Mecanismos de Control de Acceso en Web Services

William Vásquez Romero

Juan Guillermo Rojas

Proyecto de grado presentado para optar el título de Ingeniero de Sistemas

Maria Isabel Serrano
Director

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA DE SISTEMAS

SANTAFÉ DE BOGOTA D.C.

DICIEMBRE 2004

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA DE SISTEMAS

Rector Magnífico: Padre Gerardo Remolina Vargas S.J.

Decano Académico Facultad de Ingeniería: Ingeniero Roberto Enrique Montoya Villa

Decano del Medio Universitario Facultad de Ingeniería: Padre José Sarmiento Nova S.J.

Director Carrera de Ingeniería de Sistemas: Ingeniera Hilda Cristina Chaparro López

Director Departamento de Ingeniería de Sistemas: Ingeniero Germán Alberto Chavarro Flórez

TABLA DE CONTENIDO

1. RESUMEN 6
2. OBJETIVOS 8
   2.1. OBJETIVO GENERAL 8
   2.2. OBJETIVOS ESPECIFICOS 8
3. INTRODUCCION 9
4. MARCO TEORICO 11
   4.1. ANTECEDENTES 11
      4.1.1. ¿QUÉ ES UN WEB SERVICE? 11
      4.1.2. ¿CUÁLES SON Y COMO FUNCIONAN LOS REQUERIMIENTOS DE SEGURIDAD? 12
      4.1.3. ESQUEMA Y PROBLEMAS DE SOAP 14
   4.2. ANALISIS DE LOS FALLOS DE SEGURIDAD EN LOS WEB SERVICES 15
      4.2.1. ATAQUES MÁS COMUNES A UN WEB SERVICE 15
      4.2.2. PUNTOS DÉBILES DE LOS WEB SERVICES 21
5. METODOLOGIA DE LA INVESTIGACION 23
6. TECNOLOGIAS EXISTENTES PARA LA SEGURIDAD EN WEB SERVICES 25
   6.1. MODELO DE UNA ARQUITECTURA SEGURA 25
      6.1.1. MICROSOFT .NET FRAMEWORK EN LA SEGURIDAD DE LOS WEB SERVICES 25
      6.1.2. USO DE CERTIFICADOS DIGITALES EN UNA INFRAESTUCTURA B2B USANDO MICROSOFT CERTIFICATE SERVICES 28
      6.1.3. COMO FUNCIONA LA AUTENTICACIÓN Y LAS FIRMAS DIGITALES CON WSE (*WEB SERVICES ENHANCEMENTS*) 37

6.1.4. ENCRIPTACIÓN DE MENSAJES SOAP USANDO WSE (WEB SERVICES ENHANCEMENTS) 41
6.2. ANÁLISIS DE DEBILIDADES Y FORTALEZAS DE LAS TECNOLOGÍAS EXISTENTES 53
7. IMPLEMENTACIÓN DE UN ESQUEMA DE SEGURIDAD CON TECNOLOGÍAS EXISTENTES 54
7.1. MÉTODOS GENERALES DE ASEGURAMIENTO DE UN WEB SERVICE 54
7.2. PLAN DE PRUEBAS PARA EL WEB SERVICE SEGURO (USUARIO – CONTRASEÑA) 55
7.3. PRUEBAS REALIZADAS AL WEB SERVICE SEGURO 57
7.3.1. INTRODUCCIÓN 57
7.3.2. ESCENARIO DE PRUEBAS 58
7.3.3. DOCUMENTACIÓN DEL SOFTWARE DE PRUEBAS DEL WEB SERVICE SEGURO 59
7.3.4. ESTADÍSTICAS DE LAS PRUEBAS REALIZADAS AL WEB SERVICE SEGURO 60
8. PROTOTIPO FUNCIONAL DE UN WEB SERVICE SEGURO USANDO SEGURIDAD BASADA EN ROLES 62
8.1. INTRODUCCION 64
8.2. ARQUITECTURA DE LA SOLUCION 66
8.3. IMPLEMENTACIÓN DE UN WEB SERVICE USANDO AUTENTICACION POR ROLES 66
8.3.1. REQUERIMIENTOS DE SOFTWARE 66
8.3.2. REQUERIMIENTOS DE HARDWARE 67
8.3.3. LIBRERÍA DE SEGURIDAD 67
8.3.4. WEB SERVICE 68
8.3.5. CLIENTE 70
8.4. FUNCIONAMIENTO GENERAL 70
8.5. PLAN DE PRUEBAS PARA EL WEB SERVICE BASADO EN ROLES 71

8.6. PRUEBAS REALIZADAS AL WEB SERVICE DE ROLES 73
8.6.1. DOCUMENTACIÓN DEL SOFTWARE DE PRUEBAS DEL WEB SERVICE POR ROLES 73
8.6.2. FUNCIONAMIENTO DEL PROGRAMA 74
8.6.3. ESTADÍSTICAS DE LAS PRUEBAS 74
8.7. ANÁLISIS DE DEBILIDADES Y FORTALEZAS DEL MECANISMO DE SEGURIDAD BASADO EN ROLES 76
9. TECNOLOGIAS DE SEGURIDAD EXISTENTES VS SEGURIDAD BASADA EN ROLES 77
9.1. COMPARACION BASADOS EN SUS CARACTERISTICAS 77
10. CONCLUSIONES 79
10.1. IMPLEMENTACIÓN 79
10.2. IMPACTO EN LOS DESARROLLOS EXISTENTES 80
10.3. COSTO DE IMPLEMENTACIÓN 80
10.4. IMPACTO EN EL FUTURO 80
11. PROYECTOS A FUTURO 82
11.1. DESARROLLO DE UN ADD-IN PARA IMPLEMENTAR LA SEGURIDAD EN UN PROYECTO DE WEB SERVICES EN VISUAL STUDIO .NET 82
11.2. DESARROLLO DE LA LIBRERÍA DE SEGURIDAD PARA LA AUTENTICACION BASADA EN ROLES EN UN AMBIENTE J2EE 83
12. BIBLIOGRAFÍA 84
13. ANEXOS 86

1. RESUMEN

Este documento presenta los resultados del proyecto de grado titulado: “Mecanismos de Control de Acceso en Web Services”

Desde el surgimiento de los Web Services, surgió junto a él el problema de la seguridad, esto debido que éstos no fueron concebidos para ser privados, sino públicos, y para estar protegidos hasta cierto punto por la arquitectura interna de la organización que los creara y los publicara.

El presente documento narra los problemas principales de seguridad que encuentran los Web Services desarrollados hoy en día, y plantea dos soluciones; una de ellas enfocada a solucionar el problema del control de acceso por medio de los mecanismos existentes basados en los estándares de la W3C[1] y su implementación por parte de la empresa Microsoft; y otra enfocada a crear un mecanismo de control de acceso basada en roles, desarrollada desde su concepción hasta su implementación por los autores, buscando minimizar el impacto en los desarrollos existentes y dando una opción para el escenario particular en donde una autenticación de usuario y contraseña no es suficiente para controlar el acceso a un Web Service, todo esto sin perder la interoperabilidad y la publicidad ya que son estos dos los pilares de la concepción de estos servicios.

Finalmente estos dos desarrollos son contrastados para ver sus debilidades y fortalezas, y poder ver en que casos, cual de ellos es mejor y por cuál se debe optar para lograr un mejor aseguramiento de los Web Services.

El presente trabajo no pretende dar una única solución al problema del control de acceso, pero si presentar un avance considerable en el acceso por roles a los Web Services, presentando no solo un prototipo funcional, sino varias librerías de seguridad que permitirán implementar la seguridad basada en roles en cualquier desarrollo previamente creado usando el .Net Framework, y que afecta de manera mínima el desarrollo previamente realizado y respeta la lógica de la organización.

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Diseñar e implementar un mecanismo de control de acceso para Web Services.

2.2. OBJETIVOS ESPECIFICOS

- Extraer las características que debe tener un mecanismo adecuado de control de acceso a servicios Web.

- Analizar las tecnologías existentes, para entender su funcionamiento, a nivel de Hardware y de Software para entender el alcance de estas soluciones y los problemas que abordan.

- Construir un Web Service seguro, que sirva como plataforma de pruebas sobre la que se validará la solución propuesta.

- Implementar una solución al problema del control de acceso a los Web Services.

3. INTRODUCCION

A través de las investigaciones se pueden encontrar a veces variaciones de las tecnologías existentes, otras veces descubrir tecnologías que con el tiempo pueden cambiarnos incluso nuestro modo de vida. Ese es el caso de Internet, el cual nació como un intento de comunicación primario, pero con su evolución tan acelerada, se convirtió en el medio de comunicación, incluso de negociación, de hoy en día.

Pero todo cambio trae consecuencias buenas y malas; en el caso de Internet uno de los problemas mas estudiados ha sido la seguridad que podamos ofrecerle a sus usuarios para que pueda llegar a ser un medio confiable. Existen muchas investigaciones encaminadas a minimizar este problema y encaminadas a encontrar cada día medios a través de los cuales el intercambio de información sea de una manera rápida, eficaz y segura. Uno de estos medios se llama los servicios Web, Sun Microsystems dice: *"Los Web Services son actualmente la mayor innovación en el sector de la Informática. ¿Que entendemos con este término? Debido a la diversidad de interpretaciones en la industria, la forma más genérica de definir los Web Services es XML en acción. Si la red es el ordenador, Los Web Services significan el software que se ejecuta en la red." – Simon Phipps, Chief Technology Evangelist en Sun Microsystems*. [2]

En un principio esta idea abrió muchas posibilidades de expansión y confiabilidad en las comunicaciones, pero nuevamente nació la misma preocupación: la seguridad de este servicio, y es aquí en donde este trabajo de investigación busca encontrar una variación de los métodos existentes de seguridad aplicándolos a los *Web Services*, algunos métodos de seguridad

hablan de autenticación, autorización, integridad de datos, confidencialidad, no repudiación, etc. Y para poder llegar a dar un juicio correcto acerca del comportamiento de las aplicaciones actuales frente a la forma de ofrecer un servicio de seguridad que pueda unir estos métodos de la mejor manera y luego poder llegar a dar una solución propia, se debe haber hecho una investigación completa; así pues, dentro de este documento se analizará las diferentes opciones de seguridad y luego se dará una solución propia aplicando todos estos conocimientos adquiridos. Luego de analizar estos requerimientos se podrán ver los métodos de comunicación existentes, y cuales son sus ventajas y desventajas, analizando sus fallos en el área de seguridad a través del estudio de los ataques más comunes existentes para los *Web Services*, con el fin de dar una solución final que también cubra la comunicación entre las diferentes entidades[1] y posea la suficiente seguridad para brindar un mejor servicio.

Las investigaciones encontradas y estudiadas acerca de la seguridad de los *Web Services* han generado ciertos parámetros que se deben tener en cuenta al momento de implementar estos servicios para poder obtener un ámbito seguro de comunicación. Y es aquí donde en este trabajo de investigación se ofrecen posibilidades de expansión y de implementación de nuevas opciones de seguridad, refiriéndose a la autenticación por medio de roles para Web Services; una aplicación que al momento de integrarla con los servicios existentes, ofrecerá una nueva visión de seguridad y le dará al Web Service una nueva dimensión en la cual explorar nuevas ideas e innovaciones.

[1] Con Entidad se entiende todo sistema que interactúe con otro, ya que los Web Services están orientados a este tipo de comunicación, aunque también se podría dar la comunicación con un usuario y este también seria parte de la definición de entidad.

4. MARCO TEORICO

4.1. ANTECEDENTES

Antes de entrar a analizar como llegar a la implementación de la propuesta de este trabajo de investigación, y a la explicación de cómo esto puede darle un valor agregado a los *Web Services* existentes, debemos entender el funcionamiento global y los componentes generales de estos servicios.

4.1.1. ¿QUÉ ES UN WEB SERVICE?

