Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática

Los servicios Web, su implementación, utilización y evolución en la informática distribuida

Tesis de Grado para optar al Título de
Ingeniero Civil en Informática

Patrocinante:
Sra. Gladys Mansilla G.
Ingeniero Matemático.
Analista de Sistemas.
Magíster en Estadística

Elsa Johanna Díaz Sobarzo

VALDIVIA - CHILE
AÑO 2006

VALDIVIA, 3 de Agosto del 2006

DE: GLADYS MANSILLA GOMEZ.
A : JUAN PABLO SALAZAR. DIRECTOR ESCUELA ING. CIVIL EN INFORMATICA

MOTIVO INFORME TRABAJO DE TITULACIÓN

Nombre Trabajo de Titulación: LOS SERVICIOS WEB, SU IMPLEMENTACIÓN, UTILIZACIÓN Y EVOLUCIÓN EN LA INFORMÁTICA DISTRIBUIDA

Nombre del alumno: ELSA JOHANNA DIAZ SOBARZO

Nota: 6.8 seis coma ocho
(en números) (en palabras)

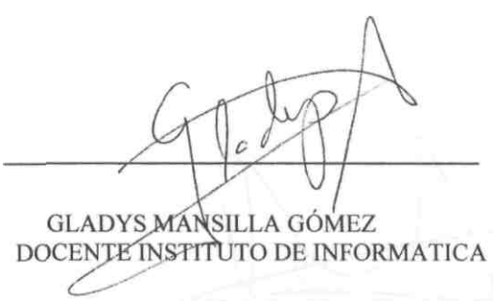
Fundamento de la nota:

- Este trabajo de tesis es de gran valor dado que la alumna logró con éxito incorporar una gran cantidad de nuevos conocimientos a su formación de modo de de aplicar con éxito esta tecnología, y describir sus utilidades y las futuras implementaciones.

- En este trabajo es posible destacar la claridad con que el alumno utiliza la nueva terminología relativa a servicios web, la seguridad y su arquitectura, adquirida durante su trabajo.

- En la realización de este trabajo de titulación se alcanzan plenamente los objetivos planteados al inicio.

  La presentación y redacción del informe están bien elaboradas, abarcando tópicos que inciden directamente en esta tesis y expresado en un lenguaje formal apropiado.



GLADYS MANSILLA GÓMEZ
DOCENTE INSTITUTO DE INFORMATICA

VALDIVIA, Miércoles 2 de Agosto de 2006

DE : **JULIO DANIEL GUERRA HOLLSTEIN**
A : **DIRECTOR ESCUELA INGENIERÍA CIVIL EN INFORMÁTICA**

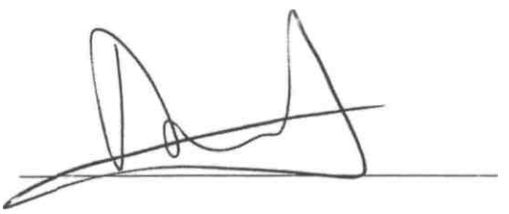
MOTIVO: **INFORME TRABAJO DE TITULACIÓN**

Nombre Trabajo de Titulación:	Los servicios Web, su implementación, utilización y evolución en la informática distribuida Elsa
Nombre del Alumno:	Johanna Diaz Sobarzo

Nota **4,7** *cuatro amia siete*

FUNDAMENTO DE LA NOTA:

Cumplimiento del objetivo propuesto	5,5
Satisfacción de alguna necesidad	5,5
Aplicación del método científico	4,0
Interpretación de los datos y obtención de conclusiones	5,5
Originalidad	5,0
Aplicación de criterios de análisis y diseño	4,5
Perspectivas del trabajo	5,0
Coherencia y rigurosidad lógica	4,5
Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración	3,0
PROMEDIO	**4,7**



Julio Daniel Guerra Hollstein
Ingeniero Civil en Informática

Valdivia, 03 de agosto de 2006

De: Juan Pablo Salazar Fernández
Profesor Informante

Ref: Calificación proyecto de título

De mi consideración:

Habiendo revisado el trabajo de titulación **"Los servicios web, su implementation, utilización y evolución en la informática** distribuida". presentado por la alumna srta. Elsa Johanna Díaz Sobarzo, mi evaluación del mismo es la siguiente:

Nota: 5,5 (cinco, coma cinco).

Fundamento de la nota:

El presente trabajo de titulación se planteó como objetivo el análisis de los servicios web. Este objetivo se cumplió en términos generales, pero con un nivel de profundidad insuficiente en algunos casos, y con poca claridad en otros.

Los capítulos que exponen aspectos conceptuales son en general bastante completos, pero habría sido recomendable la utilización de diagramas y ejemplos más precisos para clarificar conceptos.

El documento presenta algunas deficiencias en su forma, que debiesen ser corregidas.

Aspecto	**Evaluación**
Cumplimiento de objetivos	5,5
Satisfacción de alguna necesidad	5,0
Aplicación del método científico	5,0
Interpretación de los datos y obtención de conclusiones	5,5
Originalidad	6,0
Aplicación de criterios **de** análisis y diseño	5,0
Perspectivas del trabajo	6,0
Coherencia y rigurosidad lógica	5,5
Precisión del lenguaje técnico	6,0

Sin otro particular, saluda atentamente a usted,

Juan Pablo Salazar Fernández
Profesor Informante

DEDICATORIA

Al pilar fundamental de mi vida, quien me dio siempre la fuerza para seguir adelante, quien a pesar de estar a muchos kilómetros de distancia siempre estuvo a mi lado, diciéndome que no podía dejar de luchar, MI HIJO.

A MIS PADRES, quiénes a pesar de todas las dificultades vividas, creyeron en mí: Mi papá que con su sacrificio supo ayudarme y darme las armas para luchar; mi mamá que con su cariño hizo sentirme segura y confiada de que todo marchaba bien. Los quiero mucho, les estoy muy agradecida y siempre tendrán un lugar en mi vida y en mi corazón.

A MIS AMIGOS, gracias por todo, gracias por estar cerca y ayudarme a conllevar las ausencias, gracias sobre todo a ti Mónica, que con tu constancia me obligaste a superarme, sin tu apoyo, tus retos y el obligarme a estudiar quizás me hubiera costado más llegar al final.

AGRADECIMIENTOS

Sólo queda agradecer a todas las personas que desinteresadamente ayudaron en la realización del presente trabajo de Tesis.

Principalmente quisiera nombrar a Juan Pablo Schmiede (el jefe), arquitecto de soluciones de SONDA S.A., quien tuvo la paciencia de soportarme y enseñarme; Juan Pablo García, MVP (Most Value Professional) de Microsoft, quién pese a su poquito tiempo igual tuvo un ratito para ayudarme y Alvaro Olivares, Líder de Desarrollo de SONDA, quien toleró responderme el mar de dudas y aclararme la ideas.

LOS PROFESORES, la Sra. Gladys Mansilla, gracias por su confianza; Don Juan Pablo, creo que ambos creímos que las metas no se iban a cumplir, pero se logró, gracias por su apoyo; y a Daniel quién a última hora molesté pero tuvo la predisposición para ayudarme. Muchas Gracias.

INDICE

INDICE DE FIGURAS

SINTESIS

Como resultado de una combinación de las mejores aspectos de la programación orientada a componentes y la programación Web, surgen los servicios Web, los cuales pueden ser reutilizables sin que la implementación o el lenguaje, sistema operativo o modelo de componente utilizado en su generación sean un obstáculo. Lo que implica que el usuario no tiene necesariamente que saber qué se tiene instalado o cómo funciona para poder utilizar su funcionalidad, logrando así que su acceso se realice a través de protocolos de Internet basados en XML (HTTP o SMTP).

Para diseñar, desarrollar e implantar Servicios Web Seguros, los arquitectos deben aprender las nuevas tecnologías y considerar los posibles riesgos que la utilización de estos lleva. La implantación de la seguridad, en ocasiones, se hace muy complicada por el hecho de que para diferentes proyectos, o diferentes organizaciones, se poseen diferentes requerimientos de seguridad.

Con la ayuda de la estandarización de los métodos o protocolos de seguridad para los Servicios Web, su utilización ya no presenta un riesgo a nivel empresarial. Es por esto que cada vez estos servicios se hacen más utilizados y requeridos en los diferentes ámbitos de negocios.

En el presente trabajo de tesis se hace especial énfasis a la seguridad y los métodos que esta tecnología lleva asociada para el logro efectivo de la interoperabilidad o comunicación entre plataformas. Los conceptos que se deben tener en cuenta al

momento de implementar la seguridad, su forma de implementación y su utilización se podrá ver reflejada en el transcurso de esta tesis.

SYNTHESIS

As a result of the combination of the best aspects of component oriented programming and web programming arise web services, which are reusable, no matter the implementation, language, operating system or component model used in their generation. That implies that the user doesn't necessarily have to know what's installed or how it works to use it, making it possible to access them through internet protocols based on XML.

To design, develop and implant Secure Web Services must learn new technologies and consider the possible risks involved in their use. The implantation of security sometimes turns out to be quite complicated, because different projects or organizations have different security requirements.

With the standardization of methods or security protocols for web services, their use is no longer a threat at enterprise level. That's the reason why they are more used and required in different business areas each day.

This thesis emphasizes security and methods associated with this technology to effectively achieve interoperability and communication between platforms.

The concepts that owe to bear in mind the moment to implement the safety, his form of implementation and his use it will be possible to see reflected in the course of this thesis.

CAPITULO 1

1 INTRODUCCION

1.1 ANTECEDENTES HISTÓRICOS

Revisando un poco la historia, a comienzos de los años noventa apareció un protocolo para transmitir información de manera simple a través de la naciente Internet, este protocolo se llamó HTTP (Hyper Text Transport Protocol), que es el método mas común de intercambio de información de lo que hoy conocemos como WWW (World Wide Web) método con el cual se transfieren las páginas Web a un computador, cuya principal característica es que hace uso del texto activo, o sea, es capaz de poseer texto normal, y a su vez posee palabras resaltadas, que están enlazadas, ya sea con un objeto, ventana o imagen, sobre las cuales se puede hacer clic con el Mouse y realizar toda una navegación sobre ellas, avanzar y retroceder a través de ellas a nuestro antojo.

Ya a mediados de los noventa, y gracias a un experimento de Sun Microsystem, surge un nuevo lenguaje de programación, con la novedad de que está orientado a objetos (OOP, siglas de su nombre en inglés Object Oriented Programming), lo cual se vuelve una revolución en la forma de hacer aplicaciones y/o programas, ya que podían ser ejecutados en diferentes sistemas operativos, sin cambiar el lenguaje de programación en el que fueron escritos.

Al mismo tiempo, también aparece un estándar de intercambio de mensajes, hoy conocido como XML (e**X**tended **M**arkup **L**enguage). Si bien, XML nace en los años 70 (creado por IBM), es en estos años, los 90, cuando la W3C (World Wide Web

Consortium, organización que produce estándares para la WWW), se hizo cargo de su desarrollo y estandarización [URL12].

1.2 INTRODUCCIÓN A WEB SERVICES

Hoy en día, ya no es una novedad hablar sobre el uso de los Web Services. Esta tecnología utiliza XML como forma de comunicación y flujo de información entre los sistemas distribuidos de redes existentes. Para poder hacer que esta nueva tecnología funcione es necesario agregarle algunos elementos adicionales.

Para que un Web Service funcione, es necesario que las aplicaciones que lo utilicen sepan la cantidad y tipos de parámetros que éste necesita, así como la forma o tipo de los mismo (números, signos, letras, listas, etc.) y el nombre del método que necesita utilizar (un Web Service puede poseer muchos métodos, o sea, servir para varios propósitos). Hasta aquí, un Web Service es muy similar a un objeto de la OOP.

A continuación se pasa a detallar la diferencia: para utilizar estos servicios Web, es necesario saber donde están ubicados. Esta localización es una componente adicional nuevo, es el que informa a la aplicación que necesita utilizarlo el como “acceder” al mismo. Para informar la localización, el servicio utiliza una dirección que se llama URI (**U**niform **R**esource **I**dentifier, Identificador Único de Recurso), el cual es capaz de identificar un servicio o recurso disponible en una red.

Normalmente un URI consta de dos partes:

1. **Identificador del método de acceso** (protocolo) al recurso, por ejemplo http, mailto, ftp.

2. **Nombre del recurso**, por ejemplo "//mipagina.servicio.com"

Entonces el URI dice del Web Service no sólo en “dónde” está, sino el “cómo” acceder apropiadamente a él. El “Cómo” viene expresado por el identificador del método (http:, ftp:, mailto:).[URL13]

Lo más importante de esta tecnología es los servicios y/o la utilidad que se prestan entre diferentes plataformas de desarrollo, esto gracias a toda la colección de estándares y protocolos definidos para intercambiar datos entre aplicaciones.

A lo largo de esta tesis se ayudará a comprender el cómo y el para qué de los servicios Web, si bien no se realizará una aplicación que haga uso de esta tecnología, si se mostrará la mejor forma de implementarlos, haciendo especial énfasis en la seguridad, tópico esencial a la hora de trabajar en aplicaciones distribuidas. El hecho de que no se haga una implementación es debido a que en el correr de los años el formato XML (su forma de escribirse) no ha variado, pero si se han modificado los estándares para lograr un sistema seguro y confiable, gracias a lo cual este servicio ha logrado posicionarse a un nivel mas arriba que cualquier otro tipo de intercambio de información entre sistemas.

1.3 ANTECEDENTES ACTUALES

Los servicios Web (Web Services) y el XML están cada vez más de moda en el mundo de la programación, día a día se realizan más estudios para aumentar sus funcionalidades y servicios que ofrecen en los entornos o ambientes en los que son utilizados.

Las ideas detrás de los Web Services no son nuevas, ya estaban presentes en tecnologías como RPC (Remote Procedure Call). Pero el problema con RPC fue la seguridad, su simpleza, además de tener la desventaja que su implementación en un ambiente como Internet es muy complicada, debido a que existen muchos cortafuegos que bloquean este tipo de mensajes.

En un principio se hicieron intentos de crear estándares pero fracasaron o no tuvieron suficiente éxito, debido a que la mayoría eran muy dependientes de la plataforma del vendedor (DCOM → Microsoft, CORBA → ORB).

Entonces, los Web Services (WS) nacieron para lograr la tan ansiada comunicación entre las diferentes plataformas. Es por esto que en 1999 se comenzó a plantear un nuevo estándar, el cual terminaría utilizando XML, SOAP, WSDL y UDDI

De hecho uno de los protocolos para la confección de WS que ahora se utiliza se llama precisamente XML – RPC y el otro se llama SOAP. A la hora de programar un WS se debe elegir cual de estos dos protocolos utilizar, ya que no son compatibles entre ellos. Microsoft trabaja con SOAP.

1.3.1 SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

Es el protocolo de comunicación de mensajes independiente del transporte, el cual, hoy en día, es el que más se utiliza para la implementación de WS.

Este protocolo, tal como lo dice su nombre, es tan simple que sólo es una combinación de HTTP + XML y su labor es describir como enviar los mensajes XML sobre HTTP.

Su estructura consta de lo siguiente:

- Sobre (Envelope): Cabecera (Header) [Opcional] + Cuerpo (Body)

Un mensaje de petición, en el cual se invoca un método (ObtenerUltimoPrecio) solicitando específicamente el valor de un producto (DIS) se escribiría de la siguiente forma:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <m:ObtenerUltimoPrecio xmlns:m="http://example.com/stockquote.xsd">
         <symbol>DIS</symbol>
      </m: ObtenerUltimoPrecio >
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Y el mensaje con la respuesta del valor consultado, se escribe:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
      <m: ObtenerUltimoPrecioResp xmlns:m="http://example.com/stockquote.xsd">
         <Precio>350</Precio>
      </m: ObtenerUltimoPrecio Resp>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

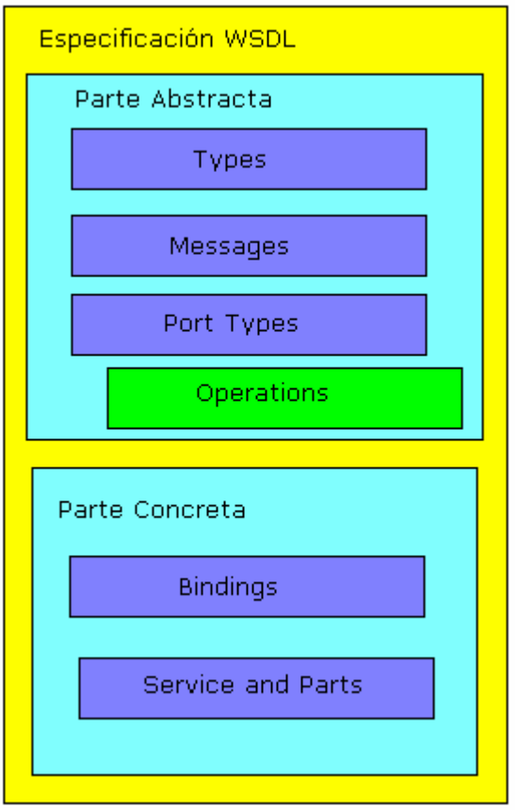
1.3.2 WSDL (WEB SERVICE DESCRIPTION LANGUAGE)

Es utilizado para describir el Servicio Web, especificar su localización y describir las operaciones y métodos que él expone. En pocas palabras, es una librería utilizada para describir el servicio.

