanto podemos arribar a las siguientes conclusiones:

- Se incorporaron los mecanismos de seguridad al middleware para el SCADA “Guardián del ALBA”.
- Como resultado se obtuvo un middleware seguro y eficiente, listo para ser explotado en el SCADA “Guardián del ALBA” con mecanismos de seguridad incorporados.
- Con la incorporación de los mecanismos de seguridad propuesto se hace posible la trasmisión de datos de forma segura en un tiempo lo más real posible.

Recomendaciones

Se recomienda:

- Realizar pruebas más estrictas a la comunicación asincrónica usando protección SSL, para determinar la influencia de SSL en el rendimiento del Middleware.
- Probar otros protocolos que brinden seguridad en tecnologías de transmisión de datos en tiempo real para determinar si existe alguno con mejores prestaciones que SSLIOP, para implementarlo en la versión 2.0 del Middleware.

Bibliografía

1. **2004.** Mecanismos de Seguridad e Integridad. [En línea] 2004. [Citado el: 7 de febrero de 2008.] http://alfa.facyt.uc.edu.ve/computacion/pensum/cs0347/clases/Mecanismos%20de%20seguridad%20e%20integridad%20-%20clases.pdf.
2. **Alfinal.com. 2000.** Sistemas SCADA.Fundamento teorico. [En línea] 2000. [Citado el: 10 de febrero de 2008.] :http://www.alfinal.com/Temas/sistemascada.shtml.
3. **Anderson, James P. 1980.** Computer Security Threat Monitoring and Surveillance.Technical report. [En línea] abril de 1980. [Citado el: 25 de mayo de 2008.]
4. **Así-Red Servicios Telemáticos, S.L.L. 2000.** Middleware XML. [En línea] Enero de 2000. [Citado el: 18 de enero de 2008.] http://www.asired.es/pdf/asired_middleware_xml_web.pdf.
5. **Autores, Colectivo de. 2004.** DERMI: Middleware para aplicaciones de trabajo en grupo descentralizadas . [En línea] diciembre de 2004. [Citado el: 18 de enero de 2008.] http://www.rediris.es/rediris/boletin/66-67/ponencia12.pdf..
6. **Badani, Sigifredo E. 2001.** Arquitectura. Herramientas de desarrollo. [En línea] septiembre de 2001. [Citado el: 5 de mayo de 2008.] http://eiec.ucentral.cl/ftp/material/apuntes/iec61/Arquitectura/3%20Arquitectura.PDF.
7. **Bannan, Joan. 2001.** Intranet Document Management. Intranet Design Magazine. [En línea] 2001. [Citado el: 25 de mayo de 2008.] http://idm.internet.com/features/docmgmt9a-1.shtml.
8. **BEA Systems, Inc. 2001.** Overview of the CORBA Security Feature. [En línea] 2001. [Citado el: 20 de febrero de 2008.] http://edocs.bea.com/tuxedo/tux80/security/overview.htm.
9. **Berry, Bob. 2005.** SCADA Tutorial: A Fast Introduction to SCADA. Fundamentals and Implementation. [En línea] agosto de 2005. [Citado el: 13 de diciembre de 2007.] http://www.dpstele.com/pdfs/white_papers/scada.pdf .
10. **Borges, Luis E. 2000.** Sistemas de lectura remota del consumo. *Revista Electronica.* [En línea] 2000. [Citado el: 10 de diciembre de 2007.] http://neutron.ing.ucv.ve/revista-e/No7/.
11. **Bray, Martin M. 1997.** Middleware. [En línea] 1997. [Citado el: 3 de diciembre de 2007.] http://www.sei.cmu.edu/str/descriptions/middleware.html .