Los Web Services nacieron como una respuesta a la necesidad de comunicar las aplicaciones que existen en Internet haciendo uso de los protocolos de comunicación existentes, que permitan su utilización a través del uso de sistemas automatizados, los cuales realicen estas tareas de intercambio de datos de una forma automática [3].

La evolución de Internet ha llevado a los proveedores de tecnologías a buscar nuevas formas de garantizar a las organizaciones un mejor aprovechamiento de este recurso, de allí que hoy podamos pensar en tener un E – Business puro, donde los Web Services sean la espina dorsal. Para garantizar una solución robusta, se requiere de mucho más que simple interacción programática (sin intervención de la mano humana). Aspectos como la seguridad y el manejo de transacciones distribuidas son débiles hoy en día y requieren de más estudio.

Para invocar métodos en objetos remotos se creó SOAP[2] el cual, basado en XML, construye un mensaje en el que se encuentran embebidos los parámetros y las peticiones de los servicios requeridos, de esta manera se hace transparente para el cliente la petición de los recursos, delegando al servidor el problema de transformar estas peticiones de acuerdo con su lógica de programación y a su arquitectura. Una vez estandarizado el modelo de comunicación entre el cliente y el servidor se necesita de un depósito que contenga la información correspondiente a los métodos, sus argumentos y su localización para su posterior uso, así nace UDDI[3], que de forma similar a un DNS[4], resuelve la ubicación de un servicio, entregando al cliente la forma de acceder a los servicios.

Ahora se debe entrar a analizar los requerimientos de seguridad y la forma de comunicación de los Web Services.

4.1.2. ¿CUÁLES SON Y COMO FUNCIONAN LOS REQUERIMIENTOS DE SEGURIDAD?

Para poder ofrecerle al cliente seguridad en la comunicación con los proveedores de servicios se utiliza una combinación de las siguientes características:

Autenticación. El requerimiento fundamental en las aplicaciones de Internet es poder identificar al usuario así entonces, la autenticación es un proceso por el cual se puede identificar a una persona o a un sistema y validar sus credenciales, normalmente se hace mediante el empleo de una combinación

[2] SOAP: Simple Object Access Protocol.
[3] UDDI: Universal Description, Discovery and Integration.
[4] DNS: Domain Name System.

de nombre de usuario y contraseña. Si la contraseña no es protegida adecuadamente, entonces la autenticación se verá comprometida.

Otra manera de garantizar la autenticación, es por medio del uso de certificados, el cual se basa en llaves privadas y publicas para encriptar la información y garantizar la autenticidad de los servicios, para lograr esto se necesita de una Entidad Certificadora externa al Servicio con el fin de poder garantizar la identidad de los clientes.

Autorización. Una vez que se ha autenticado al usuario, se le debe dar la autorización de acceso, esto quiere decir que identifica a qué recursos tiene derecho el usuario, y cuáles recursos le serán negados al momento de una petición. Esta autorización se hace a través del uso de ACL (*Access Control List*), en esta lista se encuentran los usuarios y los tipos de acceso que tienen a los recursos.

Integridad de Datos. Consiste en ofrecerle al cliente la seguridad de que la información que se está enviando a través de la conexión con el proveedor, no ha sido alterada, esto quiere decir que no se hará ninguna modificación a la información transportada por quien no esté autorizado a hacerlo.

Confidencialidad. Es el proceso por el cual se asegura que la información no pueda ser leída, a menos que se tenga autorización para ello. No basta con emplear la identificación del usuario y la contraseña, sino que además la información es encriptada para protegerla.

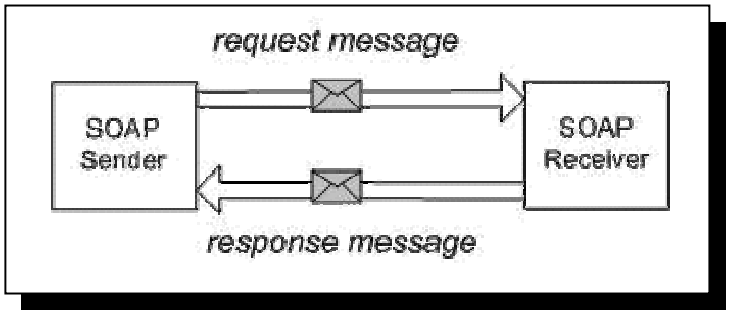
No Repudiación. Asegurar que no se pueda negar el haber participado en una operación, esto quiere decir que se debería mantener un historial de participación o de uso del sistema, de los usuarios.

Auditar. Es el proceso de grabar los eventos relacionados con la seguridad en la comunicación y tomar acciones basados en las ocurrencias es estos.

4.1.3. ESQUEMA Y PROBLEMAS DE SOAP

SOAP (*Simple Object Access Protocol*) "Es un protocolo elaborado para facilitar la llamada remota de funciones a través de Internet, permitiendo que dos programas se comuniquen de una manera muy similar técnicamente a la invocación de páginas Web.[5]"

Figura 1. Funcionamiento de los mensajes SOAP



Fuente: Autores

Este protocolo tiene varios problemas al momento de ofrecer una solución completa de seguridad:

- SOAP no puede diferenciar entre la petición por parte de un usuario anónimo o un socio de negocio conocido. [4]
- SOAP no puede realizar la autenticación, la autorización y el control de acceso del usuario. [4]

[5] SOAP y Web Services: http://www.microsoft.com/spanish/msdn/articulos/archivo/mtj/voices/art51.asp

Analizando estas desventajas se puede encontrar que este protocolo da carta abierta al análisis y la búsqueda de oportunidades para crear herramientas que basadas en el uso de este método, puedan dar un valor agregado mediante mecanismos que corrijan los inconvenientes de seguridad de los mensajes SOAP, y permitan ofrecer una comunicación más robusta y confiable, como por ejemplo métodos de control de acceso con el fin de identificar a los usuarios.

4.2. ANALISIS DE LOS FALLOS DE SEGURIDAD EN LOS WEB SERVICES

En este capitulo se enunciaran los ataques más comunes sobre Internet a un Web Service, y los puntos débiles que suelen tener los mismos, esto en aras de ver de forma macro los problemas existentes, para poder formular una arquitectura con las herramientas de seguridad que existen a nivel de estándares, y poder de esta manera ver que puntos no son cubiertos por estos, y que puntos siguen siendo vulnerables aún, después de todo un desarrollo y planeamiento de una arquitectura segura.

Después de visto esto, se podrá formular una arquitectura segura, y se hará un análisis parte por parte, con el fin de ver que aspectos de seguridad nos provee dicha arquitectura, y cuales no.

4.2.1. ATAQUES MÁS COMUNES A UN WEB SERVICE

Hoy en día cuando Internet esta más desarrollada que nunca, y cuando las empresas han empezado a globalizarse, es cuando los problemas asociados a la seguridad cobran más valor.

Ahora con el uso de los Web Services, las empresas tienen la forma de comunicarse con un grupo de personas a las cuales desean poner a disposición alguna información de su empresa. El problema radica en el momento en que alguien que no tiene los privilegios necesarios acceda a la información, pudiendo tomar ventaja de la misma para su propio beneficio.

Es por esto que antes de comenzar el desarrollo de los Web Services, las empresas deben saber a que se enfrentan y cuales son los mayores riesgos de seguridad en Internet.

A continuación se mencionan algunos de los ataques mas comunes a los Web Services, con el fin de entender un poco el funcionamiento de estos, y poder tener una base de la cual partir para llegar a plantear soluciones que además de dar un valor agregado, sigan manteniendo la constante de evitar estos tipos de ataques pero que también nos den la flexibilidad de poder soportar nuevas formas de invasión a la seguridad.

Spoofing: Es el ataque clásico a las páginas que requieren autenticación de usuarios. El Hacker lo que intenta hacer es averiguar las credenciales de la persona que desea suplantar, esto puede ser posible si conoce a la persona en cuestión y si la ha estado investigando (ingeniería social). Otra forma de lograr esto es por medio de un ataque basado en un diccionario, el cual intenta encontrar la palabra adecuada para poder acceder al sistema.

Otra técnica es la suplantación DNS, que consiste en colocar una máquina entre el cliente y el servidor, la cual filtrará los paquetes y los redireccionará a una máquina que contiene una copia exacta del sitio, solo que su código fue modificado para poder extraer las credenciales del cliente, para esto existen

diversas herramientas que nos ayudan a tener una copia espejo de un sitio Web, entre estas herramientas podemos encontrar una llamada Teleport pro.

Está técnica de Spoofing se puede repeler fácilmente, usando estrategias de contraseñas complejas, y mecanismos seguros de autenticación, entre estos mecanismos, uno de los mas comunes es el uso de Certificados Digitales.

Aprovechamiento de los Bugs: Los bugs son errores en el código que no se descubrieron dentro de los controles de calidad al software en el momento del desarrollo. Una persona que este interesada en explotar esto puede aprovecharse de errores de sintaxis, *querys* mal realizados, etc., para tratar de acceder al código fuente, o para tratar de extraer los archivos del sistema, o inclusive pasar la seguridad y acceder al servicio como si fuera el cliente.

Denial of Service: Este tipo de ataque no tiene como objetivo violentar la seguridad del sistema ni tampoco obtener información del mismo, su único fin es hacer que el servidor no provea un buen tiempo de respuesta, hasta el punto de no poder responder en absoluto, ya que se encuentra ocupado atendiendo muchas solicitudes simultáneas.

Esto se logra, enviando muchos paquetes de diferentes tipos, que fuerzan al servidor a dar una respuesta; el problema esta cuando la cantidad de requerimientos sube exponencialmente, el servidor también se ocupa exponencialmente, y por lo tanto llegará un momento en que no podrá atender más requerimientos.

En el caso específico de los Web Services, se pueden enviar una serie de requerimientos SOAP, a un método particularmente complicado, los cuales al incrementarse y enviarse simultáneamente, provocan que el servidor no pueda responder al requerimiento.

Ataques XML: Cuando nos referimos a los ataques XML, nos estamos refiriendo específicamente a aquellos mensajes SOAP que se envían a un Web Service para obtener algún tipo de información. Los principales ataques son los siguientes: [5]

- Documentos muy grandes en XML: Al enviar el paquete SOAP, se puede crear un documento XML muy grande para esta petición, provocando que el servidor se demore mucho, desembocando en un problema de DoS (*Denial of Service*), tratado anteriormente.

- Entidades que se refieren al sistema de archivos: Cuando alguna de las entidades tiene una referencia a uno de los archivos del sistema es vulnerable a que se haga una solicitud que provoque un error interno, desembocando en acceso al archivo, o en algunos casos a saltar la seguridad por completo, esto es debido a que muchas veces en el momento de mostrar el error en pantalla, podemos visualizar la ruta completa del archivo y así poder hacer un seguimiento dentro del servidor.

- Session Hijacking: Consiste en tomar el control de una sesión de usuario ya establecida con un servidor después de tomar control del ID de esta sesión. Session Hijacking involucra el uso de captura, fuerza bruta e ingeniería inversa a los IDs(identificaciones) de Sesión para tomar el control legitimo de una sesión Web mientras esta sesión esta aun en progreso.

Con el fin de hacer un seguimiento de estas sesiones en HTTP, los programadores deben desarrollar una forma de rastrear el estado de múltiples conexiones al mismo usuario, en vez de solicitar al usuario la autenticación por cada petición a la aplicación Web. Una sesión es una serie de interacciones entre dos puntos finales de comunicación que ocurre

durante el tiempo que dura una conexión simple. Cuando un usuario se logea en una aplicación Web una sesión se crea en el servidor con el fin de mantener el estado de las peticiones originadas por parte del usuario.

Las aplicaciones usan las sesiones para almacenar los parámetros que son relevantes para los usuarios. La sesión se mantiene "viva" en el servidor tanto tiempo como el usuario permanezca usando el sistema. Esta se destruye después que el usuario sale del sistema o también después de un tiempo predefinido de inactividad. Cuando la sesión es destruida también se destruyen los datos del usuario localizados en los espacios de memoria.