Lo mejor de todo es que no es necesario conocer este protocolo para construir o consumir servicios Web, ya que los lenguajes más conocidos, como por ejemplo .NET, ofrecen herramientas que generan automáticamente estas descripciones y también son capaces de leerlas y mostrarnos la información relevante. Una vez localizada la dirección de la descripción WSDL de un servicio, integrarlo en una aplicación es inmediato.

1.3.3 UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION)

Se trata de una norma en proceso de evolución que permite publicar y descubrir información acerca de los servicios Web que están desarrollando empresas de todo el mundo, ofreciendo una infraestructura necesaria para poder registrar servicios Web o buscar otros utilizando descripciones estándar basadas en WSDL. Gracias a este sistema descubriremos donde podemos acceder a las descripciones WSDL que permiten utilizar un Servicio Web.



La estructura de un documento WSDL está compuesta por una parte abstracta, independiente de la aplicación; y una concreta, la cual define los enlaces.

La parte abstracta:

. Port Type: es una colección lógica de operations. Hacen referencia a las definiciones de mensajes, para describir la firma de las funciones (nombres de la operación, parámetros de entrada, parámetros de salida)

. Operations: cada una de ellas define un intercambio simple de mensajes.

. Message: unidad de comunicación. Contiene parámetros de funciones (entradas separadas de las salidas) o descripciones de documentos.

. Types: son utilizados por operations (XML). Definiciones de tipos independientes del lenguaje y del equipo.

La parte concreta:

. Bindings: se especifica la codificación de los mensajes y los enlaces a protocolos definida en un Port Type.

. Port: define un punto donde se encuentra el mensaje. Punto final único que se define como la combinación de un enlace y una dirección de red.

. Service: definen una agrupación de Ports. Dirección o direcciones de puerto de cada enlace

1.3.4 OTROS PROTOCOLOS

- WSIL (Web Service Inspection Language): Al igual que UDDI proporciona un método de descubrimiento de servicios para WS. La diferencia está en que utiliza un sistema descentralizado.

- JAC – RPC: define bibliotecas API de JAVA que los desarrolladores pueden utilizar en sus aplicaciones. También utiliza el protocolo de mensajería XML [BOR00]. Se puede utilizar con otros lenguajes.

Microsoft es una de las entidades que frecuentemente se encuentra realizando capacitaciones, charlas y/o conferencias técnicas acerca de las novedades de estos servicios para las personas que trabajan utilizando este ambiente de desarrollo

Figura 1.- Certificado Conferencia Técnica Web Services

1.4 IMPORTANCIA E IMPACTO DE LA INVESTIGACIÓN

Desde que Microsoft, por el año 2001, anuncia el uso de Web Services y la importancia que tendría para el futuro de la Web y la vida cotidiana, es que esta herramienta ha comenzado a surgir como la "gran herramienta" de comunicación que el hombre puede utilizar para unir dos o mas plataformas. Es ahora donde la tecnología se pone al servicio del hombre y no al revés como, hasta ese entonces, estaba ocurriendo, ya que el hombre debía elegir con que plataforma trabajar y estas eran excluyentes entre sí.

Si bien los Web Services aún están en etapa evolutiva, estos ya se están utilizando masivamente a nivel de empresas comercio y de entidades públicas del estado (a modo de ejemplo se puede mencionar que la alumna ha trabajado con WS del Servicio de Impuestos Internos, Tesorería General de la República, etc. Y de empresas tales como

SONDA S.A., CONSALUD, entre otros), ya que logran los objetivos principales para los que han sido creados.

Uno de los principales objetivos que tiene este estudio es dar a conocer la importancia de la implementación de estos servicios, sobre todo a niveles empresariales, ya que, si utiliza esta tecnología, la empresa tiene la posibilidad de implementar los servicios y exponerlos para aquellos sistemas que requieran consultar una información específica de ella, de manera rápida, segura y confiable, sin la necesidad de establecer un link hacia ésta y hacer que el usuario tenga que navegar en otro sitio para encontrar lo que busca.

Un ejemplo concreto que se puede citar es saber si una persona consultada tiene protestos y/o morosidades en el servicio de impuestos internos, para esto el sistema local sólo establece la comunicación y en la misma página inserta la respuesta y el detalle. Entonces, ya no es necesario que el consultante deba navegar a la página de impuestos internos, ingresar nuevamente el Rut y presionar el botón consultar.

Las entidades públicas del estado, están haciendo uso masivo de estos servicios, ya que la colaboración entre ellas es casi obligatoria para conseguir un buen nivel de servicio y/o atención con los usuarios online que mantienen.

También en el caso de la educación virtual la utilización de los Web Services es de vital importancia, ya que los sistemas se pueden comunicar con otros sistemas legacy internos (ejemplo RRHH) o con almacenes de contenidos educativos online distribuidos en cualquier parte del mundo.

Entonces, he aquí la importancia de conocer la utilidad de estos servicios y de saber identificar exactamente que parte de nuestro negocio es necesario implementarlo utilizando esta tecnología. Algunas de las ventajas con las que se encontrará serán:

- **Facilitan la comunicación,** mayor facilidad para que sus datos o los de su empresa u organización sean conocidos.

- **Facilitan la comunicación**, mayor facilidad para consultar datos de personas, servicios, organizaciones, etc., desde su sistema.

- **Aminoración de costos**, si es que le llegasen a cobrar por revisar datos, sólo se le cobrará el hecho de poder visitarlos desde su página, no se le cobrará por cada visita que realice al sitio externo.

- **Aminoración de costos**, la forma de implementar una llamada a un Web Services específico, siempre va a ser el mismo. Si la llamada se implementa en una empresa que desarrolla software, la llamada a un determinado servicio se puede “prestar” entre aplicaciones.

- **Aumentan los ingresos**, exponiendo servicios y/o costos asociados a su empresa u organización al conocimiento de otros.

Todo lo anteriormente descrito pone de manifiesto que lo Web Services son fáciles de encontrar, compartir, implementar y reutilizar, pero cómo lograr la confianza para que sean implementados y cómo conseguir que la seguridad sea accesible desde cualquier sistema. En el desarrollo de este trabajo de tesis se podrá encontrar la respuesta a este desafío, el cual ya no es un obstáculo a la hora de implementar estos servicios.

1.5 OBJETIVOS GENERALES Y ESPECÍFICOS

1.5.1 OBJETIVO GENERAL

Investigar el estado actual de los servicios Web, los avances que ha tenido desde sus comienzos, sus utilidades y las futuras implementaciones que se pretenden realizar para lograr una mayor utilidad del servicio.

1.5.2 OBJETIVOS ESPECÍFICOS

- **Investigar la Arquitectura de los servicios Web:** Se conocerá como están implementados estos servicios y el porqué de su arquitectura.

- **Definir los Estándares de servicios Web:** Se conocerán las normas sobre las que se basan, hasta ahora, los servicios Web y el estado actual de las implementaciones de los futuros estándares y/o mejoras de los ya conocidos.

- **Analizar la Seguridad en Servicios Web:** Se analizará la fiabilidad de estos servicios mediante el análisis e investigación de la arquitectura de seguridad de los solicitantes del servicio, además de cómo transferir el contexto de seguridad a través de los servicios Web.

- **Implementación y Comparación de servicios Web mediante diferentes tecnologías:** Se investigarán y compararán las técnicas de implementación de estos servicios en plataformas tales como jBuilder y .Net, entre otras.

- **Comparación de los Servicios Web con otras técnicas de envío/recepción de datos:** Se realizarán comparaciones de rendimiento, funcionalidad, rapidez, entre otros, con tecnologías tales como NET.Remoting, CORBA, Message Queue, entre otras, etc.

- **Web Services, El futuro:** Se analizará, principalmente a través de la Web, los avances que están teniendo estos servicios y, lo que se prevé, será en un futuro cercano.

CAPITULO 2

2 ARQUITECTURA Y SEGURIDAD DE LOS WEB SERVICES

2.1 ARQUITECTURA GENERAL DE LOS WEB SERVICES

La arquitectura de servicios Web plantea algo más que una técnica para el desarrollo de aplicaciones Web, representa un modelo de computación distribuida para Internet basado en XML. Bajo este concepto ya no sólo se trata la comunicación usuario - aplicación, sino que de manera adicional se maneja la interacción aplicación - aplicación.

La **Arquitectura básica de Web Services** debe incluir lo necesario para:

- El intercambio de mensajes entre las entidades
- Publicar en la Red, poner en conocimiento la existencia de ese Web Service, para que pueda ser utilizado por algún cliente.
- Encontrar otros Web Services.

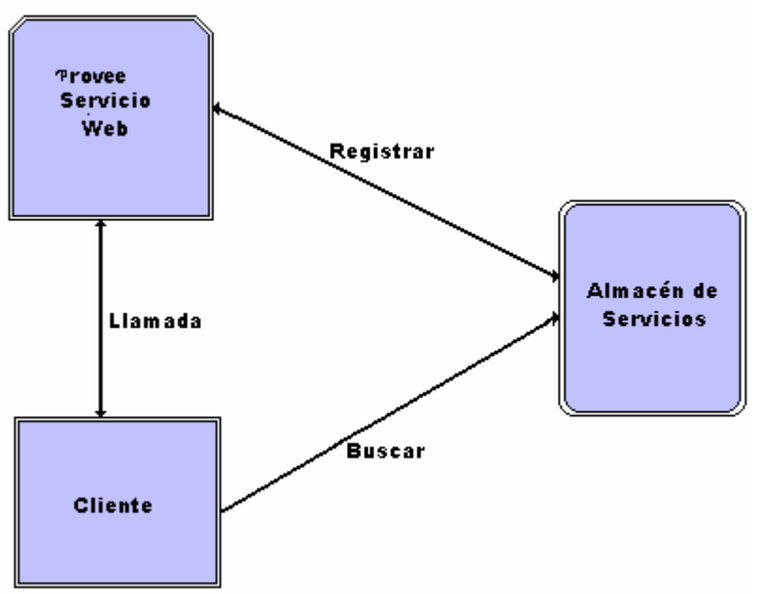


Figura 2.- Esquema General del funcionamiento de los Servicios Web

Según lo que se muestra en la figura, un Servicio Web se registra en un almacén de servicios, el cliente lo busca, lo encuentra y luego lo invoca.

Un Servicio Web puede:

- estar descrito mediante el lenguaje de descripción de servicio WSDL

- estar publicado utilizando el método de registro UDDI

- ser asociado a una descripción para crear una instancia de servicio disponible

- ser invocado SOAP,

- estar compuesto por otros servicios para integrar servicios y aplicaciones nuevas y

- ser publicado HTTP

Entonces, con estas definiciones llegamos a una figura de la arquitectura de un Servicio Web mas detallada

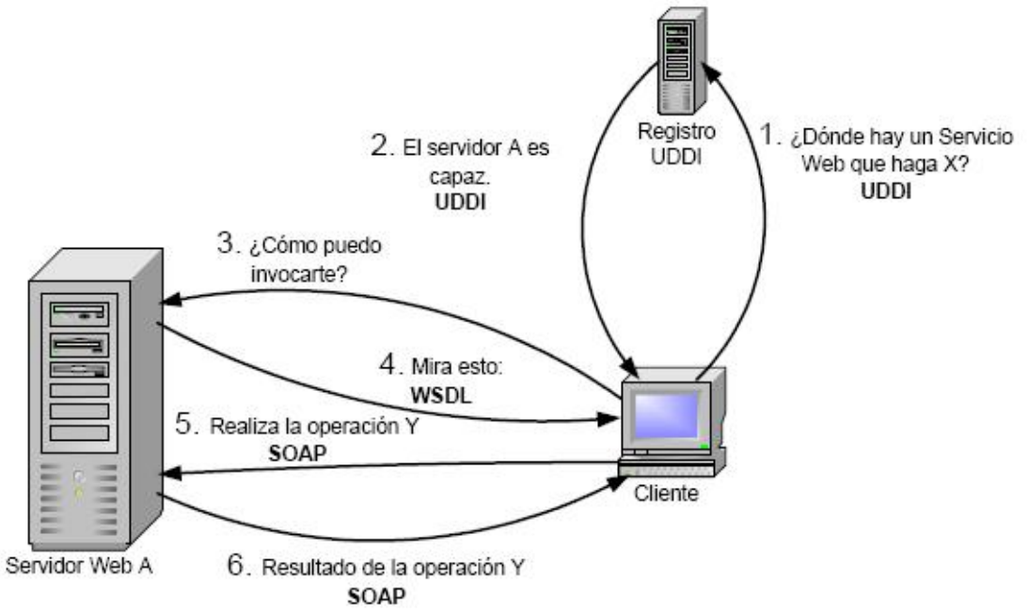


Figura 3.- Esquema General del funcionamiento de los Servicios Web

En pocas palabras, el Servicio Web es construido y luego descrito por medio de WSDL y registrado bajo el estándar UDDI, el cliente busca en el registro UDDI (como si fuese un motor de búsqueda al estilo de Google) y obtiene el descriptor WSDL del servicio que necesita, lo invoca haciendo uso de SOAP el cual también es utilizado para comunicar la petición entre los diferentes componentes del servidor que aloja el Servicio Web, para entregar una respuesta utilizando nuevamente SOAP.

La principal ventaja de la arquitectura de los Web Services es, como ya se ha dicho anteriormente, la característica de que los programas escritos en lenguajes y plataformas diferentes, se pueden comunicar entre ellos. Se puede decir que los 2 elementos que brindan el mayor valor agregado de esta arquitectura son el débil acoplamiento entre los servicios (componentes) que redundan en una mayor velocidad de respuesta (facilidad para planear y ejecutar cambios) y en el aumento en la reutilización de los recursos

tecnológicos, que implica una reducción en los costos y un mejor retorno de las inversiones en tecnología.

Para aclarar un poco más el concepto tomemos como ejemplo una rutina de programación, como sabemos una rutina es como una caja negra, la cual encierra un proceso y que cumple una función claramente definida, luego para construir una aplicación llamamos dichas rutinas enviando parámetros y recibiendo la respuesta respectiva. Un Servicio Web se puede considerar como una rutina a la cual se le envían los parámetros utilizando XML encapsulados en el protocolo HTTP.

2.2 ROLES EN UNA ARQUITECTURA WEB SERVICE

Los roles primordiales en una arquitectura WS se detallan a continuación, cabe mencionar que existen otros tipos de roles, pero que son opcionales dependiendo del tipo de WS que se desea consultar, de la seguridad que se desee implementar y algunos otros factores.

Proveedor del servicio: desde la perspectiva de un negocio, este es el proveedor del servicio. Desde una perspectiva de arquitectura, es la plataforma en la que esta alojado el servicio y la cual da acceso a él.

Solicitante del servicio: desde la perspectiva del negocio, éste es el negocio que requiere que ciertas funciones sean satisfechas. Desde una perspectiva de arquitectura esta es la aplicación que está mirando para invocar o inicializar una interacción con un

servicio. El rol de solicitante lo puede poseer un browser manejado por una persona o programa sin una interfase de usuario, por ejemplo, otro Web Service.

Registro del Servicio: este es un registro sobre el cual se puede realizar una búsqueda de la descripción de un servicio, el cual fue puesto allí por un proveedor de un servicio. El solicitante busca un servicio y obtiene información de cómo y dónde realizar la llamada. El registro del servicio es un rol opcional en la arquitectura de los WS, debido a que, por ejemplo, el proveedor puede enviar directamente la información al solicitante.

2.3 OPERACIONES EN UNA ARQUITECTURA WEB SERVICES

Los roles antes mencionados deben interactuar entre ellos. Para esto los WS tiene definidas las operaciones que se pueden realizar entre ellos. En una arquitectura WS estas operaciones están claramente definidas:

Publicación: El desarrollador, o la entidad desarrolladora, publica el WS para ofrecerlo a la comunidad de potenciales usuarios del servicio. Lo que se publica es la descripción del servicio

Búsqueda: A través de esta operación un desarrollador busca un servicio directamente o realizando consultar al registro del servicio. Esta operación se realiza cuando se desea encontrar un servicio y cuando se desea acceder a él

Enlace: Este enlace puede ser sincrónico o asincrónico y es el tiempo que requiere el solicitante para que se ejecute una acción sobre el WS.

2.4 SEGURIDAD EN LOS WEB SERVICES

En esta sección se realizará un estudio de los aspectos de la seguridad que se deben considerar a la hora de diseñar sistemas basados en los principios y las tecnologías definidas en el contexto de los servicios Web.

Se explicarán cuáles son las amenazas al momento de implementar o utilizar WS, así como, se indicarán posibles soluciones en este contexto o, en los peores casos, mecanismos para maximizar la reducción del riesgo.

La seguridad es un tópico esencial al momento de crear un WS, ya que si deseamos utilizar uno o más de estos servicios debemos asegurarnos de que pueda ser capaz de mantener su integridad en la red.