12. **Carrasco, Eddy H. y Lopez, Roberto. 2001.** Aspectos de Seguridad en el Comercio Electrónico. [En línea] diciembre de 2001. [Citado el: 21 de mayo de 2008.] htpp://espejos.unesco.org.uy/simplac2002/Ponencias/Segurm%E1tica/VIR006.doc .
13. **Chris Cleeland, Rob Martin. 2006.** CORBA Security: An Overview. . [En línea] 2006. [Citado el: 18 de febrero de 2008.] http://www.ociweb.com/cnb/CORBANewsBrief-200205.html.
14. **Colectivo, de Autores. 2007.** *Introduction to the New Mainframe: Security.* New York : redbook, 2007.
15. **Creus, A Soler. 2005.** *Fiabilidad y seguridad: su aplicación en procesos industriales.* Barcelona, España : Marcombo, ediciones tecnicas, 2005.
16. **Curiel, Mariela. 2004.** Sistemas Distribuidos. [En línea] 2004. http://www.ldc.usb.ve/~mcuriel/Cursos/sop3/Tema1.pdf .
17. **Daly, P.G. 2001.** "P.G. Daly's Intranet Talk: How Others Create And Manage Intranet Content. [En línea] 2001. [Citado el: 6 de junio de 2008.] http://idm.internet.com/articles/200004/ia_04_29_00a.html.
18. **Facultad de Ciencias Exactas y Naturales y Agrimensuras 2002.** Seguridad en Sistema Distribuidos. [En línea] 2002. [Citado el: 5 de mayo de 2008.] http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/SOMA.htm.
19. **D'Sousa, C. 2005.** Sistemas de Control. [En línea] 2005. [Citado el: 10 de diciembre de 2008.] http://www.monografia.com/trabajos11/sisco/sisco.shtml.
20. **eProsima. 2007.** RTI Data Distribution Service. [En línea] 2007. [Citado el: 10 de mayo de 2008.] http://www.eprosima.com/dnnEprosima/Default.aspx?tabid=62#middlewareadecuado.
21. **Facultad, de Ciencias Exactas y Naturales y Agrimensuras. 2003.** Seguridad. [En línea] 2003. http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/MonogSO2003/SeguridadenlosSistemasDistribuidos.zip.
22. **—. 2001.** Seguridad en Unix. [En línea] 2001. [Citado el: 5 de mayo de 2008.] http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/SEGUNIX01.htm#introducci%F3n .
23. **Facultad, de Informática Universidad Politécnica de Madrid. 2003.** Autoridades de Certificación y Confianza Digital. [En línea] septiembre de 2003. [Citado el: 15 de mayo de

2008.] http://tirnanog.ls.fi.upm.es/CriptoLab/Biblioteca/Conferencias/ICAI97_mem.pdf .

24. **Ferrari, Pablo J. 2005.** Monografias sobre sistemas de control distribuidos. [En línea] 2005. [Citado el: 10 de diciembre de 2007.] http://www.monografias.com/trabajos-pdf/sistemas-de-control-distribuido/sistemas-de-control-distribuido.pdf..
25. **Gall, N. 2002.** The Origin(Coining) of the Middleware. [En línea] november de 2002. [Citado el: 14 de diciembre de 2007.] http://radio.weblogs.com/0126951/2003/11/02.html#a72.
26. **2004.** Gestion de seguridad en IRISGrid. [En línea] 2004. [Citado el: 6 de febrero de 2008.] http://www.irisgrid.es/testbed/seguridad.es.html.
27. **Ghosh, Anup K. 1998.** E-Commerce security: Weak Links. [En línea] 1998. [Citado el: 6 de junio de 2008.]
28. **Hernández, Raul Alberto Burguete. 2007.** ALGORITMOS DISTRIBUIDOS AUTOESTABILIZANTES APLICADOS EN LA SEGURIDAD COMPUTACIONAL. [En línea] 2007. [Citado el: 1 de abril de 2008.] http://webdia.cem.itesm.mx/ac/rogomez/Tesis/presRaul.ppt .
29. **Hostalot, R. 2005.** El Middleware. [En línea] 2005. [Citado el: 17 de noviembre de 2007.] http://www.rifd-magazine.com.
30. **Huerta, Antonio. 2002.** Seguridad en Unix y Redes. [En línea] julio de 2002. [Citado el: 18 de febrero de 2008.] http://www.ibiblio.org/pub/linux/docs/LuCaS/Manuales-LuCAS/doc-unixsec/unixsec-html/.
31. **IBLNEWS, A. 2007.** Cuba y Venezuela crean software buscando sobreranía informática. [En línea] febrero de 2007. [Citado el: 20 de noviembre de 2007.] http://notiblog-hugorueda.blogspot.com/2007/02/cuba-y-venezuela-crean-software.html.
32. **Iliana Perez, Irina Argota. 2007.** *Desarrollo del flujo de requisitos para el subsistema de Gráficos Vectoriales del producto SCADA nacional.* 2007.
33. **INEI, G. P. 1997.** *La importancia de un Middleware Robusto y Escalable en las Soluciones Empresariales Cliente/Servidor.* [En línea] 1997. [Citado el: 28 de noviembre de 2007.] http://www.inei.gob.pe/web/metodologias/attach/lib616/CAP0307.HTM.
34. **Kalakota, Ravi y Whinston, Andrew B. 1997.** Electronic Commerce, A Managerr's Guide. [En línea] 1997. [Citado el: 12 de junio de 2008.]
35. **Klemencic, Joe. 2001.** Basic Security Mechanisms for Wireless Networks. [En línea] julio de 2001. [Citado el: 8 de enero de 2008.]
36. **Krakowiak, S. 2005.** What's middleware? . [En línea] 2005. [Citado el: 14 de diciembre de