Un ID de sesión es un *string* (usualmente un tipo de dato *long*, *random*, o un dato *alpha – numeric*) que es transmitido entre el cliente y el servidor. Estos IDs de sesión se almacenan en *cookies*, URLs y campos ocultos de las páginas Web. Cuando una URL contiene el ID de sesión es algo como:

```
http://localhost/TestWSSession/(0wcics450yuwi555bshhuz45)
```

En una pagina HTML, un ID de sesión puede ser un campo oculto:

```
<input type="hidden" name="sessionID" value="54321abcd">
```

Algunas veces, las *cookies* son eliminadas después de cerrar el *browser*, estas reciben el nombre de "*session cookies*" o *cookies* no persistentes. Las *cookies* que van mas allá de la sesión del usuario son llamadas *cookies* persistentes, estas son almacenadas en el Disco Duro del usuario, su ubicación es determinada dependiendo del sistema operativo y el *browser* utilizado, por ejemplo en Windows 2000, para el Internet Explorer se pueden ubicar en:

```
C:\Documents and Settings\username\Cookies.
```

Existen diferentes problemas con la identificación de los IDs de sesión, muchos de los *Websites* más populares usan algoritmos basados en datos sencillos, como el tiempo o la dirección IP, con el fin de generar el ID de sesión ocasionando que estos IDs puedan llegar a ser predecibles. Además si no se usa un tipo de encripción este ID viajaría por Internet de una forma clara y haciéndola susceptible a poderse ver con el uso de *backdoors*.

Session Hijacking involucra un ataque usando captura, fuerza bruta o ingeniería inversa para los IDs de sesión, con el fin de tomar el control de una sesión legítima mientras la conexión esta aun activa. En la mayoría de aplicaciones, después de lograr robar la sesión, el atacante obtiene acceso total a todos los datos del usuario, y se le es permitido realizar operaciones que el usuario, cuya sesión fue robada, tenia permisos para ejecutar.

Tenemos tres técnicas principales para robar sesiones:

- **Fuerza Bruta.** El atacante intenta múltiples IDs de sesión hasta obtener éxito.
- **Calcular.** En muchos casos, las identificaciones se generan de una manera no al azar y pueden ser calculadas.
- **Robar.** Con el uso de diferentes tipos de técnicas, el atacante puede obtener los IDs de sesión.

En la primera, Fuerza bruta, el atacante puede probar diferentes IDs de sesión, por ejemplo:

```
http://www.somesite.com/view/VW30422101518909
http://www.somesite.com/view/VW30422101520803
http://www.somesite.com/view/VW30422101522507
```

Los IDs de sesión pueden ser robados haciendo uso de diferentes técnicas: "*sniffing network traffic*", usando *trojans* en los PC del cliente, usando encabezados HTTP en donde el ID de sesión es un parámetro en la dirección y usar ataques de *cross-site scripting"*.

En un ataque "referrer", el atacante intenta hacer que el usuario realice un clic a un link de otro sitio (Ej. www.hostile.com) y tener un código como este:

```
GET /index.html HTTP/1.0
Host: www.hostile.com
Referrer: www.mywebmail.com/viewmsg.asp?msgid=438933&SID=2343X32VA92
```

El browser envía la referencia URL conteniendo el ID de sesión al sitio del atacante www.hostile.com, y con esto el atacante obtiene el ID de la sesión del usuario.

El ID de sesión puede robarse también usando "*script injections*", el usuario ejecuta un script corrupto que redirecciona la información del usuario al atacante.

4.2.2. PUNTOS DÉBILES DE LOS WEB SERVICES

En estos momentos, cuando ya existen muchos desarrollos de Web Services tanto empresariales como individuales, se debe tener muy en cuenta que el desarrollo de los Web Services es todavía muy pequeño, ya que todavía faltan muchas preguntas por responder, y por lo tanto muchos puntos débiles que fortalecer. A continuación se enumeran los más importantes: [6]

Seguridad/privacidad. Al asegurar que todos los usuarios tienen el mínimo privilegio, es decir que solo tienen acceso a lo que deben tener acceso, se

previene el acceso de personas no autorizadas. Este problema no lo aborda SOAP puntualmente, ya que se han desarrollado herramientas adicionales para asegurar esto, como el uso de certificados digitales, pero que no son óptimas en todos los escenarios de desarrollo.

Enrutamiento/confiabilidad/transaccionalidad. Se deben desarrollar métodos que permitan monitorear el paso de mensajes y se pueda garantizar que en el caso de que una transacción falle, esta se pueda devolver (*rollback*). Hasta que esto pueda ser posible, la capacidad de un Web Service es limitada.

Manejo transaccional. Este es uno de los puntos con mayor importancia cuando se habla de un Web Service, ya que al no mantener un estado ni poder manejar sesiones[6], es imposible saber como manejar una transacción distribuida y como deshacerla en el caso de que un error ocurra. Muchas empresas desarrolladoras han abordado este tema, y han desarrollado herramientas, pero W3C[7], no ha demostrado iniciativa en desarrollar un estándar para el desarrollo de un manejador transaccional, lo que implica que cada desarrollo es diferente y por lo tanto específico a la solución.

Perfomance. Esto se refiere a que un Web Service todavía no puede tener la robustez que tiene un sistema altamente distribuido, en el cual varias máquinas pueden trabajar al mismo tiempo en un requerimiento.

[6] Las sesiones en una aplicación es un estado que se mantiene para cada usuario y en donde su vigencia solo es durante el tiempo que se haga uso de la aplicación. Dentro de una sesión se pueden mantener datos necesarios para las transacciones.
[7] The World Wide Web Consortium

5. METODOLOGIA DE LA INVESTIGACION

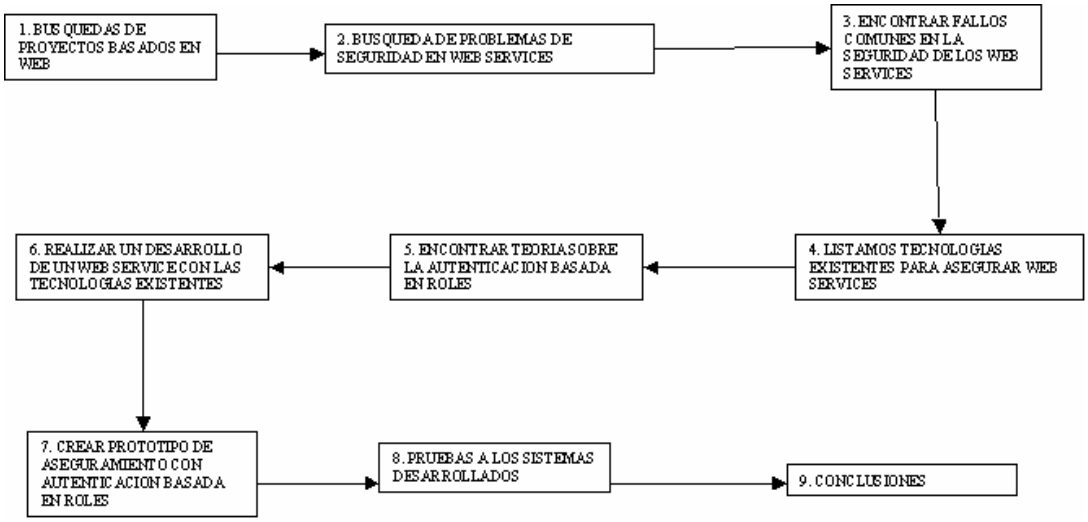
El proceso de investigación que se llevó acabo para el desarrollo de la presente tesis, es producto del interés de los autores de desarrollar un producto que cubriera alguna de las necesidades actuales de seguridad de los sistemas distribuidos existentes, específicamente los Web Services.

Este proceso comenzó indagando sobre que problemas de seguridad existían en estos momentos en los desarrollos Web, acortando el espectro y limitándolo solamente a los Web Services, esto por ser uno de los paradigmas más importantes del desarrollo Web.

Una vez encontrados los fallos particulares en la seguridad de los Web Services, se concluyó que la autenticación basada en roles era un tema no tratado, mostrando así un área de oportunidad digna de explorar y explotar. Una vez encontrado el tema principal de la investigación, se creó un desarrollo utilizando las tecnologías existentes, con el fin de tener una base contra la cual contrastar la solución planteada.

Finalmente se creo un desarrollo de seguridad basada en roles para Web Services desarrollados usando el .Net Framework, garantizando un mínimo impacto en las soluciones existentes y no perdiendo ninguna de las características principales de ellos.

Figura 2. Diagrama de flujo Proceso de la Investigación



6. TECNOLOGIAS EXISTENTES PARA LA SEGURIDAD EN WEB SERVICES

6.1. MODELO DE UNA ARQUITECTURA SEGURA

Antes de entrar a analizar el modelo de arquitectura a proponer, se debe tener en cuenta sobre que herramienta de desarrollo se basa el análisis, esto no con el ánimo de limitar la investigación, sino de darle una sustentación lógica al trabajo realizado.

Dentro de este trabajo de investigación, se hizo uso de Visual Studio .NET como herramienta de desarrollo, con el lenguaje C#, así como el uso de las tecnologías Microsoft para el manejo de firmas digitales, certificados digitales y uso del estándar de la W3C para el aseguramiento de Web Services.

6.1.1. MICROSOFT .NET FRAMEWORK EN LA SEGURIDAD DE LOS WEB SERVICES

Se tiene claro que el objetivo de este trabajo de investigación, es encontrar una forma menos costosa de implementar una solución de seguridad en los Web Services y que afecte lo menos posible a la arquitectura presente en la empresa, es por esto que la solución al problema debe ser dada a manera de estándar, lo cual implica crear una serie de reglas y pasos los cuales se deben seguir para poder implementar una seguridad que nos permita dar confiabilidad de servicio al igual que nos permita darle valor agregado a la aplicación dentro de la empresa.

También se debe considerar que el estándar de los Web Services implica que deben poder ser accedidos desde cualquier parte, es decir un Web Service desarrollado en .Net debe poder ser accesado desde un cliente java o un cliente .Net con un mínimo de impacto en el desarrollo; entonces la solución que se plantee al final de este proyecto, debe también conservar la interoperabilidad propuesta por la W3C, la cual es una entidad que se encarga de regular y generalizar la forma de comunicación de las aplicaciones, esto con el fin de mantener un estándar de comunicación.

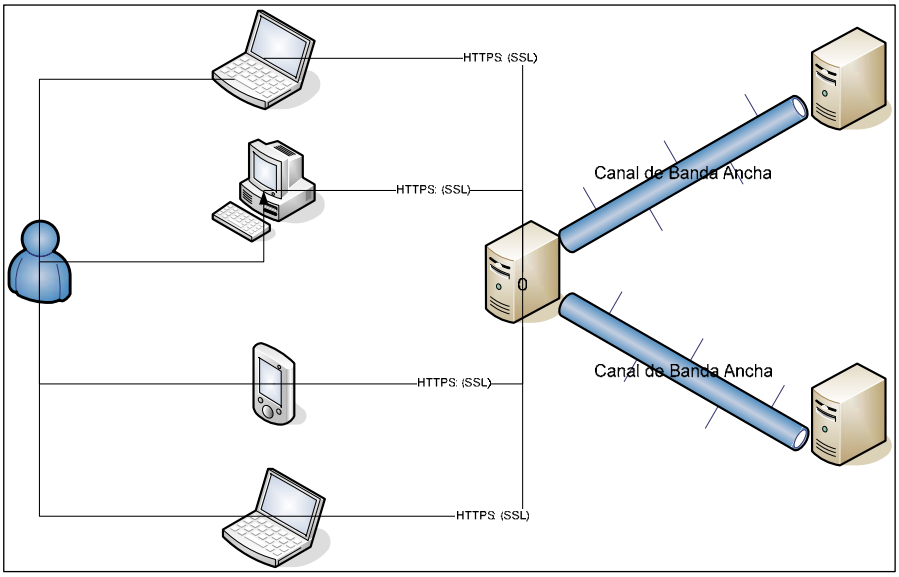
Todo lo anterior nos lleva a una sola conclusión, la herramienta que se use para hacer el desarrollo es indiferente. Al darnos cuenta de esto escoger .Net o Java para realizarlo es lo que menos nos debe preocupar, sin embargo, nosotros, el grupo investigador, escogimos la plataforma .Net por ser una plataforma muy robusta y en la cual podemos encontrar mucha ayuda ofrecida por las diferentes entidades orientadas a la investigación y a la colaboración a los desarrolladores de .NET, creando así un área de oportunidad buena para nosotros y para las personas que luego deseen seguir este desarrollo, para lograr más y mejores cosas, basados en las investigaciones y los conceptos que en el presente trabajo se recopilen.