Hay ciertos puntos de los que debemos cuidarnos al momento de utilizar la red y enviar o trabajar con datos a través de ésta. Algunos de estos puntos son:

El tráfico de la red puede estar siendo “observado” (privacidad).

Quién envió el mensaje no es quién se esperaba (procedencia)

Puede que en la ruta seguida, el mensaje haya sido alterado (integridad).

Opcionalmente, seguridad de que el solicitante no pueda retractarse.

La arquitectura de seguridad de un WS se observa en el siguiente diagrama:

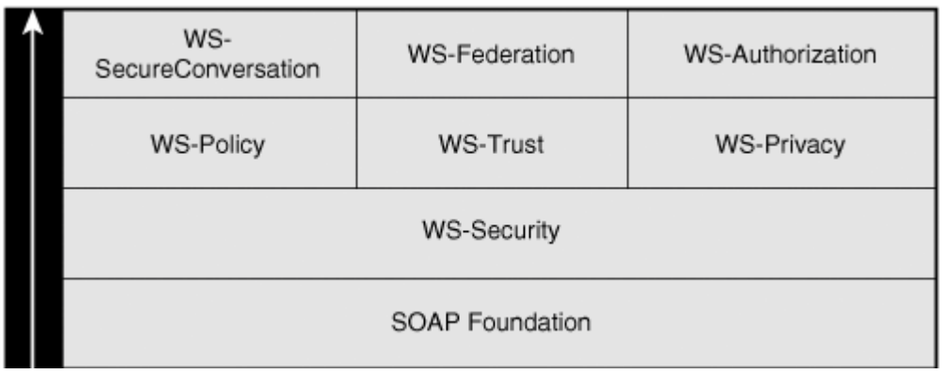


Figura 4.- Secciones de seguridad de un WS

A continuación se detalla cada una de las capas:

WS – Security: en esta capa se verifica lo siguiente [CHR05]:

Autenticación de su origen: Identifica si el origen de un mensaje es seguro

Integridad: Verifica que el mensaje se encuentre en las mismas condiciones en las que fue creado y por ende con la misma data.

Procedencia: verifica si el mensaje procede de una fuente esperada.

WS – Security hace uso de los estándares ya existentes para abordar la seguridad, tales como: Kerberos y X.509 que se ocupan de la autenticación, XML Encryption y XML

Signature, describen las formas de encriptación y codificación de claves. XML Canonicalization analiza los modos para que el XML esté preparado para la firma y cifrado [URL1]

WS – Security, define un encabezado SOAP para transportar los datos referentes a la Seguridad

WS-Security aumenta la seguridad existente en los sistemas y provee un modelo unificado para programadores de aplicaciones y administradores de sistemas.

WS – Policy: Esta especificación crea un marco para definir y especificar las expectativas y requerimientos ambos para recibir y enviar mensajes desde un Web Services, define el formato como sus requerimientos

WS-Trust: define el nivel de confianza que tendrán las aplicaciones. O sea, define un modelo conceptual para utilizar un WS, un solicitante revisa la declaración de la política de seguridad asociada a un WS. La declaración de la política especifica el tipo y la autoridad del token de seguridad que el WS requiere para que el mensaje pueda ser procesado. [CHR05]

WS-Privacy: Describe la sintaxis y la semántica para unir las políticas de privacidad de los Web Services e instanciar la data en los mensajes. El principal énfasis de WS-Privacy es habilitar las políticas para ser procesadas para el proveedor y el receptor del WS. No define un nuevo lenguaje de política de privacidad, pero si ofrece el medio para unir lenguajes existentes al servicio.

WS-Authorization: El propósito de WS-Authorization es describir como las políticas para un Web Services son especificadas y eventualmente manejadas. Su meta es describir como demandar con un token de seguridad y como interpretarlos. Esta diseñado para ser flexible y extensible con respecto a su formato y lenguaje de autorización.

WS-SecureConversation: Mientras WS – Security se centra en un modelo de autenticación de mensajes, WS – SecureConversation introduce el concepto de seguridad contextual y describe como utilizarlos, así permite que se intercambien más eficientemente las claves o el material de éstas. Para establecer este modelo de autenticación, WS – SecureConversation define una serie de cabeceras y extensiones del framework de mensajería SOAP. Por lo tanto su principal función es establecer los contextos de seguridad [URL2].

WS-Federation: Define mecanismos para habilitar identidades, cuentas, atributos, autenticaciones e información para la autenticación compartida a través de dominios seguros [CAB05], dar con un token de seguridad y como interpretarlo. Esta diseñado para ser flexible y extensible con respecto a su formato y lenguaje de autorización

2.5 MODELOS DE PROTECCIÓN EN LOS SERVICIOS

2.5.1 DETECCIÓN DE MENSAJES CLONADOS (REPLAY)

Cuando un cliente llama a un WS enviando mensajes a través de una red pública. Al llegar el mensaje al WS, éste lo procesa y puede suceder que el procesamiento que esta solicitud conlleva es el de la actualización o ingreso de datos, o sea se inicializan procesos de negocio, pero si uno de estos mensajes es interceptado uno de lo riesgos que se pueden presentar es que la misma acción se realice una cantidad de veces no deseada. Esto puede provocar, entre otras cosas, inconsistencia de datos.

Una de las formas de solucionar esto es depositando en caché un identificador del mensaje y así evitar que puedan ser registrados o ingresados dos o mas mensajes con el mismo identificador. Por cierto, el tiempo de vida en el caché va a estar limitado a la durabilidad que se le pretenda dar al mensaje. Para esto nuevamente se utiliza el TimeStamp.

La implementación de detección de mensajes clonados también requiere que el mensaje no haya sido alterado en tránsito, esto asegura que el identificador del mensaje no haya sido cambiado. Este identificador incrustado al mensaje es un tipo de firma digital.

Los participantes en la detección de mensajes clonados son los siguientes:

Cliente: es quién accede el WS.

Servicio: es el servicio ofrecido por el WS, el cual requiere autenticación para autorizar al cliente

Caché de Clonación: Se encarga de colocar el identificador único a los mensajes entrantes.

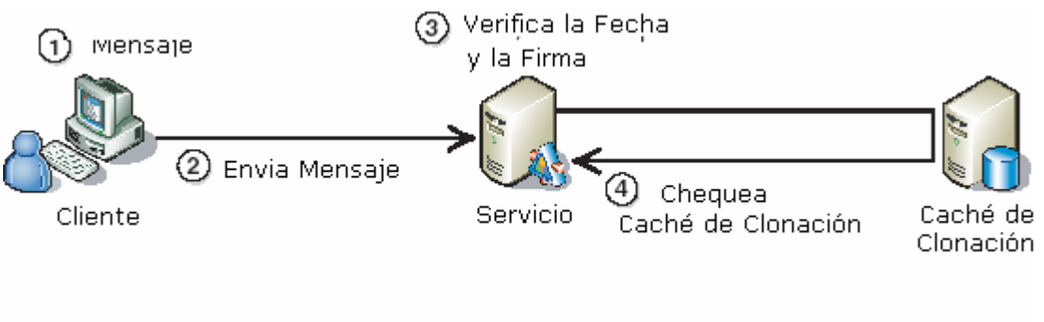


Figura 5.- Modelo de detección de mensajes clonados

El de envío/recepción de mensajes evitando la clonación, consta de los siguientes pasos:

1. Se envía el mensaje, la firma asegura que el mensaje no sea alterado en el tránsito.

2. El cliente envía el mensaje firmado al servicio.

3. El servicio verifica la firma del mensaje y el TimeStamp. Si la firma es inválida o el mensaje ha sido recibido fuera de la fecha aceptable, el mensaje es rechazado.

4. El servicio verifica en la caché de clonación si no existe un identificador previo de la firma digital que acompaña el mensaje. Si no existe un identificador previo se realizan los procesos que solicita el mensaje y se guarda el identificador en la caché, en caso contrario, el mensaje es rechazado.

Ejemplo de mensaje que contiene el TimeStamp para evitar su clonación:

```
<soap:Header>
⋮
```

```
    <wsu:Timestamp>
       <wsu:Created>2002-08-13T08:42:00Z</wsu:Created>
       <wsu:Expires>2002-08-14T08:42:00Z</wsu:Expires>
    </wsu:Timestamp>
    <wsse:Security>
    ⋮
    </wsse:Security>
    ⋮
  </soap:Header>
  <soap:Body> . . . </soap:Body>
```

Se puede ver que no es un elemento de WS-Security, ya que se puede implementar por separado.

Algunas Consideraciones en la Detección de Mensajes Clonados

- El identificador del mensaje debe ser guardada en la caché de clonación antes de que el mensaje se ejecute para evitar que éste no haya terminado de ejecutar su acción cuando llegue un “atacante”

- Se debe implementar el sistema de manera tal que el tiempo que se demora en guardar su identificador y hacer su validación no afecte el TimeStamp del mensaje. O en su defecto, implementar el mensaje de manera tal que posea cierto tiempo de holgura para estas operaciones.

- Para obtener óptimos resultados o los resultados deseados, se debe tener sincronización entre el cliente y el servidor, ya que si un mensaje llega en un periodo de tiempo no aceptable será rechazado.

2.5.2 VALIDACIONES DE LOS MENSAJES

Un WS interactúa con otras aplicaciones sobre la red y puede ser que en alguna de estas interacciones que se realicen, se puede encontrar con mensajes mal formados o con datos maliciosos. También puede resultar que al ser alterado en tránsito un mensaje, en él se trate de ingresar sintaxis adicionales.

Asumir que un mensaje es malicioso es la mejor forma de defenderse de este tipo de ataques, ya que se implementará el servicio buscando la mejor forma de protegerse. La validación lógica de un servicio define la política para especificar que parte del mensaje es la que se va a utilizar para la ejecución del servicio. Se valida el mensaje XML para asegurarse de que este documento se encuentre bien formado y que sea consistente con lo que el WS espera procesar. La validación lógica también especifica ciertos criterios para examinar el tamaño del mensaje, su contenido y el conjunto de caracteres que utiliza. Cualquier mensaje que no cumpla los criterios es inmediatamente rechazado.

Los participantes en esta validación, sólo son 2, el cliente y el servicio:

Cliente: es quién accede el WS.

Servicio: es el servicio ofrecido por el WS, que procesará los requerimientos cliente. El servicio implementa la validación lógica.

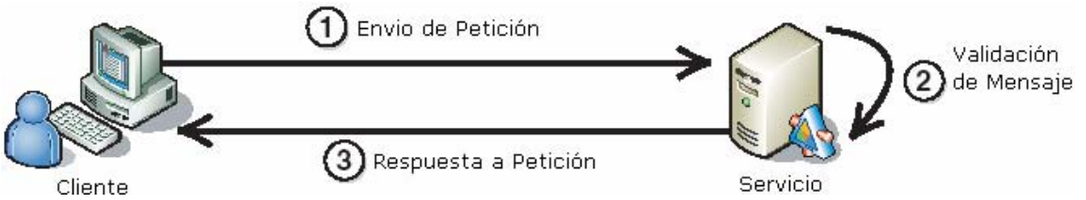


Figura 6.- Modelo de validación en los mensajes

1) El cliente envía la solicitud al Servicio. El proceso de validación está oculta para el cliente.

2) El servicio valida el mensaje. La validación lógica realiza un determinado número de chequeos para validar el mensaje. Algunas de ellas puedes ser:

   - Verificar el tamaño de la consulta, comparándola con el tamaño máximo permitido para la recepción del mensaje
   - Verificar la firma para asegurarse que el mensaje no ha sido cambiado en su tránsito.
   - Verificar que el mensaje se encuentre bien formado y conforme al esquema predefinido, con datos de tipo aceptable y el rango de valores.

3) El servicio procesa la consulta y entrega la respuesta al cliente.

Algunas Consideraciones en la Validación de los Mensajes

- La validación lógica no considera el formato binario. Si se desea implementar este tipo de validación se debe ser capaz de reconocer cada tipo de dato en este formato, para poder así evitar la data maliciosa.

- Utilizar un parser de validación para verificar el ingreso de mensajes, constituye un incremento en el tiempo de proceso de CPU, por lo que se debe estar dispuesto a afrontar ese costo.

- Para evitar la implementación de la validación lógica en el servidor se puede utilizar un intermediario. Esto puede permitir que varios WS utilicen el mismo intermediario para sus validaciones y así el WS sólo se dedica al procesamiento del mensaje. Sin embargo, esto también implica el hecho de que al utilizar un único punto de validación, haya, también, un único punto de falla.

2.5.3 PROTECCIÓN DE EXCEPCIONES

Si un cliente esta ejecutando un servicio Web, puede ocurrir que dentro del sistema donde se encuentre el servicio, ocurra una excepción, pero esta información de la excepción generada debe manejarse dentro del sistema implementador y el cliente no tiene que enterarse de lo que ocurrió. El cómo lograr esto se detalla a continuación.

Existen muchos casos por los que se puede producir una excepción. Estas pueden suceder por una mala implementación del WS, un mal funcionamiento del o los recursos que contienen el servicio, un atacante buscando información, etc.

También se pueden mencionar factores por lo que no es bueno que la excepción sea conocida por el cliente, como por ejemplo, los detalles de la excepción puede contener información que un atacante puede utilizar para explotar los recursos, como string de conexión, nombre de los servidores, consultas SQL, etc.

Para solucionar este problema, para los WS existe la posibilidad de implementar el Exception Shielding, cuya funcionalidad es sustituir las excepciones inseguras por excepciones seguras por diseño y sólo retorna estas al cliente.

El exception shielding tiene 2 participantes:

Cliente: es quién accede el WS.

Servicio: es el servicio ofrecido por el WS.

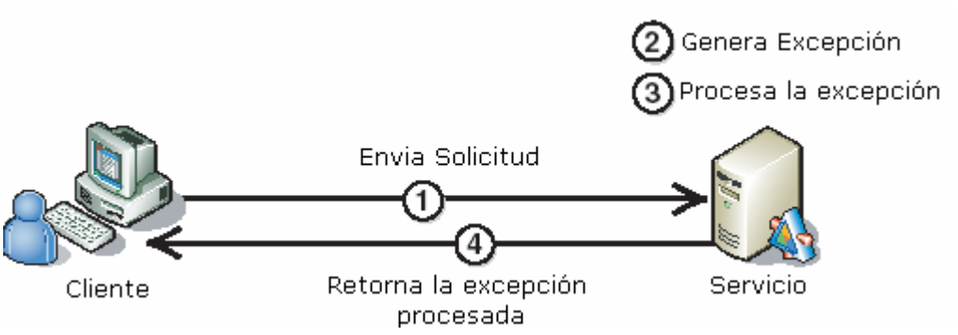


Figura 7.- Modelo de WS que implementa Exception Shielding

1) El cliente envía la solicitud al servicio.

2) El servicio intenta procesar la solicitud, pero se genera una excepción.

3) La lógica del Exception Shielding procesa la excepción. Si la excepción es segura por diseño, la excepción es considerada “sana” y es devuelta al cliente sin modificaciones, de lo contrario, la excepción es reemplazada por una excepción que es segura por diseño y, recién ahí, es retornada al cliente.

4) La excepción procesada es enviada al cliente.

Algunas Consideraciones en el Exception Shielding

- El manejo de las excepciones de toda la aplicación debe ser manejada en el código base, para así evitar que el cliente consiga acceso a ella.

- Una excepción no controlada, puede solaparse con una excepción controlada, con esto, puede resultar que el administrador no tenga la información certera del error que ocurre o que al cliente le llegue información errónea. Para esto se debe asegurar que todas las excepciones sean controladas.

- El modelo “denegado” es similar al modelo “permitido” del Exception Shielding, pero el primero es considerado mas inseguro debido a que las excepciones no presupuestadas no están “sanas”

2.6 MODELOS DE PROTECCIÓN DE MENSAJES

Los WS envían y reciben mensajes en texto plano sobre protocolos estándar de Internet como HTTP. Es por esto que el mensaje se ve frecuentemente amenazado por ataques de personas o entidades que quisieran realizar propósitos maliciosos con esta información. Utilizando la protección en los mensajes se puede evitar que esto ocurra.

Para el buen funcionamiento de la protección de los mensajes, se deben cumplir 3 requisitos importantes, estos son:

Integridad de la Data: Verificar que el mensaje no sea cambiado en el trayecto

Autenticación del origen de la Data: identifica y valida el origen del mensaje, además de verificar que no haya sido cambiado en el trayecto.

Confidencialidad de la Data: es la encriptación de la data del mensaje, para que entidades no autorizadas no puedan leer su contenido.

A continuación se especifica cada uno de estos requisitos.

2.6.1 INTEGRIDAD EN LOS DATOS

Si el remitente de un mensaje SOAP se firma con un token de seguridad, el receptor de dicho mensaje podrá estar relativamente seguro de que el mensaje recibido es el mismo que el enviado. Es decir, que el mensaje no ha sufrido modificaciones durante el envío. Esta comprobación de **integridad** se puede llevar a cabo porque el remitente incluye un **algoritmo implícito** de XML Digital Signature junto con el mensaje SOAP.