2007.] http://middleware.objectweb.org/.

37. **Lancharro, Javier F. 2004.** Firewall, la muralla defensiva de un PC.Diferentes tipos de intrusiones o ataques. . [En línea] febrero de 2004. [Citado el: 7 de febrero de 2008.] http://www.terra.es/tecnologia/articulo/html/tec10590.htm.

38. **López, Lourdes y Portillo, Eloy. 2007.** Seguridad en redes telemáticasParte I: La problemática de la seguridad. [En línea] 2007. [Citado el: 21 de mayo de 2008.] http://www.rediris.es/rediris/boletin/31/enfoque1.html.

39. **Marañon, Gonzalo Alvarez. 2000.** Criptología y seguridad. [En línea] 2000. [Citado el: 6 de febrero de 2008.] http://www.iec.csic.es/CRIPTonOMICon/seguridad/.

40. **Martínez, Javier Hernández. 2002.** Derecho de las Nuevas Tecnologías. Firma Digital. [En línea] 2002. [Citado el: 7 de abril de 2008.] http://www.tuguialegal.com/firmadigital.htm .

41. **Matín, C.R. 2006.** Oracle certifica Siebel para Oracle Fusion Middleware . [En línea] junio de 2006. [Citado el: 10 de mayo de 2008.] http://www.todosobrecrm.com/noticias/2006/06/oracle-certifica-siebel-para-oracle-fusion-middleware/.

42. **Miller, Darren. 2006.** Es tu cifrado de datos realmente segurol. [En línea] marzo de 2006. [Citado el: 7 de abril de 2008.] http://www.articleset.com/Computadoras-e-Internet_articles_es_Es-tu-cifrado-de-datos-realmente-seguro.htm .

43. **Molina, Hernán D. 2006.** Firewalls Distribuidos. [En línea] 2006. [Citado el: 5 de mayo de 2008.] http://www.textoscientificos.com/redes/firewalls-distribuidos .

44. **Mortera, Pablo. 2002 .** .Introducción a la Java Message Service (JMS). [En línea] 2002 . [Citado el: 6 de mayo de 2008.] http://personales.ya.com/pmortera/Pagina_web_jms/jms_parte.doc .

45. **Palacio, David Melendi. 2007.** .Principios para la Integración. Aplicaciones Telemáticas. [En línea] octubre de 2007. [Citado el: 5 de mayo de 2008.] .http://www.it.uniovi.es/docencia/Telematica/appTel/material/Tema3.pdf .

46. **Palacios, Juan Carlos y Salazar, Hector. 2000.** Marco Teórico del Comercio Electrónico (Modelo Negocio-Consumidor):Propuesta para el desarrollo de sitio para venta de contenido. [En línea] 2000. [Citado el: 14 de mayo de 2008.] http://espejos.unesco.org.uy/simplac2002/Ponencias/Segurm%E1tica/VIR006.doc.

47. **Universidad Nacional de la Plata. 2007.** Arquitectura Orientada a Servicios . [En línea] 2007. [Citado el: 3 de febrero de 2008.] http://www.linti.unlp.edu.ar/tiki-

download_file.php?fileId=554.