Al haber visto los problemas de seguridad que existen podemos entrar a explicar el funcionamiento de la solución a la que se desea llegar a nivel de seguridad en los Web Services. Una vez que tenemos los principales fallos a nivel de seguridad en un Web Service, debemos entonces tratar de maximizar los niveles de seguridad basados en los estándares existentes, en las herramientas desarrolladas, y en la creatividad del arquitecto que lo este desarrollando.

Es por esto que teniendo como base la arquitectura de Windows, y el Framework de .Net, se propone la siguiente arquitectura que incorpora

nuevos conceptos a nivel de seguridad, los cuales serán explicados más adelante.

Figura 3. Arquitectura B2B, de Web Services



Fuente: Autores

Como podemos ver en la gráfica anterior, hemos ilustrado toda una arquitectura, partiendo de clientes seguros, es decir clientes que se comunicarán por medio de canales encriptados y sistemas de seguridad ya desarrollados, y a los cuales se les agregará funcionalidad para ofrecer servicios interoperables, y no restrictivos a un usuario y una contraseña. Cabe anotar que los clientes pueden ser otros servidores y/o clientes normales que se comuniquen por medio de navegadores o aplicaciones de escritorio.

Adicionalmente se hace una distinción entre los clientes y los servidores, porque en los servidores es donde se va a alojar la solución y se da por sentado que los servidores se conocen entre si y que existen canales y relaciones de confianza que permiten unos niveles de seguridad excelentes entre ellos.

Teniendo como base este diagrama, y la idea general de que deseamos establecer comunicaciones seguras, vamos a ver a continuación uno de los métodos existentes para lograr esto, el cual ya se había nombrado anteriormente como un ejemplo; nos referimos al uso de los certificados digitales y como es su funcionamiento. Este método fue implementado para ver su comportamiento como se especifica en el capitulo siguiente.

6.1.2. USO DE CERTIFICADOS DIGITALES EN UNA INFRAESTUCTURA B2B USANDO MICROSOFT CERTIFICATE SERVICES

Basados en el diagrama de arquitectura de la sección anterior, ahora se dará una propuesta de solución a algunos de los problemas de seguridad que se plantean en cualquier ambiente de negocios, utilizando mecanismos ofrecidos por la plataforma, en este caso los servidores Microsoft. Esto no se hace con el fin de encasillar la investigación, sino con el ánimo de dar un ejemplo de cómo se hace una implementación de este tipo, y darnos cuenta de las ventajas que ofrece a nivel de seguridad.

6.1.2.1. CONCEPTO Y FUNCIONAMIENTO DE UN CERTIFICADO DIGITAL

Un Certificado Digital, es un documento electrónico emitido por una entidad reconocida a nivel mundial (Ej.: Verisign[8]), la cual llamaremos de ahora en

[8] http://www.verisign.com

adelante Entidad Certificadora. Existe un caso particular en el cual es posible usar una Entidad Certificadora local a una organización o a una serie de organizaciones que desean comunicarse entre si y confiar en los documentos que se intercambian entre ellas, por medio de Certificados locales a la organización.

“El propósito principal de un certificado digital es comprobar que la clave pública contenida en el certificado pertenece a la entidad a la que se emitió el certificado.”[7]

De esta manera el cliente o el servicio que trate de interactuar con este servidor de manera segura, tendrá la certeza de que esta comunicándose efectivamente con quien desea comunicarse y podrá enviar los mensajes por un canal seguro (SSL), usando encriptación por medio de la llave pública del certificado de la entidad, la cual a su vez podrá desencriptar los datos con la ayuda de su propia llave privada que fue emitida por la Entidad Certificadora antes de comenzar a funcionar.

Al principio, cuando el servidor todavía no puede recibir requerimientos Web, y desea instalar un Certificado Digital para su organización debe pasar por los siguientes pasos:

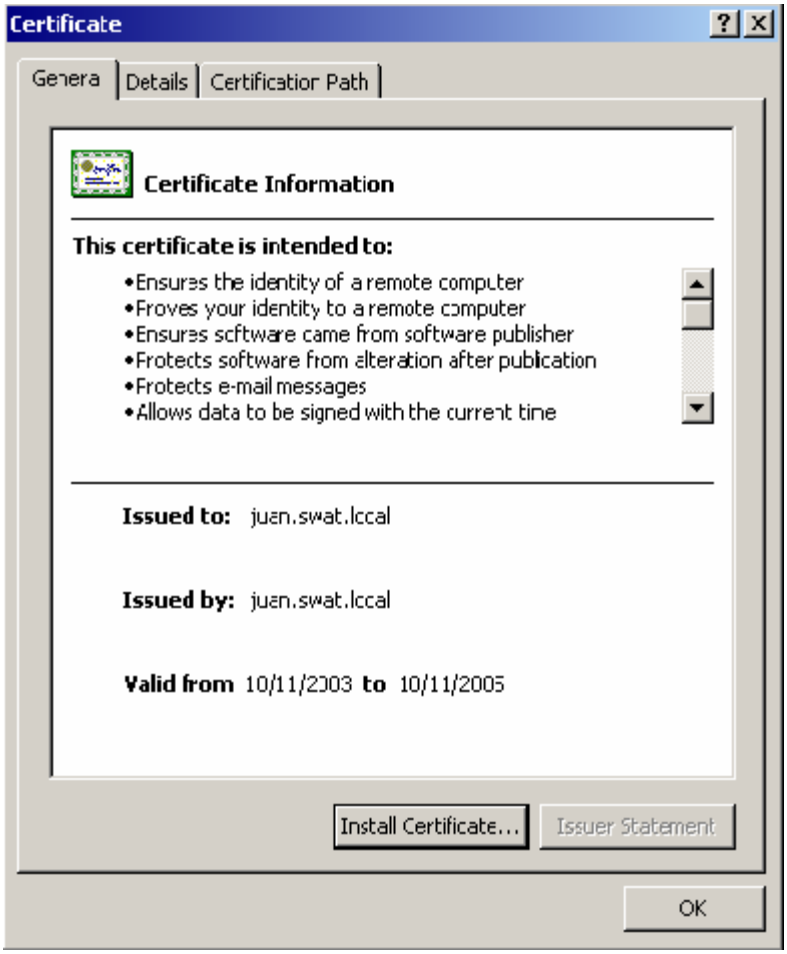
- **Contratar los Servicios de una Entidad Certificadora**. Para lograr esto la organización debe ponerse en contacto con la entidad a contratar y entregarle ciertos atributos, como son los datos de la empresa, y en retorno recibirá el certificado con su clave publica, su clave privada (utilizada para desencriptar los mensajes), y otra serie de datos incluidos en el certificado, que el cliente en el momento de recibirlo, podrá revisar y verificar para comprobar que la entidad con la cual se esta comunicando es quien dice ser.

- **Implantar el certificado en su servidor.** Esto se hace por medio del IIS, el cual permite colocar el certificado de tipo servidor para que pueda ser entregado a los clientes que requieran una comunicación segura.

- **Elegir que sitios desea que ofrezcan el servicio:** Para lograr esto se debe administrar el IIS, y revisar que directorios virtuales serán accedidos de manera segura, eligiendo así, que opciones se van a habilitar (SSL, requerir certificados de clientes, etc.).

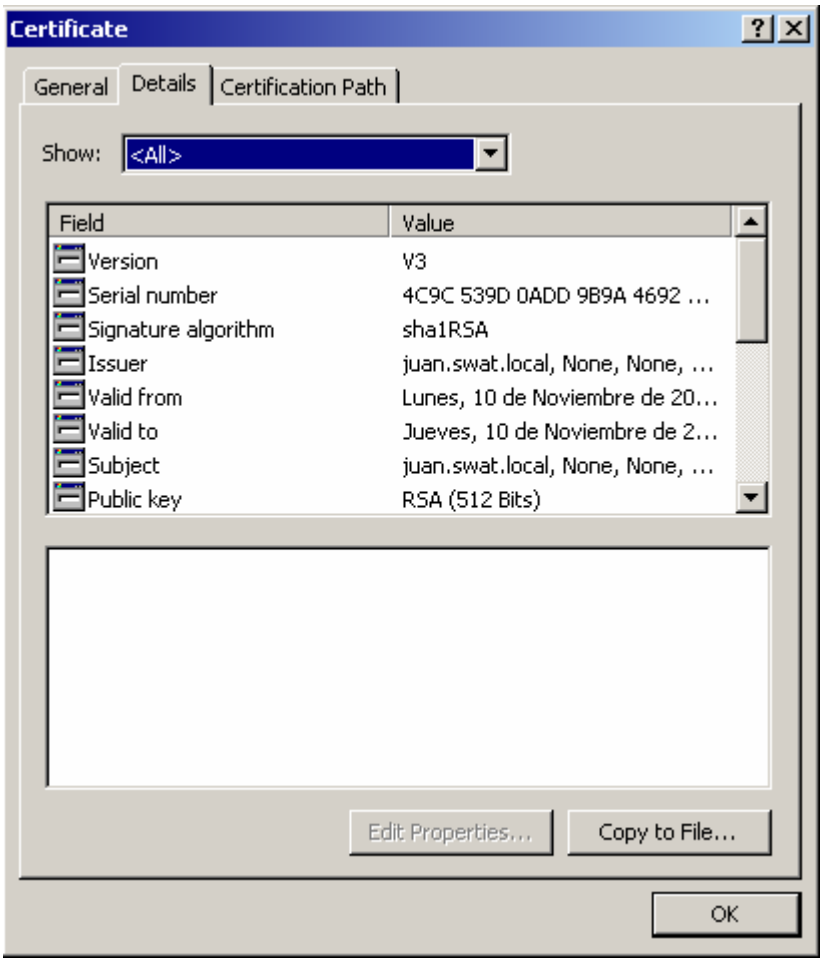
Una vez que la infraestructura del Web Server ha quedado configurada, los requerimientos de los clientes de manera segura (es decir usando https), recibirán, si es la primera vez que acceden a la página, el Certificado Digital correspondiente a la Entidad, el cual si se desea podrá ser revisado para verificar los datos de validez, y de la entidad que lo emitió. Si el cliente acepta este certificado se entablará una comunicación segura con el servidor y se sobreentiende que el cliente acepta que este certificado comprueba la identidad del servicio con el que se desea comunicar. (Ver Fig. 4, Fig. 5).

Figura 4. Certificado Digital Pág.



Fuente: Microsoft

Figura 5. Certificado Digital Pág. 2.



Fuente: Microsoft

6.1.2.2. COMO SE INSTALA UNA ENTIDAD CERTIFICADORA

Microsoft Windows Server ofrece una interfaz amigable y sencilla para poder instalar una entidad Certificadora en una organización, a continuación se enumeran los pasos generales para poder lograrlo[9].