Un algoritmo implícito es una huella digital binaria de longitud fija de un mensaje. En el extremo receptor el WS crea un algoritmo implícito utilizando el mismo algoritmo y lo compara con el algoritmo implícito empaquetado junto con el mensaje. Si el mensaje ha sufrido el más leve cambio, los dos algoritmos implícitos, también denominados hashes, no coincidirán y el receptor sabrá que deberá rechazar la solicitud.

Otra ventaja de la firma de mensajes es la capacidad de satisfacer una necesidad empresarial común conocida como **no repudio**, la cual permite al receptor probar que el mensaje ha sido firmado realmente por una entidad específica. El no repudio se puede llevar a cabo porque el algoritmo implícito del mensaje se combina matemáticamente con un valor **SecurityToken** que sólo debe conocer el remitente. El receptor podrá comprobar el algoritmo implícito utilizando un valor asociado con el **SecurityToken**. Por ejemplo, si el **SecurityToken** es un certificado X.509, el valor del remitente será la clave privada del certificado y el receptor comprobará la firma utilizando dicha clave. A la hora de elegir un SecurityToken para el no repudio, en la mayoría de los casos se recomienda seleccionar un token específico de usuario (por ejemplo, **UsernameToken**, o X509SecurityToken o **KerberosToken** específicos de usuario) en lugar de un token específico de un equipo.

2.6.2 VERIFICACIÓN O AUTENTICACIÓN DEL ORIGEN DE LOS DATOS

La data, frecuentemente, pasa entre el cliente y el proveedor a través de uno o más intermediarios. Los datos contenidos en el mensaje de la petición, generalmente influye en el comportamiento del WS, esto es un riesgo, ya que un atacante puede manipular los

datos mientras el cliente y el WS los mantengan en tránsito. El tipo de manipulación que podría ocurrir es: modificación de la data del mensaje, sustitución de credenciales o cambiar el origen del mensaje. Un mensaje alterado puede provocar que los datos se comporten de manera inesperada. Otro problema que puede surgir es el hecho de que un cliente solicito realizar una acción “x” con un WS y luego pueda negar haberlo solicitado, para esto también se debe mantener una “constancia” de las acciones realizadas y quien las solicitó (no-repudio).

Uno de los pasos a seguir es utilizar la firma digital, mediante la cual se posee la evidencia de que un cliente ha solicitado una acción relacionada con la manipulación de datos.

Existen dos tipos de firma que pueden ser utilizadas para enviar un mensaje: la simétrica y la no-simétrica.

Firma Simétrica: una firma simétrica, comúnmente es conocida como Código de Autenticación de Mensaje (MAC, Message Authentication Code). Un MAC es creado con un cheksum (resumen del mensaje codificado) computarizado con el contenido del mensaje y las credenciales.

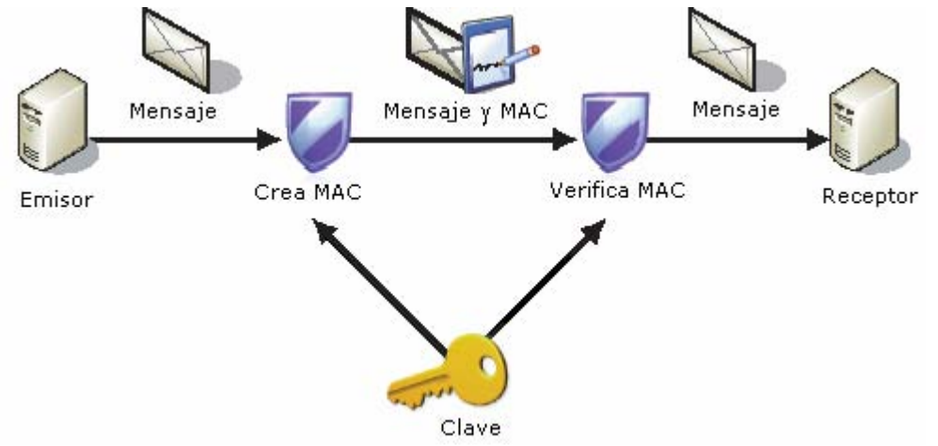


Figura 8.- Representación de firma simétrica

Al enviar un mensaje con la clave, la integridad de la data y la confidencialidad deben ser provistos por el emisor del mensaje, sin embargo, la firma simétrica no es utilizada para evitar el no- repudio, ya que la clave puede ser conocida por múltiples entidades.

Firma Asimétrica: Una firma asimétrica es procesada con 2 diferentes claves o llaves, una es utilizada para crear el mensaje y otra para verificar la firma, acá aparece el concepto de las llaves públicas y las privadas. La llave pública generalmente está disponible y puede ser enviada con el mismo mensaje, en cambio la privada es guardada en secreto por el propietario y nunca es enviada con el mensaje. Este tipo de firma es conocida como firma digital.

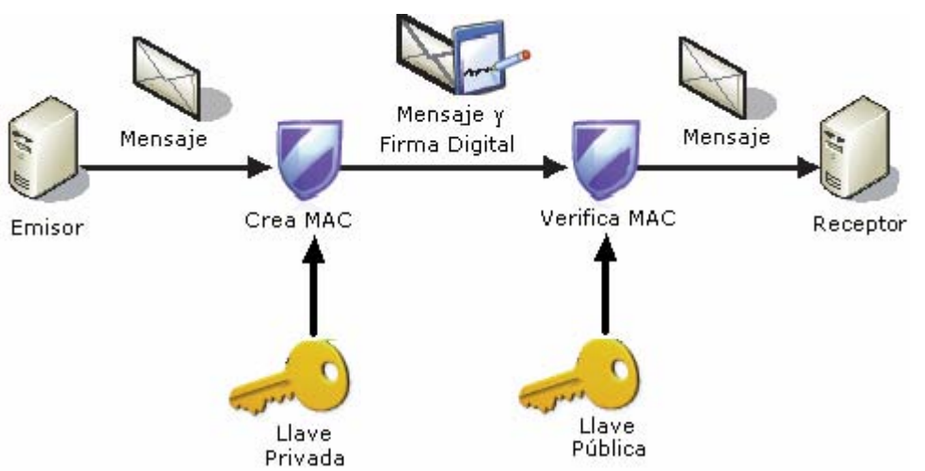


Figura 9.- Representación de firma asimétrica

En la información que envía el emisor esta obligado a enviar su llave pública, esto lo realiza a través de un certificado (X.509). Así, con el certificado, el receptor verifica con la llave pública la autenticidad del mensaje.

Esta sí es una opción para evitar el no-rechazo, ya que llave privada sólo la debe poseer el propietario del mensaje y así se hace fácil verificar la propiedad.

Algunas Consideraciones en la Verificación

- Si el mensaje es firmado, se debe asegurar que la llave pública se encuentre dentro de los datos y ambas se encuentren encriptadas. Así se evitan los ataques.

- Si gran cantidad de datos es encriptada con la misma llave simétrica un atacante puede interceptar varios mensajes e intentar descifrar la criptografía u obtener la llave. Para evitar esto se debe implementar la encriptación basada en sesiones, cuya vida útil es relativamente corta. Estas llaves son cambiada cada cierto tiempo o cada cierta cantidad de información.

- Los algoritmos para codificar que se utilizan, debieran ser los que se han patentados y poseen prestigio a través de los años, ya que si se utiliza un algoritmo “novedoso” se puede caer en detalles fácilmente accesibles para los atacantes.

2.6.3 CONFIDENCIALIDAD EN LA DATA

La confidencialidad de los datos asegura que el mensaje sólo sea leído por las partes directamente involucradas en él.

La mejor solución para evitar el problema de que un mensaje sea visto o alterado de alguna forma en su tránsito en la red, es el poder cifrar su contenido o al menos parte de él. Para esto, los datos desencriptados, conocidos como texto plano se encriptan en su origen y al llegar a destino son desencriptados. Esto se logra a través de un algoritmo de encriptación y una llave criptográfica [MIC05].

En la confidencialidad de la data se identifican los siguientes actores:

Remitente: Es el creador del mensaje. El cliente puede enviar un mensaje a un WS y un WS puede enviar una respuesta a un cliente. En ambos casos, cada uno de ellos, seria un remitente.

Recipiente: es la entidad que recibe el mensaje del remitente, al igual que el caso anterior, este rol lo pueden cumplir tanto el cliente como el WS.

Se puede utilizar dos métodos de encriptación, la simétrica (anexo) y la asimétrica. Ambas contienen el mismo proceso básico, pero sus características son distintas.

Algunas Consideraciones en la Confidencialidad

- La encriptación no previene que los mensajes sean atacados. Es por esto que es aconsejable implementarlo junto con la autenticación de su origen.

- Al igual que en la verificación, se debe tener en cuenta la cantidad de datos enviados.

- Cada país, frecuentemente posee estándares diferentes para la protección y confidencialidad de los datos. Y existen entes reguladores, para ayudar a cumplirlos.

2.7 MODELOS DE AUTENTICACIÓN

Los patrones de autentificación tienen como objetivo guiarnos en las decisiones de la aproximación con que nuestro sistema enfrentará el problema de la autentificación de los clientes que quieren usar los recursos expuestos por nuestros servicios.

Las 2 alternativas de seguridad que se estudiarán en este capítulo buscan solucionar el problema de identificación del cliente que hace uso de los recursos del Servicio Web.

Para entender en profundidad la diferencia entre autenticación e identificación se definirán los siguientes conceptos.

La **autenticación** es el proceso de identificación personal, utilizando las credenciales individuales. La autenticación frecuentemente ocurre luego de la identificación.

La **autorización** es el proceso que determinará que cliente autenticado tendrá acceso a un determinado recurso.

Las **credenciales** son una prueba que confirma la identidad del cliente. Una licencia de driver es un ejemplo de credencial en la vida real.

La **identificación** representa el uso de de un "identificador" que permite al sistema reconocer un particular sujeto y/o distinguirlo de otros.

2.7.1 AUTENTICACIÓN DIRECTA

Cuando el cliente y servidor tienen una relación de confianza que les permite intercambiar y validar credenciales donde se incluye el password, el patrón de autentificación directa aplica perfectamente. Estas credenciales son validadas contra un almacén de identidad (Base de datos). [MIC05]

En la autenticación directa se identifican los siguientes actores:

Cliente: es quién accede el WS. Debe presentar las credenciales para acceder el WS

Servicio: es el servicio ofrecido por el WS, el cual requiere autenticación para autorizar al cliente.

Almacén de Identidad: almacena las credenciales del cliente para identificarlo en el dominio.

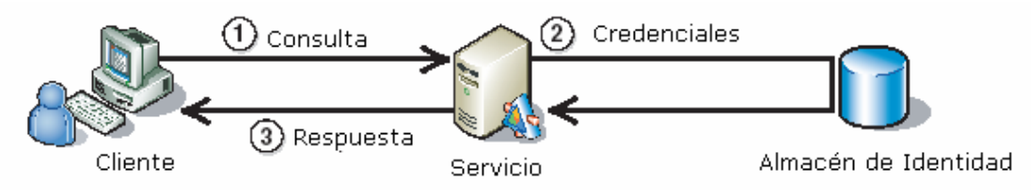


Figura 10.- Representación de la autenticación directa

Los pasos a seguir en la autenticación directa son los siguientes:

1) El cliente envía una petición al WS, adjuntando las credenciales.

2) El WS valida las credenciales contra el almacén y otorga los permisos correspondientes al cliente.

3) El WS, dependiendo de los permisos del cliente, retorna la respuesta al cliente.

Los beneficios de utilizar autenticación directa

- Es un sencillo método para autenticar clientes, sin la necesidad de un “agente de autenticación”.
- Si las credenciales entre el solicitante y el servicio se ven comprometidos, sólo la relación entre estas dos partes se ve comprometida y no el modelo entero.

- Para resguardar la seguridad el servicio puede forzar a que el solicitante se identifique cada vez que va por un WS, ya que si éste guardara sus credenciales (password) podría haber un serio riesgo.

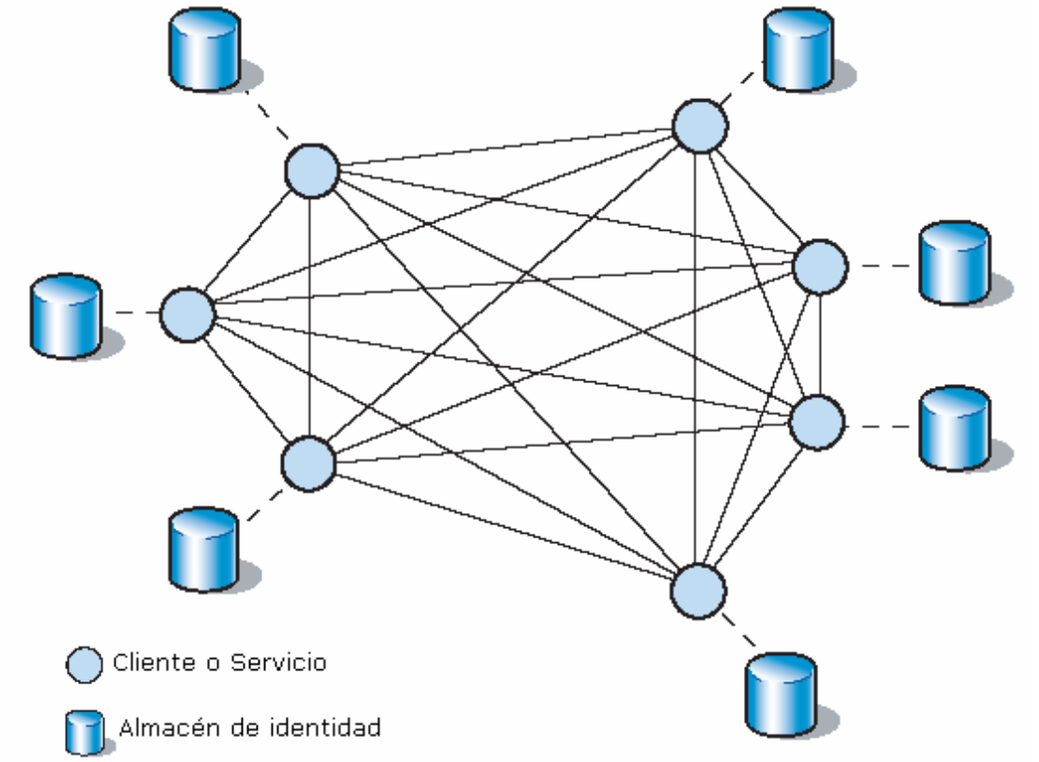


Figura 11.- Ejemplo de comunicación en la autenticación directa

Cada línea mostrada en la figura, representa la relación de secreto (credenciales) entre el cliente y el servicio, considerando varios almacenes de identidad y varios servicios y clientes. En una red así, se puede ver que el desafío de manejar y distribuir secretos se torna muy complicado.

Si un cliente solicita un WS frecuentemente, el uso de autenticación directa puede aumentar la latencia, ya que el WS, típicamente, accede a un almacén de identidad remoto.

Algunas Consideraciones en la Autenticación Directa

- Un atacante podría interceptar las credenciales del cliente. Esto podría ocurrir si éste no estuviera protegido mientras se encuentre en tránsito. Para esto es mejor cifrar las credenciales.

- Es posible que ataques al canal entre el cliente y el Servidor Web puedan obtener las credenciales del cliente y atacar el servidor usando esas credenciales. Una precaución importante a ser considerada debe ser el cifrado dentro del almacén

- Otro problema podría ser el hecho de que al consultar frecuentemente un WS o WSs en un dominio, el desarrollador haya decidido implementar un resguardo en caché del username y el password, lo que también se presenta como un riesgo para el servicio o sistema. Esto se podría mejorar solicitando autenticación cada vez que se solicite un WS.

2.7.2 AUTENTICACIÓN CON AGENTES

El contexto en el que se puede utilizar es para que un cliente al necesitar acceder un WS, éste como medida adicional posee un intermediario antes de que la solicitud llegue al servicio en si, este intermediario será el encargado de auditar y autorizar al solicitante.

En este intermediario deben confiar tanto el cliente como el servicio. El intermediario otorga un token de seguridad al cliente. El cliente, al llegar al servicio, debe presentar sus credenciales, incluyendo el token de seguridad.

En la autenticación con agentes se identifican los siguientes actores:

Cliente: es quién accede el WS. Debe presentar las credenciales para acceder el WS

Servicio: es el servicio ofrecido por el WS, el cual requiere autenticación para autorizar al cliente.

Agente de Autenticación: autentica al cliente y mantiene el control sobre la seguridad. También atestigua para el cliente otorgándole un token de confiabilidad.

Almacén de Identidad: almacena las credenciales del cliente para identificarlo en el dominio.