48. **Rojo, J. Oscar. 2003.** Introducción a los sistemas distribuidos. [En línea] 2003. [Citado el: 31 de marzo de 2008.] http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos.

49. **Rosales, Ing. Rubén Borja. 2001.** .SEGURIDAD Y DINERO ELECTRÓNICO EN LA ERA DIGITAL. [En línea] 2001. [Citado el: 7 de abril de 2008.] .http://intrauni.uni.edu.pe/intrauni8/pag12.htm .

50. **Rubio, Gabriel Buades. 2002.** Sistemas Distribuidos. [En línea] ebrero de 2002. [Citado el: 21 de marzo de 2008.] .http://dmi.uib.es/~bbuades/sistdistr/index.htm.

51. **Sallis, Ezequiel. 2006.** Dispositivos RFid, nuevas tecnologías y nuevos ataques. [En línea] 2006. [Citado el: 27 de febrero de 2008.] http://www.infosecurityonline.org/newsletter/octubre/hacking.htm.

52. **Salvador, Enrique Soriano. 2006.** TESIS DOCTORAL.Arquitecturas de Seguridad para Sistemas Distribuidos Fraccionables, Dinámicos y Heterogéneos con Aplicación a la Computación Ubicua. [En línea] julio de 2006. [Citado el: 16 de mayo de 2008.] http://pqxx.org/development/libpqxx/ .

53. **Seguridad, Digital. 2006.** Definiciones, amenazas y mecanismos de seguridad. [En línea] 2006. [Citado el: 6 de febrero de 2008.] http://www.seguridaddigital.info/index.php?option=com_content&task=view&id=83&Itemid=26.

54. **Sepulveda, Daniel. 2001.** Protocolos Seguros para el Web. [En línea] 2001. [Citado el: 10 de abril de 2008.] .http://www.tejedoresdelweb.com/307/article-5670.html .

55. **Sierra, Javier Emilio. 1997.** Protocolo de seguridad Wep. [En línea] 1997. [Citado el: 16 de abril de 2008.] .http://www.monografias.com/trabajos18/protocolo-wep/protocolo-wep.shtml .

56. **Specification, OMG Segurity. 2007.** .CORBA Security Service. [En línea] diciembre de 2007. [Citado el: 18 de febrero de 2008.] http://www.omg.org/technology/documents/formal/omg_security.htm#Security_Service.

57. **Technologies, Zebra. 2004.** Opciones de Impresión de Códigos de Barras para Impresoras Zebra con Oracle WMS y MSCA. [En línea] 2004. [Citado el: 18 de enero de 2008.] http://www.zebra.com/.../white_papers__microsites/spanish_la_microsite/oracle.File.tmp/Oracle_Spa.pdf .

58. **Teleformacion.** [En línea] http://teleformacion.uci.cu.

59. **Telemetry, ADCON. 2001.** Addvantage Pro 5 . [En línea] 2001. [Citado el: 18 de febrero de 2008.] .http://www.adcon.at/english/produkte_software_Addvantage_Pro_5_en.htm.

60. **Telleria, Ana Silvia. 2007.** *Propuesta para el diseño de un middleware para el SCADA-PDVSA.* 2007.

61. **Universidad, de Murcia. 2003.** Autoridades de Certificación. [En línea] 2003. [Citado el: 15 de mayo de 2008.] http://www.um.es/ssl/PKI/ca.html.

62. **Uruguay, Facultad de Ciencias Económicas y de Administración de. 2006.** Sistemas computacionales. [En línea] Octubre de 2006. [Citado el: 14 de diciembre de 2007.] http://www.ccee.edu.uy/ensenian/catsistc/docs/bol1.pdf .

63. **X-Force. 2002.** Authentication and Security Mechanisms in ASP.NET Web Applications. [En línea] 2002. [Citado el: 7 de enero de 2008.] .http://documents.iss.net/whitepapers/asp_net_whitepaper.pdf.

64. **Yih-Chun Hu, Adrian Perrig, David B. Johnson.** Efficient Security Mechanisms for Routing Protocols. [En línea] [Citado el: 7 de enero de 2008.] .http://www.monarch.cs.rice.edu/monarch-papers/ndss03rev.pdf.