[9] Para más información visite http://www.microsoft.com/windows2000/techinfo/planning/security/casetupsteps.asp

Antes de comenzar los pasos para la instalación de una Entidad Certificadora, es necesario saber que existen dos tipos de entidades y conocer sus diferencias, para saber cual es la que se desea instalar en el servidor. Los tipos de Entidades Certificadoras son:

Stand – Alone. Los Certificados que pueden ser emitidos por este tipo de Entidad Certificadora son para clientes o equipos que se encuentran por fuera del dominio. Adicionalmente esto quiere decir que el servidor es único, es decir no hace parte de una organización y por lo tanto los clientes no hacen parte de un dominio específico.

Enterprise. Se usa esta modalidad de Entidad Certificadora, si los certificados son para equipos o clientes dentro de un dominio, es decir dentro de la misma organización, para que esto funcione es necesario que los usuarios tengan cuentas de Active Directory en el servidor, cosa que no se necesita en el esquema Stand Alone, adicionalmente se debe tomar esta modalidad si deseamos que clientes de nuestra organización se puedan comunicar desde afuera y se puedan autenticar por medio de sus certificados con la organización. Este esquema será el que trataremos a profundidad por la afinidad con la solución que tratamos de plantear.

El Active Directory almacena información sobre los recursos de la red y provee los servicios que hacen que sea fácil localizarlos, administrarlos y de usar. El Active Directory también provee la administración de una forma de organización centralizada, administración y control de acceso a los recursos de la red.[10]

[10] http://microasist.com.mx/noticias/wn/mhawn0911.shtml

6.1.2.3. COMO SE CREA UN CERTIFICADO DIGITAL PARA AUTENTICAR A UN CLIENTE

Como ya lo habíamos establecido, un cliente puede ser cualquier persona que tenga una cuenta de usuario, o un servidor que se desea autenticar ante el sistema, es decir un servidor que tiene las cuentas de usuario asociadas a las personas o computadores.

Una vez que el usuario se encuentra conectado al sistema, debe ingresar al siguiente link: http://server/certsrv, donde Server es el servidor donde reside la Entidad Certificadora. En esta página se encuentra un ayudante que le generará automáticamente un certificado especial a la cuenta de usuario que se encuentra en el momento en el sistema, y le dará al usuario la opción de instalarlo.

Una vez concluido este proceso, si se desea tener una copia del certificado (en el caso de que desee acceder remotamente el sitio restringido), puede dirigirse en Internet Explorer a: Herramientas – Herramientas de Internet – Contenido – Certificados. En ese sitio podrá exportar el certificado, llevándose consigo su clave privada, opción que debe ser especificada por el usuario, y almacenándola en un archivo, el cual el usuario podrá transportar e instalar de la misma forma en que lo exportó.

6.1.2.4. COMO FUNCIONA UN INTERCAMBIO DE CERTIFICADOS DIGITALES ENTRE USUARIOS Y SERVIDORES

Para lograr que esta comunicación se lleve a cabo se deben cumplir los siguientes requerimientos:

a. En el IIS, debe estar aplicada la política que se refiere al mapeo de las cuentas del directorio activo con los certificados digitales.

- Para lograr esto en el servidor ejecute: Inicio – Herramientas – Herramientas Administrativas – Administrador del Servicio de Internet.

- En la raíz (la cual tiene el nombre del servidor), de clic derecho, y a continuación seleccione la opción de propiedades, allí encontrará una botón dentro de las propiedades maestras para poder editar.

- Selecciónelo y en el tab. de seguridad de directorio, seleccione la casilla que corresponde a mapear las cuentas del directorio activo.

b. Una vez completado esto, en el Administrador del Servicio de Internet, puede elegir cualquiera de las paginas que desea asegurar, dando clic derecho en el directorio virtual y a continuación en propiedades. Luego seleccione el tab. de seguridad y allí encontrará, que en la sección de comunicación con el servidor, el botón editar.

c. Selecciónelo, y a continuación aparecerá una pantalla, que le preguntará si desea requerir o aceptar certificados de clientes, así como si desea usar SSL, para la comunicación segura con el servidor.

d. Una vez echo esto, la próxima vez que un cliente quiera comunicarse con la página deberá hacerlo usando HTTPS, así como su certificado digital anteriormente emitido, esto con el fin de garantizar que es quien dice ser y poder ver la página solicitada.

Una vez que toda esta infraestructura ha quedado montada, podemos proseguir al entendimiento de este proceso.

Cuando un cliente instala el certificado en su explorador, este quedará guardado en la lista de certificados autorizados para usarlos en el caso en

que sean requeridos, esto se refiere a las páginas que anteriormente mencionamos, las cuales piden que para poder acceder a ellas, no solo sea por un protocolo seguro como es HTTPS, sino que también los clientes, deben proveer los certificados que fueron emitidos por el servidor.

De esta manera cuando el cliente intente acceder, deberá tener listo su certificado, de lo contrario el servidor no le permitirá el acceso y no podrá ver los servicios asociados.

6.1.2.5. ARQUITECTURA B2B CON USO DE CERTIFICADOS

En el caso particular que hemos estado trabajando, la aplicación puntual de los certificados digitales, viene cuando necesitamos que nuestro servidor primario se comunique de manera segura con los servidores secundarios que son los que tienen almacenado los servicios Web.

Entonces, para que nuestra arquitectura funcione, debemos crear cuentas de usuario para el servidor primario en ambos servidores, dándole los permisos necesarios para poder acceder a las páginas que tienen los servicios Web, y se deben generar los certificados digitales correspondientes, que el servidor primario usará, en el momento de comunicarse con los servidores secundarios para acceder a la información.

Planteamos entonces la necesidad de que cada servidor secundario tenga una entidad certificadora la cual creará el certificado para el servidor primario, y que el servidor primario lo instalará en su máquina para que pueda servir en el proceso de la comunicación segura.

No abordamos el problema de darle un certificado a cada usuario que desee comunicarse con el servidor primario, porque sería muy costoso e innecesario, ya que hasta el momento en nuestro modelo de seguridad, estamos asegurando la interacción entre las organizaciones. Mas adelante se tratará el cómo haremos que la comunicación sea segura entre los clientes y los servidores.

Ahora veremos como el framework de .Net integra la teoría de los certificados digitales en el desarrollo de Web Services seguro, esto con el fin de ilustrar como sería un escenario puntual de implementación de lo anteriormente mencionado.

6.1.3. COMO FUNCIONA LA AUTENTICACIÓN Y LAS FIRMAS DIGITALES CON WSE (*WEB SERVICES ENHANCEMENTS*)

En el capitulo anterior la solución fue basada en las herramientas ofrecidas para la seguridad en un servidor Microsoft; ahora debido a la escogencia de Visual Studio .Net como herramienta de desarrollo, se analizará un paquete desarrollado específicamente para esta plataforma, y orientado a contrarrestar problemas clásicos de seguridad (para los cuales existen estándares dados por la W3C[1]), y para los cuales este paquete ofrece diferentes soluciones.

6.1.3.1. AUTENTICACIÓN (TOKENS)

WSE utiliza dos métodos para reconocer al usuario por medio de un token, estos son UsernameToken y BinarySecurityToken[8].

UsernameToken funciona parecido a un usuario con contraseña, solo que con WS – Security existe la posibilidad de agregar seguridad de encripción a la contraseña de tal forma de que no sea tan fácil para un usuario anónimo clonarla y acceder al servicio como un usuario permitido.

WSE permite además de la encripción de la contraseña agregar un *TimeStamp* con el fin de dar un tiempo de validez al mensaje de identificación, esto con el fin de evitar ataques externos, de tal modo que si el mensaje es interceptado y modificado y luego reenviado, el servidor detecta un tiempo mayor a establecido y da como invalida la comunicación.

Pero no solo con esto se puede evitar un ataque, también se plantea dentro de la filosofía de WSE el no tener mas de un token dentro de un archivo y esto también es validado del lado servidor, de tal forma que el funcionamiento de este token pueda garantizar la identificación de un usuario valido. También existe un método adicional a este proceso y es tener un Proveedor de autenticación la idea es que se pueda definir una forma de guardar las parejas de usuario y contraseña, cono el fin de ser recuperadas luego en el momento de obtener un request y poder verificar su validez.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
      ...
   <wsse:Security
      soap:mustUnderstand="1"
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
     <wsse:UsernameToken
        xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
        wsu:Id="SecurityToken-58564463-5bdc-4a6b-a7fb-94a0d7357a20">
      <wsse:Username>Joe</wsse:Username>
      <wsse:Password Type="wsse:PasswordDigest">
        gpBDXjx79eutcXdtlULIlcrSiRs=
      </wsse:Password>
      <wsse:Nonce>
        h52sI9pKV0BVRPUolQC7Cg==
```

```
      </wsse:Nonce>
      <wsu:Created>2002-11-04T19:16:50Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <PersonalHello xmlns="http://tempuri.org/" />
 </soap:Body>
</soap:Envelope>
```

Ejemplo de un mensaje usando UsernameToken

BinarySecurityToken esta forma de validación se da con el fin de manejar tokens existentes, esto quiere decir, que este token esta basado en Certificados X.509[11]. La forma de utilizarse es llenando un listBox con las posibles formas de autenticación que ofrece X.509 a partir de la clase Microsoft.Web.Services.Security.X509, el usuario selecciona una de estas y con esto se arma el token que se va a enviar al servidor.

6.1.3.2. FIRMAS DIGITALES

Luego de analizar el funcionamiento de la seguridad con el uso de tokens, se continua viendo que aun existe un problema de seguridad q se relaciona con que cualquier usuario puede copiar la información contenida en el mensaje SOAP y lograr una autenticación falsa, así que ayudado por estos tokens se propone también un uso de firmas digitales, esto quiere decir el uso de llaves privadas y publicas para una comunicación más segura, de tal forma que al mensaje se le pueda añadir una firma digital y con esto el receptor del mensaje pueda verificar su validez. La firma se crea haciendo uso del objeto X509SecurityToken como parámetro para la función constructora de la firma.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

[11] Estándar Usado para definir los certificados Digitales.
http://www.webopedia.com/TERM/X/X_509.html

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsrp:path
     soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
     soap:mustUnderstand="1"
     xmlns:wsrp="http://schemas.xmlsoap.org/rp">
    <wsrp:action
      wsu:Id="Id-b856ae70-7a1b-4895-a05c-5f6596ca4429"
        ...
  </wsrp:path>
        ...
  <wsse:Security
     soap:mustUnderstand="1"
     xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
    <wsse:BinarySecurityToken
      ValueType="wsse:X509v3"
      EncodingType="wsse:Base64Binary"
      xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
      wsu:Id="SecurityToken-f6f96b4b-23c5-421e-92ff-f1050d531e82">
     MIIGkzCCBXugAwIBAgIK  . . . 39Vmjd20Lw==
    </wsse:BinarySecurityToken>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
        <CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <SignatureMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
        <Reference URI="#Id-24cc3660-6f1a-41fe-a949-71d7ed9fc636">
          <Transforms>
            <Transform
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          </Transforms>
          <DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
          <DigestValue>
           /8iL3OP9mfzuixI/ilkhHMbatV0=
          </DigestValue>
        </Reference>
        <Reference URI="#Id-b856ae70-7a1b-4895-a05c-5f6596ca4429">
          <Transforms>
            ...
      </SignedInfo>
      <SignatureValue>
ZY4MhHzBYz+CBdAz1LhAFjy6QxQoKJoA7l2eG45QV0hDIJrmXwLEGrPnpX+uPan5+MS6h
m+oL
/sGTbKJ/DJMp/t5ZyqY1qvngGQLcYXRy538zemwFfeGN5R2wmOoUSeCBUqprQVUbnkz+ql
Vp/
5f7t7VGW2Ee55Q3ol+ApVoFQE=
      </SignatureValue>
      <KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference
  URI="#SecurityToken-f6f96b4b-23c5-421e-92ff-f1050d531e82" />
```

```
        </wsse:SecurityTokenReference>
      </KeyInfo>
    </Signature>
  </wsse:Security>
 </soap:Header>
 <soap:Body
    wsu:Id="Id-24cc3660-6f1a-41fe-a949-71d7ed9fc636"
    xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
  <PersonalHello xmlns="http://tempuri.org/" />
 </soap:Body>
</soap:Envelope>
```

Ejemplo del código con firma digital y BinarySecurityToken

Los tres elementos que se ven en el encabezado del mensaje y que corresponden a la firma digital, son los siguientes:

SignedInfo: Define los datos exactos que están firmados en el mensaje, y cual es el algoritmo usado para generar la firma.
SignatureValue: Corresponde al valor de la firma digital.
KeyInfo: Este elemento contiene la información del BinarySecurityToken que contiene el certificado, y como se creo la llave (publica) con la que se firma el certificado, con el fin de que luego con la llave (privada) se pueda descifrar la información.