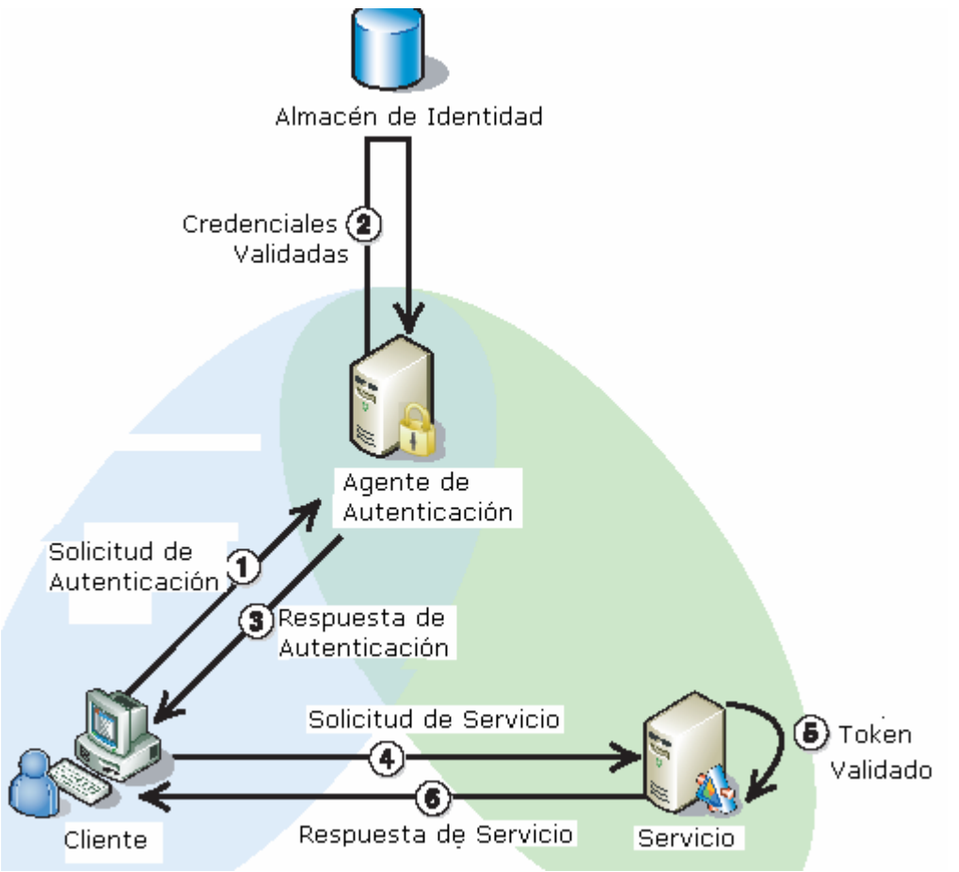


Figura 12.- Representación de la autenticación con agentes

1) El cliente envía una solicitud de autenticación al agente

2) El agente contacta el almacén de identidad para validar las credenciales del cliente.

3) El agente o intermediario responde al cliente, y si la autenticación resulta exitosa, le entrega un token o testimonio de seguridad. Puede utilizar el token por un período de tiempo determinado por el agente.

4) Un mensaje de solicitud de parte del cliente es enviado al servicio. El cliente utiliza el token para autenticarse con el servicio.

5) El servicio autentica la solicitud, validando el token de seguridad enviado con el mensaje.

6) El servicio retorna la respuesta al cliente.[MIC05]

Existen diferentes tipos de agentes de autenticación. Los más conocidos son X.509, Kerberos y Web Services Security Token Service (STS)

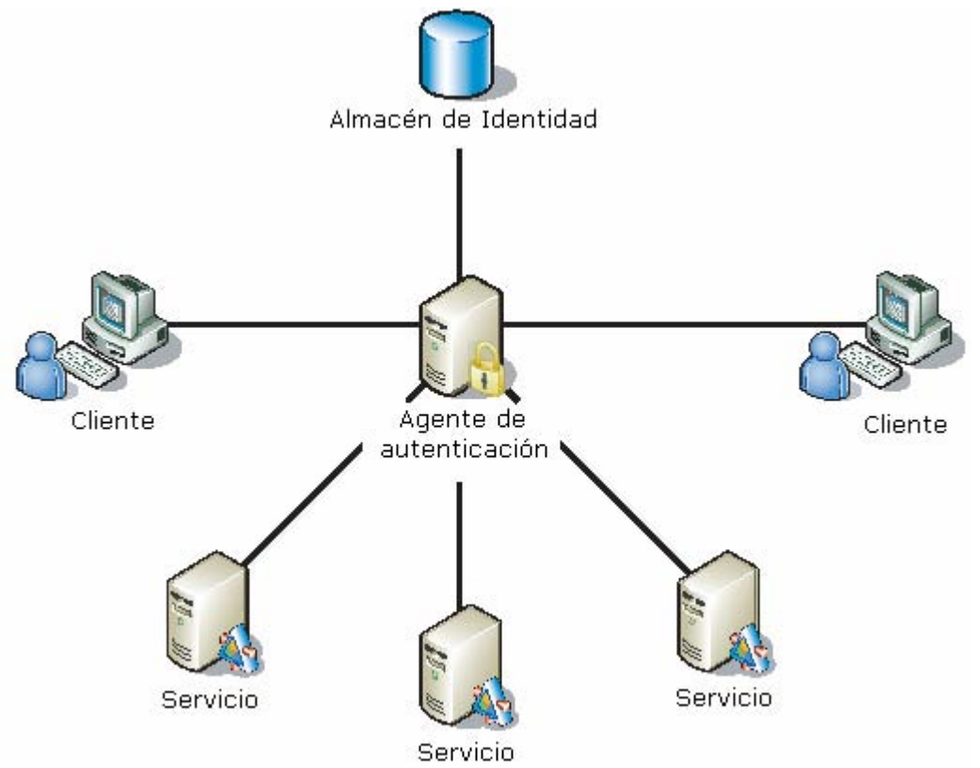


Figura 13.- Ejemplo de autenticación con agentes

Algunos de los beneficios de utilizar agentes son

- La autenticación se maneja centralmente.
- Las soluciones con mas fáciles de mantener, ya que si se agrega o elimina un usuario, esta actualización se hace centralizadamente.
- Si el usuario debe ingresar a un nuevo WS o a uno al que no ha ingresado antes, éste no debe cambiar nada en su configuración.

Algunas Consideraciones en la Autenticación con Agentes

- Los token de seguridad contienen data, la cual deben ser protegidas en su tránsito, para ello se puede utilizar seguridad en el nivel de transporte.

- El token de seguridad debe ser firmado por el publicador del WS, para evitar que los atacantes puedan crear tokens falsos.

2.7.3 AUTENTICACIÓN UTILIZANDO KERBEROS

El Kerberos fue originalmente diseñado por el Instituto de Tecnología de Massachussets (MIT) para permitir mejorar la seguridad en las redes distribuidas [URL3]. Este servicio proporciona una manera para que las entidades, como los usuarios y los servicios o los clientes y los servidores puedan autenticarse unos al otro.

También genera las llamadas llaves de sesión que serán compartidas entre un cliente y un servidor

Un servidor kerberos se denomina KDC (Kerberos Distribution Center) y provee dos servicios fundamentales: el de autenticación (AS, Autentication Service) y el de *tickets* (TGS, *Ticket Granting Service*). El primero tiene como función autenticar inicialmente a los clientes y proporcionarles un *ticket* para comunicarse con el segundo, el servidor de *tickets*, que proporcionará a los clientes las credenciales necesarias para comunicarse con un servidor final que es quien realmente ofrece un servicio. Además, el servidor posee una base de datos de sus clientes (usuarios o programas) con sus respectivas claves privadas, conocidas únicamente por dicho servidor y por el cliente al que pertenece.

La arquitectura de *Kerberos* está basada en tres objetos de seguridad: Clave de Sesión, *Ticket* y Autenticador. [URL7]

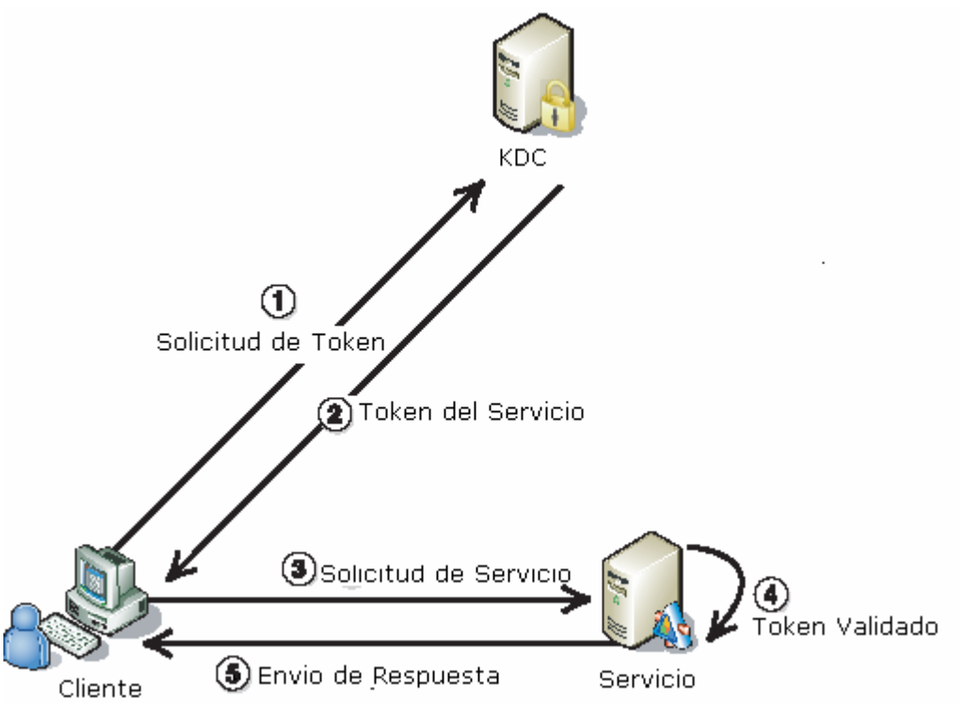


Figura 14.- Representación de la Autenticación con Kerberos

- La **clave de sesión** es una clave secreta generada por *Kerberos* y dada a un cliente para uso con un servidor durante una sesión; no es obligatorio utilizarla en toda la comunicación con el servidor, sólo si el servidor lo requiere. Las claves de sesión se utilizan para minimizar el uso de las claves secretas de los diferentes agentes: éstas últimas son válidas durante mucho tiempo, por lo que es conveniente para minimizar ataques utilizarlas lo menos posible.

- El **token o ticket** es un testigo dado a un cliente del servicio de *tickets* de *Kerberos* para solicitar los servicios de un servidor; garantiza que el cliente ha sido autenticado recientemente. Este *ticket* incluye el nombre del cliente, para evitar su posible uso por impostores, un periodo de validez y una clave de sesión asociada para uso de cliente y servidor. *Kerberos* siempre proporciona el *ticket* ya cifrado con la clave secreta del servidor al que se le entrega.

- El **autenticador** es un testigo construido por el cliente y enviado a un servidor para probar su identidad y la actualidad de la comunicación; sólo puede ser utilizado una vez. Este autenticador contiene, cifrado con la clave de la sesión, el nombre del cliente y un *timestamp* (fecha y hora de creación).

Los participantes en esta relación son básicamente los mismos mostrados en la autenticación con agentes, ya que sigue el mismo modelo, pero las propiedades que la diferencian de los demás son los siguientes:

- El protocolo del Kerberos apoya la noción de la renovación del ticket, pero no los renueva automáticamente. Por defecto, los tickets del Kerberos tienen un curso de la vida fijo de 8 horas.

- El KDC no termina un ticket del servicio cuando un cliente autenticado ha terminado de comunicarse con un servicio. Lo que hace es dejar el ticket expirar hasta el final de su curso de la vida normal. Si el ticket expira durante la comunicación con un servicio, la expiración no afectará operaciones actuales. No se notifica a los clientes cuando un ticket está a punto de expirar.

Alguna de las ventajas de usar la autentificación mediante agentes utilizando Kerberos incluye:

- El protocolo Kerberos proporciona las capacidades de SSO (Single Sign On), que permiten que un cliente se autentique sólo una vez por sesión de la conexión.

- El protocolo Kerberos tiene amplia aceptación como protocolo de la autentificación y funciona en la mayoría de las organizaciones grandes que tienen una infraestructura centralizada de autentificación.

- El protocolo del Kerberos se integra con el sistema operativo de Windows (Windows 2000 en adelante). Esto permite al sistema operativo proporcionar capacidades adicionales, tales como personificación del usuario/delegación, autorización, y revisión.

- Kerberos apoya la autentificación mutua cuando un servicio envía una respuesta que contenga los datos cifrados con la llave compartida de la sesión.

Algunas Consideraciones al utilizar Kerberos

- Los clientes deben mantener sus llaves principales secretas, debido a los consecuencias obvias que pueden producirse si son obtenidas por un intruso malintencionado.

- Con el protocolo del Kerberos, los ataques pueden ocurrir contra los mensajes encriptados con una contraseña equivalente. (Para la autentificación del cliente, ésta es la contraseña del cliente. Para la autentificación del servicio, ésta es la contraseña de la cuenta del servicio.) El protocolo Kerberos utiliza esta llave derivada para encriptar datos en la petición de la autentificación.

- Para un correcto funcionamiento se ha de disponer en todo momento del servidor *Kerberos*, de forma que si la máquina que lo alberga falla, la red se convierte inutilizable; obviamente esto es una contradicción con lo que nos dice la teoría de sistemas distribuidos, donde se recalca el uso de la distribución para mantener la disponibilidad del sistema, intentado que si un equipo falla el resto pueda seguir funcionando, si no a pleno rendimiento, al menos correctamente.

- Otro potencial problema de seguridad es el uso de TimeStamps como pruebas de frescura en Kerberos. Esto obliga a que todas las máquinas que ejecutan servicios autenticados mantengan sus relojes mínimamente sincronizados (con desfases máximos de pocos segundos), con todo lo que esto implica.

2.7.4 AUTENTICACIÓN UTILIZANDO X.509

El certificado digital es un documento firmado por una autoridad certificadora, el cual principalmente posee el nombre del sujeto y su llave pública.

La primera versión de este certificado apareció el año 1988, siendo la propuesta más antigua para una infraestructura de clave pública.[URL8]

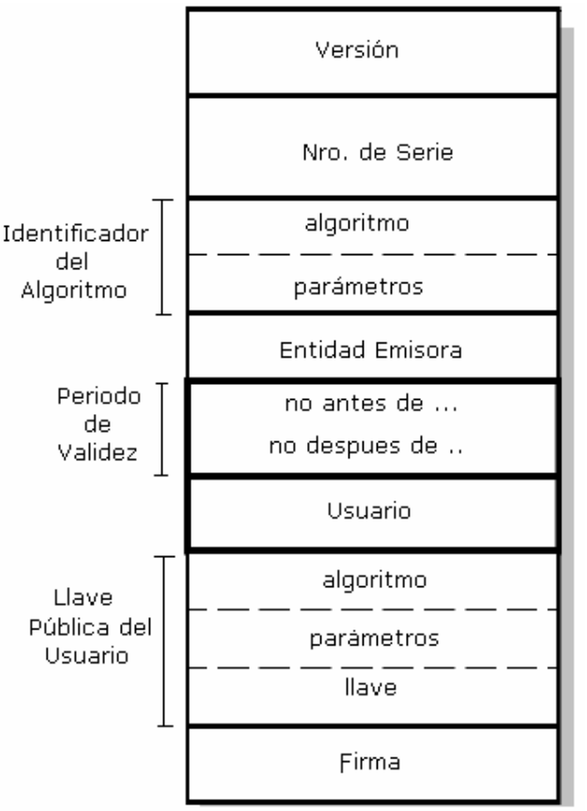


Figura 15.- Formato del Certificado X.509

Los principales actores en los certificados X.509 son:

Autoridad Certificadora (CA): es la autoridad certificadora y utiliza certificados debidamente validados.

Almacén de Certificados: aquí es donde los certificados están guardados.

Cliente: es quién accede el WS. Debe presentar las credenciales para acceder el WS.

Servicio: es el servicio ofrecido por el WS, el cual requiere autenticación para autorizar al cliente.

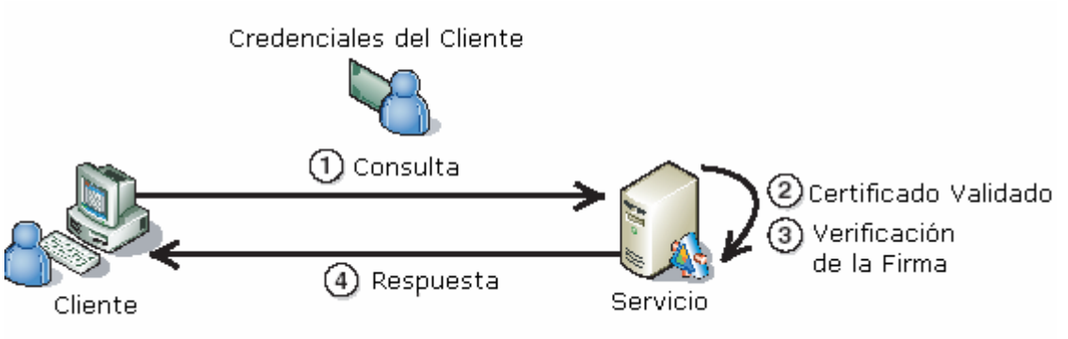


Figura 16.- Representación de la autenticación con X.509

Los pasos de la autenticación utilizando X.509 son las siguientes:

1) El cliente envía un mensaje al servicio, el cual incluye las credenciales del cliente con la llave privada, la cual es una copia de la llave pública.