Anexos

A1

Pruebas de la parte sincrónica integrado con seguridad del Middleware del SCADA ¨Guardián del ALBA¨ utilizando SSLIOP como protocolo de comunicación.

Características de la PC donde se realizaran las pruebas:
Laptop Toshiba Satellite A135-S4407 procesador Core Duo T2350 a 1.73 GHz 1 gb de memoria RAM.

Para todas las pruebas se va a levantar el Name Service utilizando SSLIOP con certificados y llaves validas de la siguiente manera:

luis@luispc:~/entregable/NS$ $TAO_ROOT/orbsvcs/Naming_Service/Naming_Service -ORBEndpoint iiop://192.168.10.103/ssl_port=12345 -ORBSvcConf ns.conf

Prueba 1:

Se levanta el Servidor sin usar protocolo SSLIOP

luis@luispc:~/entregable/Syncware/SecurityServerTest/Debug$ ./SecurityServerTest -ORBInitRef NameService=iiop://192.168.10.103:12345/NameService

Cuando se corre el servidor de esta manera el server muestra el siguiente error:
server:Caught a CORBA::Exception: COMM_FAILURE (IDL:omg.org/CORBA/COMM_FAILURE:1.0)

y en el Name Service muestra este mensaje:

ACE_SSL (18549|3078059712) error code: 336027900 - error:140760FC:SSL routines:SSL23_GET_CLIENT_HELLO:unknown protocol

---

Prueba 2:

Se levanta el Servidor sin usar llaves ni certificados pero si protocolo SSLIOP, para esta prueba se utiliza server1.conf como fichero de configuración del servidor

luis@luispc:~/entregable/Syncware/SecurityServerTest/Debug$ ./SecurityServerTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf server1.conf

Cuando se corre el servidor de esta manera se muestra el siguiente mensaje de error:

ACE_SSL (20310|3079227600) error code: 336151568 - error:14094410:SSL routines:SSL3_READ_BYTES:sslv3

alert handshake failure

server:Caught a CORBA::Exception: TRANSIENT (IDL:omg.org/CORBA/TRANSIENT:1.0)

y el Name Service muestra el siguiente mensaje de error:
ACE_SSL (18549|3078059712) error code: 336105671 - error:140890C7:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:peer did not return a certificate

Prueba 3:

Se levanta el servidor usando SSLIOP, certificados y llaves generados utilizando otra Entidad Certificadora y como fichero de configuración server2.conf

luis@luispc:~/entregable/Syncware/SecurityServerTest/Debug$ ./SecurityServerTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf server2.conf

Cuando se corre el servidor de esta manera se muestra el siguiente mensaje de error:
ACE_SSL (20614|3078420688) error code: 336151576 - error:14094418:SSL routines:SSL3_READ_BYTES:tlsv1 alert unknown ca

server:Caught a CORBA::Exception: TRANSIENT (IDL:omg.org/CORBA/TRANSIENT:1.0)

y el Name Service muestra el siguiente mensaje de error:
ACE_SSL (18549|3078059712) error code: 336105650 - error:140890B2:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:no certificate returned

Prueba 4:

Se levanta el servidor usando SSLIOP y certificados y llaves correctas utilizando como fichero de configuración server.conf

luis@luispc:~/entregable/Syncware/SecurityServerTest/Debug$ ./SecurityServerTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf server.conf

Cuando se corre el servidor de esta manera se muestra que se conecta sin ningún problema.

A2:

A partir de estas pruebas se corre el Name Service de la siguiente manera:

luis@luispc:~/entregable/NS$ $TAO_ROOT/orbsvcs/Naming_Service/Naming_Service -ORBEndpoint iiop://192.168.10.103/ssl_port=12345 -ORBSvcConf ns.conf

y el servidor de esta forma:

luis@luispc:~/entregable/Syncware/SecurityServerTest/Debug$ ./SecurityServerTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf server.conf

Prueba 5 :

Se levanta el cliente sin usar protocolo SSLIOP

luis@luispc:~/entregable/Syncware/SecurityClientTest/Debug$ ./SecurityClientTest -ORBInitRef NameService=iiop://192.168.10.103:12345/NameService