Ahora se verá como los WSE ayudan a encriptar estos mensajes SOAP dando una solución estándar y facilitando el desarrollo de la seguridad, esto debido a que toda comunicación que se da entre dos entidades se basa en mensajes SOAP

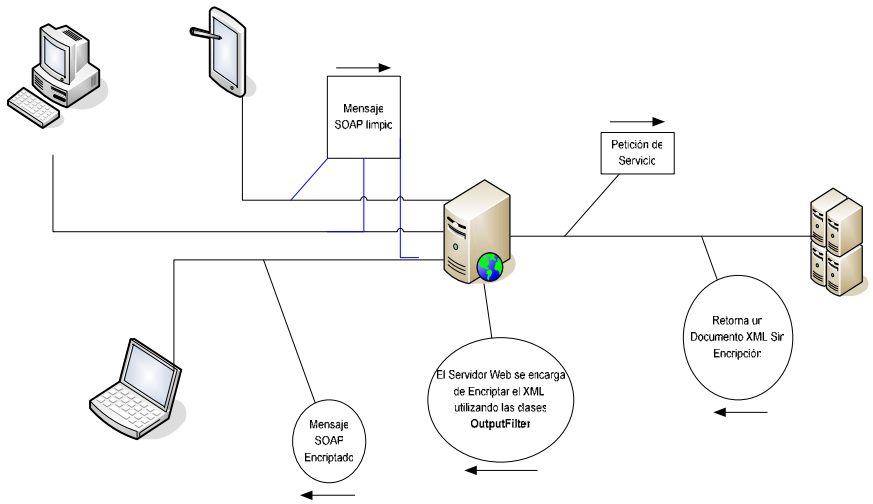
6.1.4. ENCRIPTACIÓN DE MENSAJES SOAP USANDO WSE (WEB SERVICES ENHANCEMENTS)

Los mensajes que utiliza un Web Service para comunicarse entre dos entidades son mensajes SOAP en XML, estos mensajes basados en la descripción de XML tienen la característica de tener sus propias definiciones

de tipos de datos y manejar la información a través de tags. Cuando abordamos el tema de encripción de un mensaje SOAP estamos hablando de cambiar la presentación de la información de tal forma que no sea entendible para una persona que pueda leer este mensaje en caso de que lo intercepte en medio de una comunicación, así que .NET ofrece un paquete adicional para el framework llamado Web Service Enhancement el cual ofrece unas características de encripción para los mensajes SOAP, en este capitulo se explicara el funcionamiento de este software y como se maneja la encripción, y el envío y recepción de estos mensajes, tanto desde el lado servidor como en el cliente[10].

En la figura 6 vemos un diagrama general de cómo se manejan los mensajes SOAP para una comunicación segura.

Figura 6. Flujo de los mensajes en una comunicación segura.



Fuente: Autores

6.1.4.1. WEB SERVICES ENHACEMENTS

6.1.4.1.1. Características.

Dentro de las características de seguridad en los Web Services que soporta WSE, para el filtrado de los mensajes de entrada y salida se utilizan dos objetos para manejar estos, el ***SecurityInputFilter*** y el ***SecurityOutputFilter*** los cuales incluyen:

- Firmas Digitales.
- Encriptación.
- Firmas y encriptación usando tokens para los usuarios.
- Firmas y encriptación usando certificados X.509.
- Firmas y encriptación usando tokens binarios.

6.1.4.1.2. Como funciona la encriptación del XML.

El protocolo de encriptación para un archivo XML dice que una parte del mensaje se puede encriptar cuando se hace uso de este proceso, el algoritmo de encripción se encarga de tomar el mensaje y reemplazar la parte del mensaje y cambiarlo por el resultado del algoritmo.

Mensaje sin encriptar

```
<soap:Envelope soap:xmlsn="http://www.w3.org/2002/12/soap-envelope">
 <soap:Header>
 ...
 </soap:Header>
 <soap:Body>
 ...
   <x:Order Type="Purchase" x:xmlns="http://example.com/order">
     <x:Payment Type="CreditCard">
       <x:CreditCard Type="Visa">
         <x:CardNumber>123456789123456</CardNumber>
         <x:ExperationDate>1108</ExperationDate>
       </x:CreditCard>
     </x:Payment>
     ...
   </x:Order>
   ...
 </soap:Body>
```

```
</soap:Envelope>
```

Mensaje encriptado

```
<soap:Envelope soap:xmlsn="http://www.w3.org/2002/12/soap-envelope"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:xsig="http://www.w3.org/2000/09/xmldsig#"
 xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext">
 <soap:Header>
  <wsse:Security>
   <xenc:ReferenceList>
    <xenc:DataReference URI="#OrderID"/>
   </xenc:ReferenceList>
  </wsse:Security>  ...
 </soap:Header>
 <soap:Body>
 ...
  <x:Order Type="Purchase" x:xmlns="http://example.com/order">
   <xenc:EncryptedData Id="OrderId">
    <xenc:EncryptionMethod
     Algorithm= "http://www.w3.org/2001/04/xmlenc#tripledes-cbc"
    <xsig:KeyInfo>
     <xsig:KeyName>My Symmetric Key</xsig:KeyName>
    </xsig:KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>...</CipherValue>
    </xenc:CipherData>
   </xenc:EncryptedData>
  ...
  </x:Order>
  ...
 </soap:Body>
</soap:Envelope>
```

Ejemplo de un mensaje SOAP encriptado [12]

6.1.4.1.3.Tipos de encriptación.

Los tipos de encriptación se dividen en dos:

Simétricas. Este tipo de encriptación se hace con el uso de llaves simétricas, esto quiere decir que el cliente y el servidor se comunican con una llave compartida que viaja a través de un mensaje seguro, este tipo de

[12] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wseencryption.asp

comunicación se aplica por ejemplo en Kerberos, el cual haciendo uso de un servidor de autenticación que recibe la solicitud de un cliente A haciendo la petición de querer comunicarse con otro cliente B, genera una llave simétrica entre A – B y con esta llave ambos clientes inician una comunicación segura.

Asimétricas. Este tipo de encripción se basa en el uso de llaves privadas y públicas se usa para la comunicación en Internet. Por ejemplo SSL se basa en el uso de este tipo de encriptación, la llave pública se utiliza para encriptar el mensaje, pero esta no se puede utilizar para desencriptarlo para esto se hace uso de la llave privada la cual solo la conoce cada poseedor. La llave pública se genera con un algoritmo que usa la llave privada.

6.1.4.1.4.La encripción en WSE

"Web Services Enhancements para Microsoft .NET (WSE) es una librería que implementa protocolos avanzados de Web Services" [13]

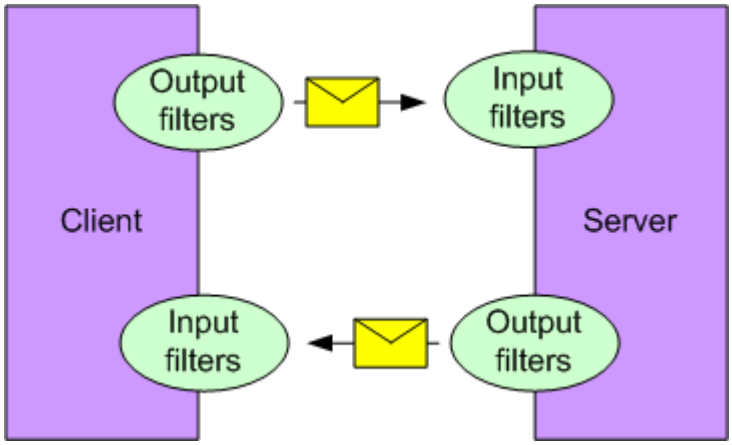
Esta librería tiene dos funciones llamadas SecurityInputFilter y SecurityOutputFilter las cuales se utilizan tanto del lado cliente como del lado servidor.

En el caso de un desarrollo dentro de otra herramienta de desarrollo, se entraría a revisar un paquete que ofrezca un servicio de encripción para los mensajes XML. Esto debería ofrecer las mismas ventajas, debido a que estos paquetes se basan en la estándar de XML dado por la W3C, y al seguir este estándar la forma de encripción debe ser la misma, así que solo se debería modificar la forma de implementación dentro de la herramienta de desarrollo.

[13] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/insidewsepipe.asp

El mensaje pasa por estos filtros para la encripción y la desencripción de tal modo que luego puedan se manejados dentro de la aplicación.

Figura 7. Flujo de los mensajes en una comunicación segura.



Fuente: Microsoft

En la figura 7 se muestra el paso de mensajes entre un cliente y el servidor y como funcionan los filtros de entrada y salida. El filtro de salida agrega una información específica en el encabezado del mensaje la cual después es leída por el filtro de entrada y con este encabezado interpreta la validez del mensaje. Este tipo de comunicación se llama centralizada, pero también existe la comunicación con filtros individuales, la cual maneja un TimeStamp en el encabezado del mensaje, y se llama TimestampInputFilter. Los TimeStamp se utilizan para conocer el momento de creación del mensaje y se puede utilizar para dar un tiempo de validez en la sesión de comunicación y con esto en caso de no recibir una respuesta pasado cierto tiempo, se da como invalida la comunicación y se termina.

Clases del TimeStamp[14]

```
public class TimestampOutputFilter : SoapOutputFilter{
  public override void ProcessMessage(SoapEnvelope envelope);
```

[14] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/insidewsepipe.asp

```
}

public class TimestampInputFilter : SoapInputFilter{
  public override void ProcessMessage(SoapEnvelope envelope);
}
```

Cada clase tiene un método diferente llamado *ProcessMessage* el cual es el encargado de recibir un parámetro de tipo *SoapEnvelope*, la librería ***Microsoft.Web.Services.SoapEnvelope*** es una extensión de la librería de .NET para el manejo de documentos XML, con esto se procesan los mensajes de entrada y los de salida adicionándoles la información del *TimeStamp* al mensaje.