2) El servicio valida el certificado: Verificando que el certificado no ha expirado, verificando que el certificado, internamente, sea consistente, entre otras validaciones.

3) El servicio utiliza la llave pública para verificar la firma digital.

4) El servicio envía la respuesta al cliente.

Tanto el cliente como el servidor pueden ser configurados para recibir al otro que posea estos certificados.

Estructura de un certificado X.509

1. El primer campo, es el subject, que contiene los datos que identifican al titular. Estos datos están expresados en notación DN (Distinguished Name), donde un DN se compone a su vez de diversos campos, siendo los más frecuentes los siguientes; CN (Common Name), OU (Organizational Unit), O (Organization) y C (Country). Un ejemplo para identificar un usuario mediante el DN, es el siguiente: CN=david.comin O=Safelayer, OU=development, C=ES. Además del nombre del titular (subject), el certificado, también contiene datos asociados al propio certificado digital, como la versión del certificado, su identificador (serialNumber), la CA firmante (issuer), el tiempo de validez (validity), etc.

2. En segundo lugar, el certificado contiene la clave pública, que expresada en notación ASN.1(Abstract Syntax Notation One), consta de dos campos, en primer lugar, el que

muestra el algoritmo utilizado para crear la clave (ej. RSA), y en segundo lugar, la propia clave pública.

3. Por último, la CA, ha añadido la secuencia de campos que identifican la firma de los campos previos. Esta secuencia contiene tres atributos, el algoritmo de firma utilizado, el hash de la firma, y la propia firma digital

Los certificados cumplen dos propósitos particulares.

Identificación de servidor

Cuando conectamos con una Web, ¿cómo estamos seguros de que es la Web que queremos? El mundo Web e Internet en general son vulnerables a ataques maliciosos, de forma que un atacante que esperara obtener cierto beneficio, podría hacernos creer que entramos en una página cuando realmente estamos yendo a la que a él le interesa.

Es por esto que cuando conectamos con la página de nuestro banco, la conexión es "HTTPS" y no "HTTP", donde la "S" significa que estamos usando un protocolo de seguridad denominado SSL. En este protocolo, el servidor presenta un certificado emitido por una CA que nuestro navegador debería conocer (debería estar instalada en él). El navegador verifica la validez y autenticidad de este certificado, que incluye la dirección de la página. Si todo es correcto, el navegador accede a la página sin más. Si falla algún elemento, el navegador advierte que hay un problema de seguridad.

Identificación del cliente.

El cliente puede identificarse ante una página Web mediante el par (usuario, contraseña). Es un procedimiento habitual, que presenta algunas ventajas, como la facilidad de transporte (basta con recordarlos) y la facilidad de implantación en los sistemas. No obstante las contraseñas son vulnerables y por ello deben renovarse con frecuencia, lo que lleva a olvidos frecuentes. Además no son universales, es decir que para autenticarnos en cuatro Webs solemos necesitar cuatro contraseñas.

La autenticación por certificados es, en cambio, un sistema basado en algo que poseemos y por su seguridad tiene un periodo de validez prolongado. Además un certificado expedido por una CA de ámbito internacional puede ser válido para autenticar en cualquier parte del mundo. La autenticación mediante certificados, aun siendo la más segura, se emplea todavía poco por la dificultad de transportar los certificados y sus llaves privadas.

Algunas Consideraciones al utilizar certificados X.509

- Es importantísimo salvaguardar la llave privada asociada al certificado X.509. Si se compromete la llave privada, la integridad del correspondiente certificado X.509 se viola, porque otra entidad además del cliente es capaz de generar las firmas digitales que representan la identidad del cliente. Si se compromete una llave privada, el CA puede revocar el certificado X.509, que lo hace llegar a ser inutilizable para el cifrado y las firmas digitales.

- El tiempo de vida de un certificado X.509 es considerablemente mayor que el de otros tipos autentificación con agentes. La mayoría expira a los minutos o las horas a partir de su inicio de vida útil, mientras que un certificado X.509 puede ser válido por varios meses.

- Sólo una copia de la llave privada de los certificados de X.509 del cliente debe existir cuando se utiliza para apoyar la no-renegación a través de firmas digitales. Esta llave privada debe ser accesible al cliente solamente.

Ejemplo de utilización de Certificado X.509

```
<S11:Envelope xmlns:S11="...">
    <S11:Header>
        <wsse:Security
            xmlns:wsse="..."
            xmlns:wsu="...">
            <wsse:BinarySecurityToken
                wsu:Id="binarytoken"
                ValueType="wsse:X509v3"
                EncodingType="wsse:Base64Binary">
                MIIEZzCCA9CgAwIBAgIQEmtJZc0…
            </wsse:BinarySecurityToken>
            <ds:Signature
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>…
                    <ds:Reference
                    URI="#body">…</ds:Reference>
                    <ds:Reference
                    URI="#binarytoken">…</ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>HFLP…</ds:SignatureValue>
```

```
                <ds:KeyInfo>
                    <wsse:SecurityTokenReference>
                    <wsse:Reference URI="#binarytoken" />
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
            </ds:Signature>
        </wsse:Security>
    </S11:Header>
    <S11:Body wsu:Id="body"
    xmlns:wsu="..."> …
    </S11:Body>
</S11:Envelope>
```

En este ejemplo se muestra un certificado embebido en un elemento <wsse:BinarySecurityToken>, referenciado por una URI con una firma.

El certificado esta incluido en el header WS – Security como un elemento <wsse:BinarySecurityToken> con el identificador binarytoken. Dentro de la firma definida por un elemento <ds:Reference> contenido en el elemento <ds:SignedInfo> se incluye el certificado de la firma referenciada mediante una dirección URI . <ds:KeyInfo> especifica la llave de la firma por medio del elemento <wsse:SecurityTokenReference> el cual contiene referencias al certificado por medio de la URI definida como #binarytoken.

2.7.5 AUTENTICACIÓN UTILIZANDO SECURITY TOKEN (STS)

Un WS necesitar autenticar clientes en un ambiente heterogéneo de tal forma que los controles de auditoria y autorización puedan ser implementados. La organización puede decidir utilizar la autenticación con agentes para proveer un común acceso a la infraestructura de comunicación para un grupo de aplicaciones.

La autenticación utilizando agentes se encarga de negociar la confianza entre el proveedor y el solicitante del servicio, quitando la necesidad de una relación directa.

Algunas de las causas por las que se hace necesario utilizar este tipo de autenticación puede ser:

- El cliente requiere que la autenticación sea implementada en una variedad de plataformas dentro de la organización, y la interoperabilidad es requerida dentro de estas plataformas.

- El ambiente de la organización puede estar protegidos por cortafuegos. El agente de seguridad debe ser capaz de poseer permisos que sean capaces de atravesar estos cortafuegos.

```
(1) <wsse:UsernameToken wsu:Id="MyToken">
(2)    <wsse:Username>MyNombre</wsse:Username>
(3)                 <wsse:Password Type="#PasswordDigest">
(4)                     weYI3nXd8LjMNVksCKFV8t3rgHh3Rw=="
(5)                 </wsse:Password>
(6)         <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
(7)     <wsu:Created>2004-09-20T01:34:44Z</wsu:Created>
(8) </wsse:UsernameToken>
```

Ejemplo sacado de [CAB05]

(1)El identificador del usuario es enviada en el elemento usernameToken, esto para que este token sea directamente referenciado de otros elementos en el mensaje SOAP.

(2) En el elemento UserName se encuentra la cuenta de usuario asignada.

[(3)-(5)] Debido a que el mensaje puede atravesar una conexión no segura, el password no es incluido en el username Token, sin embargo el digest (resumen) del password es transmitido, esto debido a que el proveedor del servicio y el administrador del sistema tienen acceso a la password real.

[(6)-(7)] El Nonce(cadena única que identifica la solicitud) y el TimeStamp son incluidos en el Username Token para prevenir ataques.

Como se ve la passsword no esta directamente puesta en el username token, esto para resguardarla.

CAPÌTULO 3

3 ANÁLISIS DE DESARROLLOS PRÁCTICOS REALIZADOS POR LA ALUMNA EN LOS DISTINTOS AMBIENTES EN QUE SE IMPLEMENTA UN WEB SERVICE

Debido a la experiencia profesional de la alumna, trabajando en empresas de desarrollo de software ha podido interactuar directamente con estos servicios Web, por lo que el conocimiento práctico obtenido en este ámbito se dará a conocer en actual capítulo de este trabajo de tesis.

Si bien, en cada capítulo del desarrollo de esta tesis hay aportes directos de la experiencia de la alumna, es acá donde se quiere dar mayor énfasis a lo aprendido, ya que con la ayuda de arquitectos de software, que han tenido la paciencia, y se han dado el tiempo de explicar el funcionamiento, se ha llegado a la redacción “comprensible” de la implementación de estos servicios dentro de las diferentes empresas cliente.

Primero se describirá cada escenario en los que, frecuentemente se implementa un WS, luego, se mencionará el mejor método, según expertos de Microsoft o mejor dicho, según los expertos que recomiendan buenas prácticas, pero que a la hora de implementar no se siguen al pie de la letra, para la implementación de la seguridad. Finalmente se analizará lo ya implementado o ayudado a implementar por la alumna, para así llegar a un nivel real de la utilización de WS en las empresas hoy en día.

3.1 WEB SERVICES PÚBLICOS:

Para poner un ejemplo de este escenario, podemos hablar de un distribuidor, el cual, posee catálogos en línea, para poder vender sus productos a los comerciantes. El comerciante a su vez posee una aplicación, la cual se comunicara con el WS expuesto por el distribuidor para poder tener acceso a los productos del distribuidor.

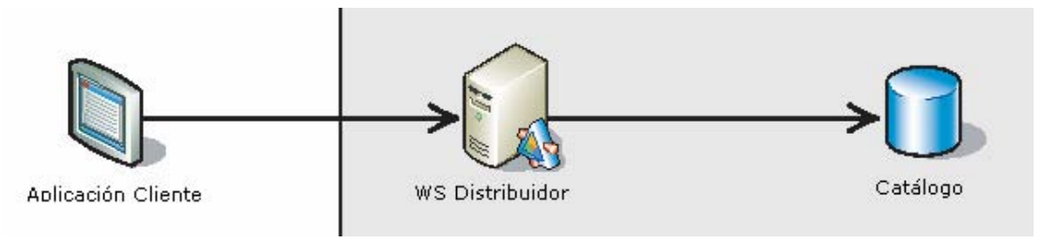


Figura 17.- Escenario de WS Público

Este escenario describe el método de seguridad que se utiliza a la hora de implementar un Web Service de acceso público.

a) El Servicio Web solicitante (cliente) requiere acceso directo al Servicio Web distribuidor,

b) el Servicio Web solicitante debe estar autenticado,

c) la data pasada entre el WS solicitante y el WS distribuidor, contiene algún tipo de información importante del primero, por lo que ésta debe ser protegida.

La mejor práctica a la hora de implementar la seguridad en este escenario se puede ver a continuación:

- El WS distribuidor, utiliza un certificado para establecer comunicación con la aplicación mediante HTTPS.
- La aplicación cliente pasa un token con su username al distribuidor para su autenticación.
- El WS distribuidor utiliza un subsistema seguro para acceder a la data.

3.1.1 INFORMACIÓN DE LA IMPLEMENTACIÓN REAL DE WS PÚBLICO

Se realizó una implementación para WS público en el sistema ChileProveedores, el cual debía obtener información acerca de los clientes en el Servicio de Impuestos Internos (SSI).

En la implantación de este sistema no fue necesario realizar la protección de los mensajes, ya que se trabajó con el canal de datos protegido.

Este servicio es la mayor prueba de interoperabilidad entre servicios ya que la aplicación, del SSI, la cual manejaba los datos que se necesitaban consultar, estaba implementada bajo la plataforma J2EE y nuestra aplicación se construyó en Visual Basic. NET

Para lograr la confianza en la operación se implementó la seguridad utilizando STS, Certificado Digital. La entidad certificadora fue CERTISUR S.A. [URL14] la cual posee los estándares adecuados y esta localmente reconocida como entidad de confianza.

Siempre la comunicación se realizó a través del puerto 80, por lo tanto no hubo conflictos en comunicación.

La comunicación se realizó enviado datos de entidades completas, por ejemplo, la entidad persona constaba de *nombres, apellidos, rut, fecha de nacimiento, dirección, comuna, etc.*, por lo que no se realizó una operación transaccional sobre los datos. Simplemente se enviaba la solicitud y se esperaba la recepción de los datos.

Los tiempos de respuesta de los datos siempre estuvieron entre los rangos permitidos, o sea, aproximadamente entre 3 y 10 segundos.

Si ocurría un error de cualquier tipo, por ejemplo de no esta activo el servidor, simplemente en la aplicación se enviaba mensaje diciendo que el servicio no estaba disponible, pero por medio de la aplicación se creaba un log de error (almacén de errores) en el que se informaba al administrador los problemas ocurridos. Ya que el servicio XML solicitante estaba implementado de manera tal que recibiera cualquier tipo de operación no presupuestada.

El único problema existente en esta implementación fue que el SSI se guardaba muy en secreto, el tiempo de vida que da a un token para establecer la relación de confianza, entonces el servicio, si quería realizar la operación por dos o mas veces consecutivas, se confiaba en que poseía los servicios necesarios. Pero de improviso ocurría que ya su token no era válido, por lo que debía volver al servidor emisor de token.

3.2 WEB SERVICES EN UNA INTRANET:

El modelo de Intranet que se indagará será el de una aplicación bancaria, debido a que es el mejor escenario que se puede presentar para representar la necesidad de la confidencialidad de la data.

Frecuentemente la aplicación bancaria es una aplicación Windows que accede directamente un WS, el cual, por su parte, accede una base de datos del banco para obtener la información.

Estas aplicaciones deben poseer, al menos, las siguientes características:

a) El usuario de la aplicación es un cliente representativo del servicio (CSR),

b) el CSR debe ser autenticado como un usuario válido para la aplicación bancaria y

c) las regulaciones bancarias requieren que las actividades que realiza el CSR deban ser auditadas.

Una de las mejores alternativas para implementar la seguridad en este tipo de aplicación sería de la siguiente manera:

Las credenciales del usuario son utilizadas para obtener un token de seguridad desde el Centro de Distribución de claves Kerberos (KDC) implementado en el Active Directory.

El token de seguridad es usado para enviar y encriptar mensajes enviados al servicio.

El token de seguridad es usado para obtener información adicional acerca del usuario desde el Active Directory.

3.2.1 INFORMACIÓN DE LA IMPLEMENTACIÓN REAL DE WS EN UNA INTRANET

La aplicación en la cual se trabajó directamente fue en un sistema para una compañía de seguros, los servicios ofrecidos eran muchos, y uno en especial que se haya ofrecido consistía en el cálculo de una prima de seguros.

El método de seguridad implementado también fue establecido en el canal de comunicación por medio de HTTPS, ya la única condición que se debía considerar era que los mensajes no fuesen leídos en tránsito.

El método de autenticación implementado fue la autenticación integrada de Windows, ya que la plataforma sobre la cual se ejecutaba el sistema (la compañía) era Windows, esto funciona a nivel de protocolo HTTP y no como un parámetro extra del SOAP.

El lenguaje utilizado tanto en el cliente como en el servidor fue C#, y no se presentaron problemas ni de rapidez ni de confianza.

3.3 BUSINESS TO BUSINESS (B2B) EN INTERNET:

Una aplicación de negocios, la cual se encarga de “abastecer” otras aplicaciones utiliza un WS interno para mejorar sus operaciones, a su vez puede necesitar acceder a otros WS externos proporcionado por otra compañía.

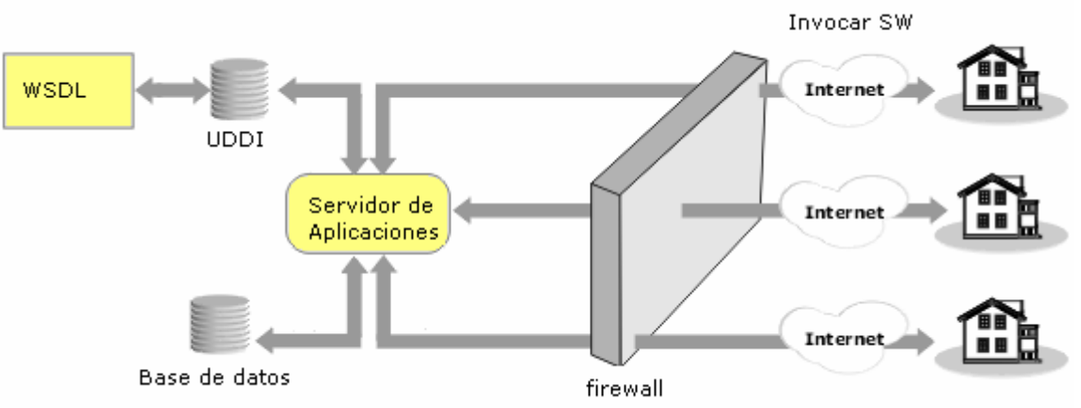


Figura 18.- Diagrama de comunicación B2B

Una aplicación de este tipo posee las siguientes características:

a) Una compañía cualquiera puede obtener información de un partner de negocios,

b) El partner debe asegurarse de que quién esta solicitando la información es realmente quien dice ser.

c) Debe ver en sus registros si realmente tiene la información que se busca.

d) Devolver una respuesta.