El cliente se queda esperando un numero 0 , 1 o 2. Estas pruebas se harán presionando "1".
Al presionar "1" el cliente muestra el siguiente error:

terminate called after throwing an instance of 'CORBA::COMM_FAILURE'

Aborted

Y el Name Service muestra el siguiente mensaje de error:

ACE_SSL (6943|3077863104) error code: 336027900 - error:140760FC:SSL routines:SSL23_GET_CLIENT_HELLO:unknown protocol

---

Prueba 6

Se levanta el cliente usando como protocolo de comunicación SSLIOP pero sin llaves ni certificados utilizando client1.conf como fichero de configuración.

luis@luispc:~/entregable/Syncware/SecurityClientTest/Debug$ ./SecurityClientTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf client1.conf

se presiona "1" y el cliente muestra el siguiente mensaje de error:

ACE_SSL (21419|3078838480) error code: 336151568 - error:14094410:SSL routines:SSL3_READ_BYTES:sslv3 alert handshake failure

terminate called after throwing an instance of 'CORBA::TRANSIENT'

Aborted

Y el Name Service muestra el siguiente mensaje de error:

ACE_SSL (6943|3077863104) error code: 336105671 - error:140890C7:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:peer did not return a certificate

---

Prueba 7:

Se levanta el cliente utilizando como protocolo de comunicación SSLIOP y llaves y certificados generados por otra entidad certificadora y utilizando client2.conf como fichero de configuración.

luis@luispc:~/entregable/Syncware/SecurityClientTest/Debug$ ./SecurityClientTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf client2.conf

se presiona "1" y el cliente muestra el siguiente mensaje de error:

ACE_SSL (21562|3078445264) error code: 336151576 - error:14094418:SSL routines:SSL3_READ_BYTES:tlsv1 alert unknown ca

terminate called after throwing an instance of 'CORBA::TRANSIENT'

Aborted

y el Name Service muestra el siguiente mensaje de error:

ACE_SSL (6943|3077863104) error code: 336105650 - error:140890B2:SSL routines:SSL3_GET_CLIENT_CERTIFICATE:no certificate returned

---

Prueba 8:

Se levanta el cliente utilizando como protocolo de comunicación SSLIOP y llaves y certificados generados por nuestra entidad certificadora y utilizando client.conf como fichero de configuración.

luis@luispc:~/entregable/Syncware/SecurityClientTest/Debug$ ./SecurityClientTest -ORBInitRef NameService=corbaloc:ssliop:192.168.10.103:12345/NameService -ORBSvcConf client.conf

Se presiona "1" y el cliente muestra que funciona correctamente

Glosario de Términos

“A”

ANSI: El Instituto Nacional Estadounidense de Estándares (ANSI, por sus siglas en inglés: American National Standards Institute) es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos.

ALBA: Alternativa Bolivariana para América Latina

“C”

CORBA: Estándar de sistema de objetos distribuidos. Especifica la arquitectura que debe tener un sistema de objetos distribuidos. Establece un modelo de objetos mínimo: cada objeto obedece a una interfaz. La definición de interfaces se basa en la utilización del lenguaje IDL de OMG. La reutilización de interfaces se consigue mediante mecanismos de herencia de interfaces. Se utiliza herencia múltiple. No es posible acceder a los detalles de implementación de un objeto

C++: El C++ (pronunciado "ce más más" o "ce plus plus") es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

CGI (Common Gateway Interface o Interfaz de Entrada Común): Es una importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME.

“F”

Framework: En los sistemas orientados a objeto un framework es un conjunto de clases que encapsulan diseños abstractos de soluciones a un determinado número de problemas en relación. Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

“G”

GNU/Linux: Aunque se ha difundido el término Linux como siglas de la frase: Linux Is Not Unix (Linux no es Unix), Linux es un núcleo de sistema operativo tipo Unix. El núcleo Linux se complementa con una serie de aplicaciones desarrolladas por el grupo GNU para conformar el sistema operativo software libre GNU/Linux. Linux/GNU es además Multiusuario, Multitarea, Multiprocesador, Multiplataforma, Multilingüe, nacido en la red de redes Internet.