Ejemplo de los mensajes filtrados[15]

Original message:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body/>
</soap:Envelope>
```

Output filtered message:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsu:Timestamp
      xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
      <wsu:Created>2002-11-14T19:03:27Z</wsu:Created>
      <wsu:Expires>2002-11-14T19:08:27Z</wsu:Expires>
    </wsu:Timestamp>
  </soap:Header>
  <soap:Body />
</soap:Envelope>
```

Input filtered message:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  </soap:Header>
  <soap:Body />
</soap:Envelope>
```

6.1.4.1.5.Trabajar con múltiples filtros en los PipeLines

[15] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/insidewsepipe.asp

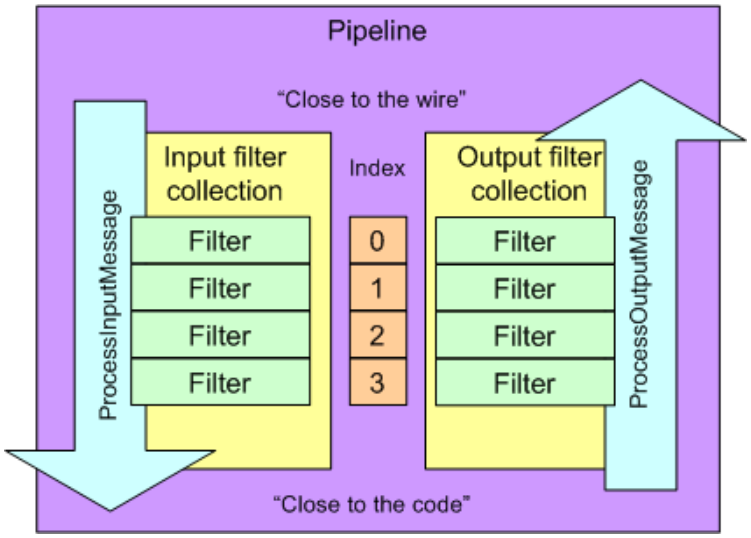
Espacio de Nombre (Namespace) Microsoft.Web.Services.	**Filtro de Entrada**	**Filtro de Salida**	**Propósito**
Diagnostics	TraceInputFilter	TraceOutputFilter	Escribir mensajes a un Log para facilitar la depuración.
Security	SecurityInputFilter	SecurityOutputFilter	Soporte para la autenticación, las firmas y la encriptación. (WS-Security)
Timestamp	TimestampInputFilter	TimestampOutputFilter	Soporte a los Timestamp (WS-Security)
Referral	ReferralInputFilter	ReferralOutputFilter	Actualización dinámica a los paths de enrutamiento (WS-Referral)
Routing	RoutingInputFilter	RoutingOutputFilter	Enrutamiento de mensajes

Tabla 1. Tabla de filtros en WSE

La comunicación con estos filtros individuales también puede hacer uso de diferentes tipos de filtros a través de PipeLines, esta es una tabla en donde se ven los posibles filtros y la función de cada uno.

Los filtros se le agregan al mensaje en el momento de su creación adicionándolos en el OutputFilter. Hay que tener en cuenta el orden de cómo se agregan al mensaje porque en el InputFilter se deben leer en el orden inverso[9].

Figura 8. Funcionamiento de los Filtros en los PipeLines [16]



Fuente: Microsoft

Los filtros que se pueden manejar son los siguientes:

TraceFilter: El cual se utiliza para hacer un rastreo del mensaje, con este filtro se puede hacer un seguimiento del mensaje, pero hay que tener en cuenta que esto puede ser un poco lento y disminuir el rendimiento y la velocidad de comunicación.

TimestampFilter: Se utiliza para conocer el momento de creación y el tiempo de validez del mensaje.

SecurityFilter: Este filtro se usa para el soporte de Autenticación, firmas digitales y encripción.

Todos estos filtros se pueden agregar o no a los mensajes, simplemente basta con hacer uso de la función ReconfigureDefaultPipeline y colocar o remover los filtros que se desean utilizar. Se debe tener en cuenta que al momento de sobrecargar esta función los cambios se aplican a los PipeLines

[16] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/insidewsepipe.asp

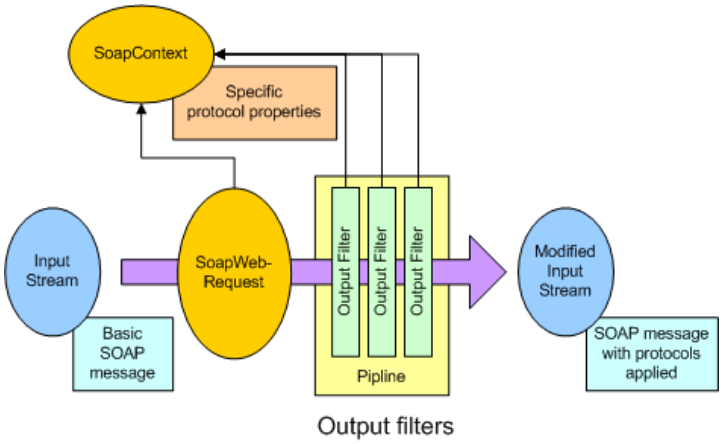
creados después de esta acción a los que se hayan creado antes de este procedimiento no se verán afectados. Esta modificación también se puede hacer en el global.asax[17] si se trata de una aplicación Web.

6.1.4.2. INTEGRACIÓN CON ASP.NET WEB SERVICES PROXIES

Los filtros de entrada y salida de los WSE son utilizados por el cliente a través de una clase Proxy llamada Microsoft.Web.Services.WebServicesClientProtocol. Esta clase es utilizada por el cliente para poder manejar los filtros de entrada y salida de los mensajes SOAP para entender mejor la forma de manejar estos mensajes vamos a conocer el comportamiento del SoapWebRequest.

La clase analiza la petición que llega en el mensaje SOAP y luego la pasa a través de los filtros, cada filtro puede tomar el mensaje y cambiarlo a su manera, debido a que puede necesitar desencriptar alguna información o agregar alguna al encabezado del mensaje.

Figura 9. Como se procesa un SoapWebRequest[18]

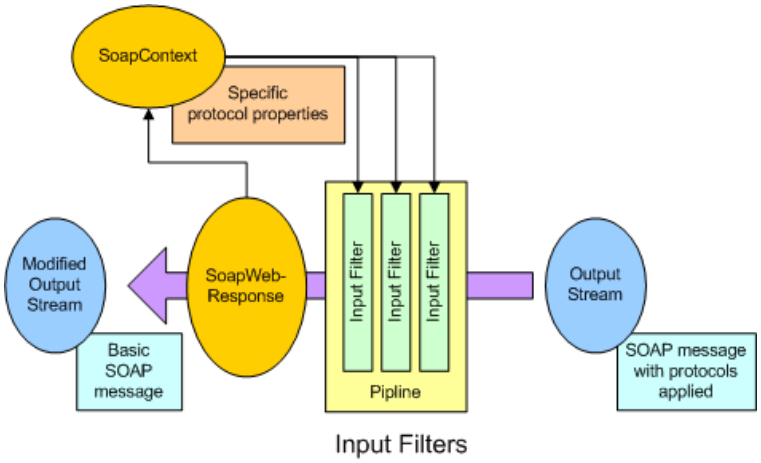


Fuente: Microsoft

[17] http://es.gotdotnet.com/quickstart/aspplus/doc/globalasax.aspx
[18] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/progwse.asp

Ahora analizamos el comportamiento del SoapWebResponse, este funciona al contrario del Request, este crea un mensaje SOAP con la respuesta y la pasa a través de los filtros, cada filtro modifica el mensaje dependiendo de lo que sea necesario, y luego envía el mensaje.

Figura 10. Como se procesa un SoapWebResponse[19]



Fuente: Microsoft

6.1.4.3. INTEGRACIÓN CON ASP.NET WEB SERVICES

La integración con los ASP.NET Web Services se hace a través de una nueva extensión SOAP del lado servidor llamada o embebida en la clase ***Microsoft.Web.Services.WebServicesExtension*** estas nuevas extensiones se hacen con el fin de garantizar que los filtros puedan procesar el mensaje SOAP asegurándose que todos los métodos necesarios sean invocados.

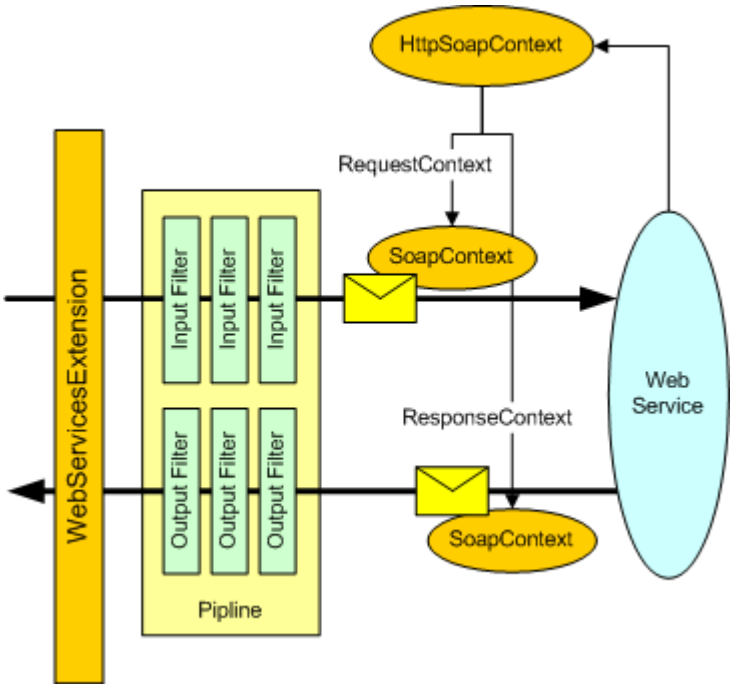
"Las extensiones del Web Services se aseguran de que todos los mensajes de entrada y de salida se les pueda atender (*Request*) y con esto generar

[19] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/progwse.asp

dos *SoapContext* (Mensajes SOAP) en donde uno va a tener la información de la solicitud (*Request Message*) y el otro la información de las respuesta (*Response Message*)"[20]

En la imagen se puede ver como el completo funcionamiento de la integración de los WSE con los Web Services.

Figura 11. Como se integran WSE con los Web Services [20]



Fuente: Microsoft

Ahora para el trabajo de investigación lo que se debe hacer, es analizar todas las ventajas de este paquete y poder encontrar cuales son las nuevas propuestas o mas bien las nuevas adiciones que debemos adaptar con el fin de lograr incrementar el nivel de seguridad y llevarlo a un nivel de identificación del usuario e identificación del rol al que puede tener acceso,

[20] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/progwse.asp

así con esto estamos haciendo uso de una tecnología existente, pero le estamos dando una adición que nos permitirá ofrecer un mejor servicio.

6.2. ANÁLISIS DE DEBILIDADES Y FORTALEZAS DE LAS TECNOLOGÍAS EXISTENTES

TECNOLOGIA	FORTALEZA	DEBILIDAD
Firmas Digitales	Asegura la autenticidad de quien envía el mensaje al tener una entidad adicional que verifique la veracidad del certificado.	Para que funcione de manera bidireccional todos los clientes deben tener certificados instalados en sus máquinas, esta opción no es viable para todo tipo de usuario.
Autenticación Usuario/Contraseña	Es ideal en el caso de pocos usuarios y servicios específicos para usuarios.	Si existen muchos usuarios la administración es más complicada.
Encripción Mensaje SOAP	Si se usa una llave lo suficiente fuerte, es decir 128 bits o más, ese sistema asegura que el mensaje SOAP no va a ser accedido fácilmente.	Al encriptar los mensajes se vuelven más grandes por lo tanto más pesados para el tráfico, adicionalmente se requiere que el cliente tenga acceso a la llave de encripción, dificultado la administración.

7. IMPLEMENTACIÓN DE UN ESQUEMA DE SEGURIDAD CON TECNOLOGÍAS EXISTENTES

Teniendo como base los conceptos anteriores, es hora de desarrollar una solución segura la cual se contrastará al final con la solución propuesta por los autores, analizando así sus diferencias y similitudes, así como las ventajas que presenta cada una de ellas.

7.1. MÉTODOS GENERALES DE ASEGURAMIENTO DE UN WEB SERVICE

Uno de los métodos más sencillos para aumentar la seguridad de los servicios Web consiste en asegurarse que la conexión entre el cliente de servicios Web y el servidor es segura. Esto se puede llevar a cabo mediante varias técnicas, dependiendo de la extensión de la red y el perfil de actividades de las interacciones. Tres de las técnicas más comunes y accesibles son: reglas basadas en un servidor de seguridad, SSL (*Secure Sockets Layer*) y redes privadas virtuales (VPN).

Si se sabe con exactitud qué equipos van a tener acceso a los servicios Web, puede utilizarse reglas de servidor de seguridad para restringir el acceso a equipos con direcciones IP conocidas. Esta técnica resulta particularmente útil si desea restringir el acceso a equipos en una red privada virtual (por ejemplo, una red LAN/WAN corporativa) y no desea mantener el contenido de los mensajes en secreto (cifrados).

Dentro de este plan de aseguramiento se entra a estudiar la forma en que se identifica (autenticación) al usuario y la forma en que se le va a permitir o no el acceso a las operaciones que el Web Service ofrece (Autorización).

Después de aplicar o de tomar algunos de estos métodos generales para asegurar el Web Service, pasamos a definir que método exacto se va a utilizar, dependiendo de las necesidades a cubrir dentro del servicio. Si se mira el común, se hace uso de la autenticación de usuario por medio de un usuario y una contraseña, en la cual consiste el primer Web Service desarrollado. Esto con el fin de poder realizar pruebas de ataques a esta forma de autenticación, y poder encontrar las debilidades que tiene y como corregirlas o mitigarlas.

Para este Web Service se define un plan de pruebas, con el fin de conocer que tanta seguridad ofrece y en que momentos puede o no llegar a ser útil este método.

7.2. PLAN DE PRUEBAS PARA EL WEB SERVICE SEGURO (USUARIO – CONTRASEÑA)

Este Web Service se basa en el uso de un usuario y contraseña con el fin de autenticar al usuario en una sesión única y segura, el funcionamiento general es:

El Web Service recibe una petición por parte de un usuario en donde le envía un usuario y una contraseña para que valide.

En el momento en que el usuario y la contraseña son validos, el Web Service crea una sesión – *session* – que mantendrá activa durante el tiempo que se mantenga la conexión con ese usuario (por cada usuario que se conecte se

creara una sesión.) el problema principal de este tipo de servicio es que nadie que no tenga un usuario valido podrá acceder a los servicios y esto estaría en contra de la definición de los Web Services en donde una característica principal es dar un servicio publico a todo tipo de usuarios.

Otro problema que tenemos en este Web Service seguro es que es sensible al ataque de – *Session hijacking* – en el cual un usuario no valido puede robarse la identificación de la sesión y reemplazar al usuario valido, así tendrá acceso completo a todos los servicios sin que el Web Service tenga manera de identificar este tipo de suplantación.

Basados en lo anterior, se pueden pensar en diferentes tipos de pruebas, el plan de pruebas será el siguiente:

Tratar de encontrar un usuario valido y mediante algoritmos de búsqueda de contraseña, hacer intentos para encontrar la contraseña correcta de ese usuario. A pesar de que estos algoritmos dependen de la complejidad de la contraseña, en algún momento podremos obtener ésta y poder así acceder a todos los servicios ofrecidos.

También se puede aplicar el ataque de – *Session Hijacking* – el cual es un proceso un poco mas complejo, ya que para esto necesitamos conocer la trama de validación de sesión, luego de esto tener algún software que nos de, de acuerdo a esta trama, el nombre de las variables de sesión; o del mismo modo hacer un ataque forzado, en donde se intentan probar diferentes nombres de variables, hasta encontrar una variable de sesión valida que nos permita suplantar una sesión valida.

Para la implementación de estas pruebas se cuenta con un desarrollo tipo cliente, en donde un Web Service hace uso de la seguridad (usuario -

contraseña) del Web Service seguro, cuyos resultados se ven en la documentación del Sistema de Pruebas.

Ahora, exploremos otro método para asegurar un Web Service pero que nos brinde no solo la seguridad de poder autenticar un usuario, sino también el hecho de que nuestro servicio no debe ser 100% privado, es decir poder manejar diferentes niveles de servicios con el fin de poder dar desde servicios totalmente públicos, como servicios especiales únicamente a un grupo de usuarios definido esto ultimo nos lleva a pensar en perfiles para los usuarios, en pocas palabras, Roles. Al momento de hablar de roles para los usuarios encontramos que esto es un tema que podemos aplicar para ciertos casos en donde no se desea hacer uso de un usuario y contraseña, pero que si se necesitan cierto tipo de controles y niveles de seguridad.

Encontramos entonces un plan de pruebas para saber si es seguro o no, aplicar en un Web Service la seguridad a través de roles.

7.3. PRUEBAS REALIZADAS AL WEB SERVICE SEGURO

7.3.1. INTRODUCCIÓN

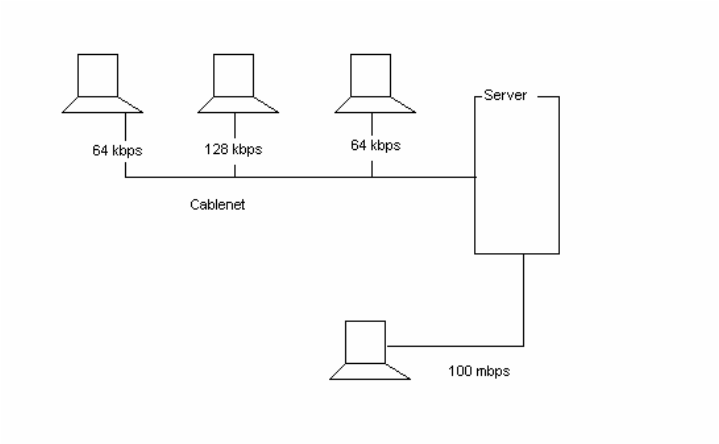
Después de haber enunciado en los capítulos anteriores los casos más frecuentes de ataques a los Web Services, se debe ahora hacer las pruebas del Web Service desarrollado con la tecnología hasta ahora conocida con el fin de probar si cumplirá con todas las necesidades, y de no hacerlo tratar de averiguar el porqué de ello.

En el texto siguiente se enumeran las pruebas que se hicieron contra el Web Service desarrollado, el cuál tiene incluida seguridad por medio del uso de

autenticación Usuario-Contraseña, y el cual guarda un identificador único de sesión por cliente que se conecte a la aplicación.

7.3.2. ESCENARIO DE PRUEBAS

Figura 12. Escenario



Fuente: Autores

Las pruebas se hicieron tomando un PC Athlon XP con un procesador de 1.8 Ghz y 512 Mb de memoria RAM, se usó un canal de 128 KB el cual fue proveído por un proveedor de Internet por fibra óptica. Cabe anotar que las pruebas se hicieron dejando el PC quieto sin ningún programa adicional corriendo además del Web Service.

7.3.3. DOCUMENTACIÓN DEL SOFTWARE DE PRUEBAS DEL WEB SERVICE SEGURO

7.3.3.1. PRUEBAS “DENIAL OF SERVICE”

Figura 13. Screen de la Aplicación

Fuente: Autores

En esta pantalla se puede probar los dos métodos que maneja el Web Service:

- Probar un numero de peticiones para ver hasta donde resiste el Web Service, esto consiste en ingresar un número aleatorio de peticiones y

luego haciendo clic en el botón Peticiones, así el sistema se encarga de hacer n peticiones al método *IsInSession()*.

- Login, con este podemos probar el método *SignIn (mail, password)* con el fin de ver si se puede acceder al Web Service o no, luego de darle la información correspondiente a un mail y una contraseña que sea correcta o que sea incorrecta, o sus diferentes combinaciones.

7.3.3.2. FUNCIONAMIENTO DEL PROGRAMA

Tenemos dos eventos cuyo código es el siguiente:

1. Código del Evento de Peticiones