En este caso, el mejor candidato para la seguridad, sería:

Las credenciales del usuario son usadas para obtener un token de seguridad desde el Kerberos (KDC), implementado en el Active Directory.

El token de seguridad es usado para enviar y encriptar mensajes al servicio

En el distribuidor, el certificado X.509 es emitido e importado dentro de un apropiado almacén de certificados.

El certificado X.509 es usado para proveer autenticación mutua y confidencialidad en los datos y autenticación en el origen de los datos entre el proveedor del WS y el solicitante

3.3.1 EJEMPLO

```
(1). <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">

(2). <soapenv:Header>

(3). <wsse:Security soapenv:actor="http://www.jStartcustomer.com/actors#verifier"
       soapenv:mustUnderstand="1"
       xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
       <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

(4). <SignedInfo>

(5). <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

(6). <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

(7). <Reference URI="#sign_content_1043176028580">

(8). <Transforms>

(9). <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

(10). </Transforms>
```

(11). <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

(12). <DigestValue>FLuQTa/LqDIZ5F2JSaMRHSRuaiQ=</DigestValue>

(13). </Reference>

(14). </SignedInfo>

(15). <SignatureValue>

(16). kGlrr/WXKxID+JJkEXY+aGN5dy8GwbLFtB5Msll2/MhwdnO9wastJ0gLPzLy3oHL

(17). 7A8ggkM6PTzM7MdKoIAhe+xRHsamGucFJQRMrU+JQ4WATJt0bpdClwJy6mexT

(18). Su48mq1q5rM9YZh61P7UEUKt+EQ=

(19). </SignatureValue>

(20). <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

(21). <KeyValue>

(22). <RSAKeyValue>

(23). <Modulus>

(24). 2sW+eBjx5D2QMyr8ocZIZWNYHGf9zILPCTvhNV7dIe3l8ARepOA1ABFK2OMy

(25). pzb+Rb+nWQeo//yFz/28PmL63kdLiE72qmmuPa5NXaV9pJ4JKw86QdLhGGpFIRH

(26). 18Iugf3xLFwQEZqKYnblTUs7ftnTgW5r4HH492k=

(27). </Modulus>

(28). <Exponent>AQAB</Exponent>

(29). </RSAKeyValue>

(30). </KeyValue>

(31). <X509Data>

(32). <X509IssuerSerial>

(33). <X509IssuerName>OU=Java,O=IBM,L=Unk,ST=Oklaho,C=US</X509IssuerName>

(34). <X509SerialNumber>0</X509SerialNumber></X509IssuerSerial>

(35). <X509SubjectName>CN=John Doe</X509SubjectName>

(36). <X509Certificate>

(37). MIIB0TCCAToCAwDQYJKoZIhvcNAQEEB...CVVMxETAPBgNVBAgTCE9rbGFo

(38). b21hMRAwDgYDVQHEwdVbmsam3duMQwwCgYD…AsTBEphdmEwHhcNMDIw

(39). wOTI1MTAxMTQ4WjATMREwDwYDV…DEwhKb2huIERvZTCBnzANBgkqhkiG

(40). 9w0BAQEFAAOBjQAwgYkCYEA2sW+eBjx5..Gf9zYhB4XWILPCTvhNV7dIe3l8A

(41). pOA1ABFK2OMypzb+Rb+nWeo//yFz/2…qmmQuzuPa5NXaV9pJ4JKw86QdLhGGp

(42). FIRH18Iugf3xLFwQEZqKblTUs7ftnTgW5r4H….BgkqhkiG9w0BAQQFAAOBgQCs

(43). OD02WMoYcMR7Sqdb9oQyk7Nn4rQ5DBgZ5mxGGVzW…..jX1jalMvbHa9lnhPQmJi

(44). Ued923rza7fvdRG2CDalbWR3aPd5q0u3akP0/Ejb7z5o88heajCSgfR….3Oe+RBQgw8

(45). VuzbLApPnXiehowYuA==

(46). </X509Certificate>

(47). </X509Data>

(48). </KeyInfo>

(49). </Signature>

(50). </wsse:Security>

(51). </soapenv:Header>

(52). <soapenv:Body>

(53). **Contenido (datos) especifico de la aplicación**

(54). </soapenv:Body>

(55). </soapenv:Envelope>:

Ejemplo mensaje para B2B [URL14]

[(4)-(14)] Se describe el contenido cifrado del mensaje. A modo de buena práctica, se agrega un digest(resumen) con la firma digital para facilitar el rápido desempeño, ya que el mensaje del SOAP puede llegar a ser demasiado extenso el proceso de aplicar el algoritmo de la llave pública puede ser bastante costoso llevando a consecuencias negativas para el desempeño del WS.

Elementos del SignedInfo

(5) El digest se está generando en base a la canónica del mensaje. El algoritmo de canonización define que cosas van a ser importantes. Cuando se comparan 2 mensajes, se comparan 2 versiones canonizadas del mensaje.

(6) Indica que algoritmo se está utilizando para convertir el mensaje canonizado en una firma. En este caso se utilizó uno especial

(7) Indica el elemento de referencia. Incluye el algoritmo que fue utilizado para computar el digest y el resultado.

[(8)-(10)] Indica el algoritmo de transformación.

[(11)-(12)] Especifica el algoritmo para el digest y los valores del resultado.

[(15)-(16)] Contiene la firma, que es, en realidad, el valor del digest encriptado.

Keys

[(20)-(48)] introduce el concepto de clave. Transforma un texto legible a uno no legible, utilizando transformación matemática. Se utiliza la clave pública/privada conocida como asimétrica.

(20) Identifica el namespace a utilizar. Otorga la información al WS para validar el mensaje. El uso de RSA es para asegurar que el mensaje esta intacto

El hecho de que se implemente la seguridad a nivel del mensaje, cuando se trata de una aplicación de negocios, no es suficiente, ya que el riesgo es enorme para la empresa y/o el usuario al transmitir datos a través de la red. Es por esto que las empresas también deben estar provistas de un canal seguro de envío / recepción de mensajes, para esto se habilita cortafuegos y se otorgan permisos especiales a sus partners de negocios

CAPITULO 4

4 ANÁLISIS DE LOS WS A TRAVÉS DE LOS AÑOS

Para realizar este análisis se ha utilizado material, para ser más específico tesis, de años anteriores. Y para lograr nuestro objetivo se realizó una lectura comprensiva, objetiva y posteriormente un estudio de lo, antes, investigado.

La idea principal de esta investigación y/o comparación es vislumbrar la evolución que han tenido estos servicios en el poco tiempo que se compara. Basta con echar una rápida mirada sobre la documentación existente sobre este tema para comprobar su evolución.

4.1 COMPARACIÓN SEGÚN TRABAJOS ANTERIORES DE LA ESCUELA DE INGENIERÍA CIVIL EN INFORMÁTICA

Hoy en día el principal foco de desarrollo de los WS esta en el área comercial, sobre todo en los negocios, ya que gracias a estos ha logrado un gran grado de transaccionalidad, confidencialidad y seguridad de datos.

Hasta el año 2002 y hasta el 2003, se desconfiaba de esta tecnología, sobre todo a nivel empresarial, ya que la ausencia de técnicas de seguridad estándar era un obstáculo para su adopción [REY04].

Día en día, vemos como a nivel empresarial, esta tecnología va posicionándose en los primeros lugares cuando se realiza el análisis de implementación de algún sistema en

una empresa. La infraestructura de seguridad, también debe ser flexible, para apoyar la amplia variedad de políticas de seguridad requeridas por organizaciones diferentes.

WS-Security es el componente básico para la implementación de WS seguros. Hoy, los servicios Web, generalmente, confían en el apoyo del canal de transporte y sus funciones de seguridad (ejemplo HTTPS y la autenticación básica).

En pocas palabras, la infraestructura de seguridad para el uso de WS, debe ser desarrollada con una combinación de tecnologías existentes y protocolos de WS para mitigar los riesgos asociados con la mensajería punto a punto.

Y es este el procedimiento a seguir que ya tienen claro los expertos y lo que se ha realizado a nivel empresarial, es por esto que cada vez suman mas la empresas que adoptan los WS como una componente esencial de sus sistema y confían en su seguridad y la ayuda que les ofrecen. [CAB05]

También hasta el año 2003 esta tecnología estaba en desarrollo, como lo esta hoy en día también, pero la mayoría de los protocolos en los que se basaba aún no eran estándar. [REY04]

Hoy, se puede ver que la interoperabilidad entre las plataformas se ha conseguido de una forma muy eficiente, esto gracias a entidades de nivel internacional (OASIS, entre otras) que regulan los estándares que día a día están apareciendo en el mercado, y también

gracias a que las grandes empresas de área de desarrollo informático (IBM, Microsoft, entre otras) se han unido para lograr estandarizar herramientas comunes de utilización de estos estándares, esto llevado a la buena práctica de uso de estos estándares ha llevado al crecimiento sustancial de esta tecnología.

Para el desarrollo de este capítulo, del presente trabajo de tesis, se consultaron varias tesis relacionadas a la implementación de WS, pero no se encontraron antecedentes directamente relacionados a la seguridad de ellos, lo cual es el tópico esencial que se analiza.

Cabe mencionar, que en los últimos 3 años, las especificaciones de los WS han sido definidas, refinadas y hasta criticadas, para lograr alcanzar el grado de estandarización, utilización y confianza al que han llegado hoy. Las diferentes plataformas han lanzado kits de herramientas para la mejor y mas fácil implementación de estos estándares, dándole especial énfasis al tema de la seguridad.

4.2 ENFOQUE ACTUAL Y FUTURO DE LOS WEB SERVICES

Con el paso de los años, está tecnología se ha ido aposentando en el mercado, logrando en muy poco tiempo un sitial no menor a nivel mundial.

Pero aunque se crea que ya se ha dicho todo con respecto a este tema y ahora sólo se debe utilizar, la realidad es muy distinta, ya que día a día las entidades encargadas de estandarizarla sacan nuevos y mejores beneficios que refuerzan esta tecnología.

Algunos de los temas en los que actualmente se trabaja, se detallan a continuación:

Semántica

La Semantic Web Technology soporta la definición de la semántica del documento para que la búsqueda inteligente pueda extenderse utilizando para ello palabras claves y técnicas de mejora de información o localización de contenido Web relevante. [CHR05].

La Semantic Web Technology brinda una solución prometedora para divulgar la información y servicios sobre la World Wide Web, aumentando con las descripciones en una forma para que las computadoras puedan comprender y procesar. Esto ayudará a agentes de la Web a funcionar mejor y "hacer tareas" de parte de sus usuarios, como descubrimiento de información, integración, negociación del servicio y su composición.

La creación de conocimientos pueden puede ser automática, definiendo la pericia de una persona como un juego de reglas de la empresa.

La Semantic Web Technology es vista como la próxima generación de sistemas de Web, para proporcionar mejor recuperación de información, mejor atención y mejor interoperabilidad entre diferentes sistemas de información. Esta iniciativa es supervisada, actualmente por el W3C. Además a nivel empresarial es la alternativa que estas poseen para iniciar la correcta clasificación de la información que generan.

Plug-and-Play Interoperability

Dentro de la interoperabilidad en los WS aún quedan muchos pendientes. En el futuro los WS deben ser capaces de negociar un canal de comunicación compartido de manera que una aplicación posea sus propias credenciales de seguridad para realizar sus acciones sobre el proveedor del servicio. Esta interoperabilidad debería ser responsabilidad de un intermediario de WS o hasta de aplicaciones individuales; o en su defecto, la mensajería podría ser manejada hasta por el sistema operativo, quedando sólo como responsabilidad del sistema manejar sólo la lógica de negocio de la aplicación [GAI04]

Cloud Storage

Encontrar hardware económico ahora es una realidad, en Internet existen miles de grandes servidores que prestan servicios a millones de usuarios (decir cifras es para representar).

Ahora bien, si un WS, por alguna razón no controlada, falla, el solicitante se queda sin poder ejecutar su acción o recibir sus datos, es por este motivo que no es tan descabellado pensar en un almacenamiento alternativo de un WS, o sea, si por cualquier razón fallara el WS original, el solicitante debería poseer la “inteligencia” de redireccionar la solicitud de su servicio, y por supuesto, esto debería ser invisible a los ojos de los usuarios. [GAI04]

CAPITULO 5

5 COMPARANDO LOS WS

5.1 LAS OTRAS TECNOLOGIAS EN LA INFORMÁTICA DISTRIBUÍDA

5.1.1 NET REMOTING

Net Remoting es un protocolo para objetos ofrecidos por Microsoft y fue diseñado para ser utilizados en ambientes de intranets donde existe un alto acoplamiento entre los componentes o para Internet donde existe un bajo acoplamiento entre los componentes. [URL6]

Soporta múltiples protocolos de transporte. Por ejemplo, permite transferencias binarias de alta velocidad sobre TCP, también utiliza HTTP. NET Remoting permite definir el formato del paquete de información en SOAP en binario

A la hora de decidir cual de las dos tecnologías se puede utilizar para la implementación de un sistema de transporte de datos, se deben tener en cuenta algunos factores para realizar una buena elección. Algunas de ellas se pasan a mencionar a continuación:

Su ambiente

Si bien, una de las principales características con la que cuenta NET Remoting es su robustez, lo que más la desfavorece es que esta especialmente diseñado para .Net, no permitiendo la interoperabilidad con otros lenguajes de programación.

Programándolos

El WS tiene a su favor el esquema XML, y provee de un modelo de programación simple y de amplio alcance a través de múltiples plataformas .NET Remoting favorece el tipo runtime del sistema, y provee de un modelo de programación más complejo y de un alcance mucho más limitado. Esta diferencia esencial es el factor primario en la determinación de qué tecnología a utilizar.

Accesándolos

Si bien SOAP no establece HTTP como el protocolo de transporte, al implementar WS con ASP.NET el cliente sólo puede accesarlos utilizando este protocolo, ya que es el único protocolo de trasporte que HTTP soporta. NET Remoting otorga la flexibilidad de consultar objetos remotos o realizar acciones sobre él con cualquier tipo de aplicación incluyendo Windows Form, Windows Service, una aplicación de consola o cualquier tipo de proceso .Net.

Manejo de Estados

ASP.NET asume la arquitectura de los WS estáticas por defecto. O sea, cada vez que un cliente invoca un WS, se crea un nuevo objeto para realizar el servicio, este objeto es destruido cada vez que la llamada culmina.

Net Remoting soporta un conjunto de manejos de estados, puede o no realizar un conjunto de correlaciones para un mismo usuario, dependiendo del ciclo de vida del objeto que se elija.

Performance

En términos de funcionamiento Net Remoting ha demostrado ser más rápido que los WS siempre y cuando se utilice el canal TPC y el formateador binario [URL9]

Su seguridad

En un escenario sencillo, un WS puede utilizar la seguridad estándar de Internet funcionando con bastante eficacia, Si Net Remoting es implementado para ser utilizado con HTTP también puede hacer uso de esta seguridad. En cambio, si Net Remoting es implementado para TCP no asegura la seguridad, por lo que se debe crear seguridad personalizada. [URL10]

Resumiendo y tomando en cuenta todos los factores antes mencionados, no es tan difícil realizar una elección entre Net Remoting o Web Services a la hora de la elección de una arquitectura. Lo único que se debe tener claro es lo que se desea.

Se han de utilizar los servicios Web ASP.NET de forma predeterminada. Esta tecnología es más fácil de implementar y utilizar, ofrece un alcance potencial más amplio para las plataformas de cliente y el código de Proxy de cliente de estos servicios se puede invocar desde código ejecutado en un entorno de seguridad con las directivas de seguridad predeterminadas.

Ahora, si lo que se necesita es un modelo de objetos distribuidos más tradicional, sin necesidad de interoperabilidad con otras plataformas y se tendrá el control de la

configuración tanto de clientes como de servidores. Es preferible optar por .NET Remoting. Si se elige esta tecnología, será mejor utilizar el canal HTTP integrado con IIS y ASP.NET, ya que, de lo contrario, se deberá crear una infraestructura propia de seguridad y administración del ciclo vital del proceso. Dado que .NET Remoting requiere un cliente .NET, tiene sentido utilizar el formateador binario en lugar del formateador de SOAP; la interoperabilidad no supondrá un problema y el rendimiento se verá considerablemente mejorado.

5.1.2 CORBA

Según la ya conocida y prestigiosa enciclopedia de Internet: Wikipedia, CORBA (Common Object Request Broker Architecture), es un estándar que establece una plataforma de desarrollo de sistemas distribuidos, facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

CORBA ha logrado parte de su éxito debido a la clara separación entre la interfaz de los objetos y la implementación de los mismos. Las interfaces se definen utilizando el lenguaje IDL (Interface Definition Lenguaje, el lenguaje para definir interfaces CORBA), cuya principal característica es su alto nivel de abstracción, lo que le separa de cualquier entorno de desarrollo específico. Para la implementación de los objetos se puede utilizar cualquier lenguaje de programación que proporcione enlaces con este lenguaje. Para que un lenguaje de programación se pueda utilizar desde CORBA, debe tener definida la forma de enlazarse con IDL.