“H”

HMI: La interfaz Hombre – Máquina (en ingles, Human Machine Interface) es el aparato que presenta los datos a un operador (humano) y a través del cual éste controla el proceso

“I”

ISO: Organización Internacional para la Estandarización o International Organization for Standardization, que nace después de la segunda guerra mundial (fue creada en 1946), es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica.

“L”

LDAP (Lighweight Directory Access Protocol): Es un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP puede considerarse una base de datos (aunque su sistema de almacenamiento puede ser otro diferente) al que pueden realizarse consultas.

“M”

MAC: Es la concatenación de la autenticación y la integridad de los datos recibidos mediante el resumen de cada mensaje recibido, con un número de secuencia y un número secreto establecidos en el estado de conexión. Todo esto se añade al mensaje. Con esta base, la autenticación se comprueba mediante el número secreto, compartido por el cliente y el servidor, y mediante el número de secuencia, que viaja siempre encriptado.

MTU: (Master Terminal Unit o Estación Maestra): El término "Estación Maestra" se refiere a los servidores y el software responsable para comunicarse con el equipo del campo (RTUs, PLCs) en estos se encuentra el software HMI corriendo para las estaciones de trabajo en el cuarto de control, o en cualquier otro lado. En un sistema SCADA pequeño, la estación maestra puede estar en un solo computador, A gran escala, en los sistemas SCADA la estación maestra puede incluir muchos servidores, aplicaciones de software distribuido, y sitios de recuperación de desastres.

“O”

ORB: El elemento que diferencia una implementación de CORBA de otra es el ORB (Object Request Broker). El ORB es el "Run Time" que implementa proxies y referencias a objetos. Cómputo de referencias a objetos. La base de datos de interfaces y la base de datos de implementaciones. El mecanismo básico de invocación de métodos. Del ORB dependen la mayor parte de características de rendimiento e interoperabilidad.

OMG: El Object Management Group u OMG (de sus siglas en inglés Grupo de Gestión de Objetos) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas.

OCSP (Online Certificate Status Protocol): Es un Protocolo para comprobar el Estado de los Certificados En Línea. Es un método para determinar el estado de revocación de certificado digital X.509 usando otros medios que no sean el uso de CRL(Listas de Revocación de Certificados). Este protocolo se describe en el RFC2560 y está en el registro de estándares de Internet.

“P”

Phishing: Término informático que denomina un tipo de delito encuadrado dentro del ámbito de las estafas, y que se comete mediante el uso de un tipo de ingeniería social caracterizado por intentar adquirir información confidencial de forma fraudulenta (como puede ser una contraseña o información detallada sobre tarjetas de crédito u otra información bancaria).

PKI: En criptografía, la Infraestructura de clave pública (o en ingles, Public Key Infrastructure) es una combinación de hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital o el no repudio de transacciones electrónicas. El término PKI se utiliza para referirse tanto a la autoridad de certificación y al resto de componentes, como para referirse, de manera más amplia al uso de algoritmos de clave pública en comunicaciones electrónicas

“R”

RTU: La Unidad Terminal Remota (sus siglas en ingles Remote Terminal Unit): Dispositivos basados en microprocesadores, el cual permite obtener señales independientes de los procesos y enviar la información a un sitio remoto. Las RTU en los tiempos han sido desplazadas por los PLC quienes han fortalecido sus facilidades de comunicación a través de protocolos para sistemas de control (MODBUS, DNP3, IEC-101, etcétera.).

“S”

Sockets: Es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (API, Application Programming Interface).

SCVP (Simple Certificate Validation Protocol): Es un Protocolo de Simple Validación de Certificados, es decir, es un protocolo de acceso a los servicios de validación. Gestiona la

publicación de certificados revocados por él, LDAP, CRL y OCSP. Es uno de los mecanismos más eficaces para determinar el estado de certificados. SCVP y OCSP son protocolos en tiempo real.

“T”

TCP/IP: Conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras. En ocasiones se la denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron los dos primeros en definirse, y que son los más utilizados de la familia. Existen tantos protocolos en este conjunto que llegan a ser más de 100 diferentes.