De esta forma, y a partir de una especificación de las interfaces en IDL, se generan unos lazos (*proxies*) en el lenguaje elegido que permiten el acceso a la implementación de los objetos desde la arquitectura CORBA.

La implementación de CORBA esconde al usuario de la arquitectura toda la complejidad que pueda existir en un entorno concreto (como por ejemplo su sistema AIX de IBM), y permite al programador desarrollar siguiendo la metodología genérica aprendida.

CORBA es un estándar creado con la idea de una distribución de los sistemas basada en objetos. Con CORBA se pretende definir una arquitectura que especifique cómo se crean los objetos y como se accede a sus funcionalidades. El mundo de los objetos se recrea en su máxima expresión y mostrando toda la potencia de esta metodología de desarrollo, hasta hace pocos años demasiado costosa para los equipos disponibles.

CORBA, al igual que los WS, permite comunicar diferentes lenguajes, con la diferencia de que éste utiliza el protocolo estándar IIOP (Internet Inter-ORB Protocol) definido por OMG. Y los datos son transmitidos en formato binario.

La principal diferencia entre esta tecnología y los WS, es que CORBA proporciona una arquitectura orientada a objetos, mientras que los WS realmente están basados en mensajes.

CORBA permite la comunicación entre muchos lenguajes, siempre y cuando estos implementen el protocolo IIOP y permitan crear un IDL. De hecho esta ha sido una de las principales desventajas de esta tecnología, dado que el protocolo IIOP no necesariamente está soportado por una red y frecuentemente se deben realizar operaciones especiales sobre un cortafuegos para que permita utilizarlo. WS utiliza el omnipotente HTTP.

Otra de sus desventajas es que dado que los datos viajan en formato binario, es necesario que los lenguajes hagan implementaciones especiales para leer estos tipos de datos. Aunque también, el hecho de que tengan este formato permite modelar estructuras complejas y ser rápido. WS utiliza XML comprensible por cualquier editor de texto.

En cuanto a la seguridad, se puede decir que en CORBA existen los mismos mecanismos para asegurarla que en los WS, sólo que acá no se encuentra estandarizada. De hecho existe una alianza entre SUN y Microsoft para establecer esta estandarización en los WS, pero se puede ver que, en su mayor parte, se trata de una buena definición de lo ya implementado en CORBA.[URL11]

5.2 JAVA VS .NET EN LA IMPLEMENTACIÓN DE WS.

Si bien, este trabajo de tesis esta basado en su mayoría en la implementación de los WS mediante la tecnología .NET, siempre es bueno tener parámetros de comparación para poder escoger una tecnología u otra para realizar la implementación de estos.

Los servicios Web actualmente deben manejar 4 desafíos esenciales, estos son:

1.- La descripción del Servicio.

2.- La implementación del Servicio.

3.- Los servicios de publicación, descubrimiento y enlace.

4.- La invocación y ejecución

Ahora se realizará una comparación revisando cada uno de estos 4 puntos, para llevar a cabo esta comparación se revisará, teóricamente, su implementación en las 2 plataformas de programación mas comunes que se utilizan hoy en día, estas son JAVA y .NET.

5.2.1 DESCRIPCIÓN DEL SERVICIO

El hecho de que los WS hayan proliferado tanto en tan poco tiempo, hace necesario que sea posible describirlo de una forma estructurada.

El WSDL ataca este problema definiendo una gramática en XML para describir los WS como una colección de puntos de entradas y puertos. En WSDL la definición abstracta de los puntos de entrada y mensajes esta separada de su implementación y una colección de puntos de entrada define un servicio.

J2EE

Los WS J2EE intercambian información con las partes interesadas usando WSDL para llegar a un acuerdo en el formato apropiado para cada documento XML transferido. Los

interesados que deseen realizar transacciones con una empresa que posea WS J2EE pueden buscar la información sobre los WS de la empresa en un registro.

.NET

Así como con los WS J2EE un WS .NET soporta WSDL y los utiliza para auto describirse, en este caso, sin embargo un espacio de nombres XML es usado dentro del documento XML para identificar en forma única los puntos de entrada del WS.

.NET provee componentes del lado del cliente que le permite invocar las operaciones descritas en un WS a través del documento WSDL y un componente del lado del servidor que mapea las operaciones del WS a un método de un objeto COM como esta descrito en el WSDL y en WSML (Web Service Meta Language), este archivo es necesario para la implementación de SOAP.

5.2.2 IMPLEMENTACION DE SERVICIOS

Implementar un WS actualmente significa estructurar los datos y las operaciones dentro de un documento XML que cumpla con la especificación SOAP, una vez que el componente WS es implementado, un cliente envía un mensaje al componente por medio de XML y la componente le responde.

J2EE

Las clases y aplicaciones de java existentes pueden ser encapsuladas usando la API de java XML basada en RPC (JAXR) y expuesta como un WS. JAXR utiliza XML para hacer llamadas a procedimientos remotos y expone una API para empaquetar

(empaquetar valores y retornar valores que serán distribuidos) y desempaquetar argumentos y para transmitir y recibir llamadas remotas.

Con J2EE los servicios de negocios escritos en Enterprise JavaBeans, son encapsulados y des-encapsulados como WS. El servicio encapsulado resultante es un WS basado en SOAP que cumple con la interfaz WSDL basada en los métodos del J2EE original. La arquitectura J2EE del WS es un conjunto de frameworks que proveen la infraestructura que permite a las compañías integrar servicios de lógica de negocios que eran expuestas previamente como interfaces propietarias. Actualmente J2EE soporta los WS a través del API de java para parseo (interpretación) de XML Parsing (JAXP), esta API permite a los desarrolladores realizar una operación de WS parseando manualmente los documentos

.NET

Las aplicaciones .NET no son ejecutadas directamente en código nativo de máquina, todos los programas son compilados a un código intermedio binario llamado MSIL, este código portable binario es después compilado a código nativo usando el just in time compiler (JIT) en tiempo de ejecución y es ejecutado en una maquina virtual llamada CLR (common language runtime), esto es similar a la forma en que java funciona, excepto que .NET soporta múltiples lenguajes; cada uno traducido a MSIL, lo cual es ejecutado en CLR usando el JIT. En la plataforma .NET Microsoft proveerá varios lenguajes basándose en el CLI (common language infraestucture) como c++, vb.NET y c#.

5.2.3 PUBLICACIÓN DESCUBRIMIENTO Y ENLACE DE SERVICIO

Una vez que un WS ha sido implementado debe ser publicado en algún lugar que permita a las partes interesadas encontrarlo. La información acerca de cómo un cliente va a conectarse a un WS e interactuar con este debe ser también expuesta en algún lugar accesible para ellos. Esta conexión y la información de interacción se conoce como la información de enlace.

Los registros son actualmente la forma principal de descubrimiento, publicación y enlace de los WS. Los registros contienen la estructura de datos y taxonomías para describir el WS y el proveedor del WS. Un registro puede ser alojado por organizaciones privadas o terceras partes neutrales.

IBM y Microsoft han anunciado recientemente la especificación WSIL (WS inspection language) que permite navegar los servidores Web buscando los servios Web XML. WSIL promete complementar el UDDI, logrando hacer más fácil el descubrimiento de WS Web no listados en los registros UDDI.

J2EE

Sun esta posicionando su API de java para registros XML (JAXR) con una única API de propósito general para interactuar con los múltiples tipos de registros. JAXR permite a sus clientes acceder los WS provistos por el implementador exponiendo los WS construidos en base a una implementación de la especificación JAXR.

Existen 3 tipos de proveedores JAXR

- **El proveedor general para JAXR** el cual implementa características independientes de cualquier tipo de registro.

- **El proveedor JAXR específico** para los registros que implementa la especificación JAXR de forma especifica.

- **Proveedor puente para JAXR**, el cual no es específico para ningún registros en particular, este sirve como puente entre ebXML o UDDI.

.NET

En principios Microsoft tenía el descubrimiento de WS con DISCO (de DISCOvery) en forma de un archivo DISCO. Un archivo DISCO publicado es un documento XML que contiene los links a otros servicios que describen los WS. Debido a la gran opción de los UDDI, aunque Microsoft lo ha utilizado para maximizar la interoperabilidad entre soluciones en los que es un conjunto de operaciones para interoperabilidad.

Adicionalmente, además de proveer un servidor UDDI.NET el SDK UDDI provee soporte para Visual Studio y depende del framework, productos como Microsoft Word ofrece servicios para descubrimiento de UDDI.

5.2.4 INVOCACION Y EJECUCION DE SERVICIOS.

EL SOAP es un protocolo basado en XML que define un framework de mensajería para intercambiar estructura de datos e información de tipos a través del Web, la especificación SOAP consta de 4 partes principales:

- Sobre

- Reglas opcionales de codificación de los datos para representar tipos de datos y un modelos de serialización para modelos de datos no sintácticos

- Un patrón de intercambio de mensajes

- Un enlace opcional entre SOAP y HTTP.

SOAP puede ser usado en combinación con cualquier protocolo de transporte o mecanismo que lo permita transportar.

Los receptores de un WS, validan un mensaje SOAP con su correspondiente esquema XML definido en un archivo WSDL. El receptor puede desempaquetar el mensaje SOAP.

Generando un mensaje SOAP en base al esquema XML correspondiente y definida en el mensaje XML, el escuchador del SOAP, desempaqueta el mensaje, y dentro de este, el escuchador puede invocar el código del WS.

Finalmente la lógica de negocios es invocada para obtener la respuesta. El resultado de la lógica de negocios es transformado en un mensaje SOAP y retornado al WS.

J2EE

Usa la API de JAVA para manejo de RPC basado en XML JAXR para enviar llamadas a métodos SOAP a servidores remotos y recibir la respuesta. JAXR permite que los

desarrolladores construyan WS incorporando los WS basados en XML, una vez que el servicio ha sido definido e implementado el servicio es deployado en un sistema de ejecución JAXR del lado del servidor.

Durante la implantación de un servicio JAXR, la herramienta de implantación configura uno o más enlaces a protocolos para los servicios. El enlace es una definición de servicios abstractos a un protocolo y un transporte basado en un XML específico. Un ejemplo es SOAP sobre HTTP.

Un cliente WS utiliza un servicio JAXR al invocar métodos remotos en un puerto descrito por un documento WSDL.

.NET

En el framework .NET de Microsoft las partes interesadas obtienen acceso al WS al implementar un escuchador del WS. Para implementar esto el sistema necesita entender el mensaje SOAP, generar una respuesta, proveer un contrato WSDL para el WS y publicar el servicio a través de UDDI.

Los desarrolladores .NET que crean escuchadores y consumidores de un WS basado en SOAP, actualmente tiene 3 alternativas:

- Usar las clases incluidas en .NET para mensajes SOAP.
- Construir un escuchador WS a mano usando MSXML, ASP o ISAPI, etc.

- Usar el Microsoft SOAP TOOLKIT para construir un escuchador de WS que conecte una fachada de negocios implementada usando COM.

Revisando cada uno de estos tópicos nos podemos dar cuenta de sus ventajas y desventajas presentes.

La diferencia mas importante radica en que .NET fue diseñado para este propósito y J2EE ah sido mejorado para esto.

Antes de tomar cualquier decisión se debe pensar que la mejor alternativa de implementación depende de que plataforma se este utilizando, ya que el servicio como tal no se influencia según la plataforma.

CAPITULO 6

6 CONCLUSIONES

Las tecnologías de desarrollo Web han evolucionado de manera exponencial y con la llegada de XML el cambio de lo que conocemos en este momento como WWW es y va a ser bastante importante, ya que vamos a obtener una red con un valor agregado muy superior al actual y la importancia de Internet aumentará a tal punto que será un servicio básico para todos los miembros de la sociedad. Los servicios Web ya están vislumbrando el cambio en la red, permitiendo que las organizaciones se comuniquen unas con otras aprovechando la infraestructura de comunicaciones de Internet y las posibilidades de XML.

En el desarrollo del presente trabajo de tesis se han podido comprender como funcionan, a nivel de seguridad, estos servicios; se han dado indicios para la utilización de las mejores prácticas al momento de implementarlos y también se ha mostrado la forma en que realmente se implementan.

También se ha realizado la comparación de esta tecnología y otras que sirven y/o tiene funcionalidades similares. La teoría de la implementación en las plataformas mas comúnmente utilizadas, igual se puede encontrar en el desarrollo de este trabajo.

Al analizar este trabajo, nos podemos percatar de las ventajas de implementar servicios utilizando esta tecnología y que día a día el avance que va teniendo ayuda a nuestra empresa u organización a crecer en el ámbito de las funcionalidades y/o servicios,

haciendo nuestro sistema mas efectivo y logrando un valor agregado para que se pueda competir con éxito en el mercado.

6.1 MEJORAS

Para la realización del presente trabajo de tesis se utilizó material actualizado para el desarrollo seguro de los Servicios Web. Pero aún quedan herramientas y teoría para investigar y/o desarrollar. Algunas de ellas se mencionan a continuación:

- Windows Communication Foundation (antes conocida como INDIGO), es la nueva herramienta tecnológica estandarizada que Microsoft esta actualmente esta sacando al mercado para la creación y puesta en marcha de sistemas interconectados. Los WS son su principal foco de utilización e implementación. Sería interesante realizar un estudio acabado de las funcionalidades y servicios que trae esta herramienta y una implementación con ella.[URL16]

- La utilización de WS cada día arremete más en el mercado y cada día ofrece más estándares de comunicación, seguridad, etc. Sería importante analizar en profundidad los cambios sucedidos al pasar de los años. Esto conlleva una investigación exhaustiva en el caso de que se sigan utilizando y el caso en que ya se hayan dejado de utilizar, el por qué de esto.

- También, y para concluir, es necesario hacer notar que los WS forman parte de un tipo de arquitectura orientada a servicio (SOA), es por este motivo que también sería conveniente analizar desde el punto de vista de desarrollo (incluyendo la seguridad por supuesto) e implantación un sistema completo de este tipo.

6.2 BIBLIOGRAFIA

6.2.1 DOCUMENTOS

- [BOR04] Borland Software Corporation(2004), **JBuilderX, Guía del Desarrollador de Web Services,** Borland Software Corporation

- [CAB05] Cabrera, Luis Felipe; Kart, Chris (2005). **Web Services Architecture and Its Specifications: Essentials for Understanding WS-***. Estados Unidos. Microsoft Press.

- [CHR05] Christensen E., Curbera F., Meredith G., Weerawarana S.; Leymann, Frank; Storey, Tony; Donald F. (2005), **Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More.** Estados Unidos. Prentice Hall

- [GAI04] Gailey, Jeannine H. (2004), **Understanding Web Services Specification and the WSE.** Estados Unidos. Microsoft Press.

- [MIC05] Microsoft (2005), **Web Service Security** Estados Unidos. Microsoft Press.

- [REY04] Angela Reyes Troncoso, **Implementación de un sistema de autentificación de datos basado en Web Services,** Tesis de Grado (2004).

6.2.2 SITIOS ONLINE

- **WS-Security**

[URL1]http://www.microsoft.com/spanish/msdn/articulos/archivo/121202/voices/understw.asp#understw_topic4

[URL2] http://www.info-ab.uclm.es/descargas/thecnicalreports/DIAB-05-01-2/Seguridad_en_Servicios_Web.pdf

[URL3] http://personal.telefonica.terra.es/web/lencorredera/seguridad_xml.pdf

[URL4]http://www.microsoft.com/technet/prodtechnol/windowsserver2003/es/library/ServerHelp/d55683e8-1258-4555-93cb-77138d33beab.mspx?mfr=true

[URL5]http://64.233.187.104/search?q=cache:-5KfYNBK5AMJ:www.sun.com/software/whitepapers/webservices/securing_webservices.pdf+data+confidentiality+web+services&hl=es&gl=cl&ct=clnk&cd=10

[URL6] http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0208.pdf

[URL7] http://www.rediris.es/cert/doc/unixsec/node27.html

[URL8] http://www.criptored.upm.es/guiateoria/gt_m159a.htm

Comparación de Remoting y WS

[URL9] Performance Comparison: .NET Remoting vs. ASP.NET Web Services

Seguridad en NET Remoting

[URL10]http://www.microsoft.com/spanish/msdn/arquitectura/BuildSecNetApps/html/SecurityGuide_Chapter11.asp

Servicios Web vs. CORBA

[URL11] http://www2002.org/CDROM/alternate/395/

Historia de XML

[URL12] http://www.desarrolloweb.com/descargas/descargar.php?descarga=6643

Introducción a XML

[URL13] http://www.cintel.org.co/media/temacentral_1_14.pdf

Ejemplo de aplicación B2B

[URL14] http://www-128.ibm.com/developerworks/webservices/library/ws-security.html)

Entidad Certificadora para X.509

[URL15] http://www.certisur.com/index.html?r

INDIGO

[URL16] http://www.microsoft.com/spain/servidores/interop/indigo.mspx