```
int i = 0;
this.lstPeticiones.Items.Clear();

while (i < int.Parse(this.txtNumPeticiones.Text))
{
        string test = i + " - " + lh.IsInSession();
        this.lstPeticiones.Items.Add(test);
        i++;
}
```

2. Código del Evento de Login

```
lblResultado.Text = lh.SignIn(this.txtMail.Text, this.txtPass.Text);
```

7.3.4. ESTADÍSTICAS DE LAS PRUEBAS REALIZADAS AL WEB SERVICE SEGURO

Peticiones “Denial of Service”

Numero de Peticiones	Tiempo
5	2:01 segundos
50	13:06 segundos
500	2:12 minutos
5000	21:52 minutos

Tabla 2. Estadísticas pruebas Web Service Seguro

Pruebas de Diccionario para encontrar el password correcto

Sabiendo que tenemos un password de 4 dígitos y probándolo con el abecedario, o sea desde la a hasta la z, tenemos lo siguiente:

WXYZ → Donde cada letra representa las 26 posibles letras e iniciamos con la combinación *aaaa*, entonces, 26*26*26*26 = 456.976 posibilidades, y de acuerdo a los tiempos tomados, tenemos que por cada letra el programa se toma un tiempo de 0,23077 segundos, podríamos encontrar el siguiente tiempo:

456.976 * 0,23077 = 105.456 segundo
29:17:36 horas tomaría en recorrer todas las posibilidades.

Si tomamos ese calculo para probar cuanto tiempo tarda esta aplicación en llegar a encontrar la palabra test, tenemos:

20*5*19*20 = 38.000 posibilidades en un tiempo de **2:26:10 horas**[21]

[21] Este tiempo es basado en cálculos estadísticos.

8. PROTOTIPO FUNCIONAL DE UN WEB SERVICE SEGURO USANDO SEGURIDAD BASADA EN ROLES

Una vez realizado el Web Service seguro, se presenta entonces la solución planteada por los autores, haciendo especial énfasis en la facilidad y rapidez de la implementación de este esquema de seguridad usando las librerías desarrolladas, impactando al mínimo la tecnología existente y respetando los estándares sobre los cuales fueron definidos los Web Services.

Cuando se habla de asegurar algún sistema, es decir lograr restringir quien tiene acceso a la información (autenticación) y a que tiene acceso (autorización), existen muchas maneras de lograrlo, algunas se han mencionado en capítulos anteriores, todas ellas claro esta, enfocadas exclusivamente a los *Web Services*; entre ellas podemos recordar la autenticación usando Certificados Digitales y Firmas digitales.

Todas estas tecnologías han tenido como único propósito ofrecer alternativas de solución al problema de cómo poder restringir el acceso a determinados usuarios pero lo han logrado a expensas de sacrificar algunas de las fortalezas y ventajas que ofrece la arquitectura abierta, auto descriptiva, e ínter operable de los *Web Services*.

Cuando se habla de certificados digitales para lograr la autenticación de nuestros usuarios se presentan varios inconvenientes, entre ellos, necesitamos una entidad certificadora que garantice la validez y autenticidad de los mismos y debemos reestructurar nuestros métodos Web para que

soporten las firmas dentro del paquete SOAP[22]; esto de por si, crea un impacto muy fuerte en la arquitectura presente en cualquier organización.

Los Web Service Enhancements, plantean una serie de elementos que se pueden usar de manera única o combinada para lograr el acceso a los Web Services realizados con el framework de Microsoft. Esta aunque es un framework robusto y completo para lograr asegurar un Web Service, es incompleto ya que presenta como alternativa para solucionar la autenticación de un usuario, el uso de firmas digitales apoyadas en certificados digitales, o el uso de un nombre de usuario y una contraseña en el mensaje SOAP. Esta última solución planteada afecta seriamente el paradigma de publicidad que predican los Web Services y adicionalmente agrega una carga administrativa demasiado grande, teniendo que controlar y crear usuarios y contraseñas para cada nuevo usuario que desee tener acceso a la información de nuestro sistema.

Se concluye entonces que es necesario manejar un mecanismo de acceso que verifique la autenticidad de un usuario pero que no tenga la misma carga administrativa y que no haga perder algunas de sus propiedades al *Web Service* como lo hace el uso de usuario y contraseña como mecanismo de autenticación.

A continuación se introduce el concepto de seguridad basada en roles, la cual permitirá controlar de una manera más optima el acceso a un *Web Service*, y eliminar los impactos fuertes en los desarrollos existentes, sin perder las características anteriormente mencionadas.

[22] SOAP: Simple Object Access Protocol.

8.1. INTRODUCCION

Cuando se habla de autenticación y autorización en cualquier sistema existen muchas aproximaciones a resolver el tema de permitir que un usuario tenga acceso a determinado recurso, siendo los más comunes y útiles los mecanismos de autenticación por usuario/contraseña y la autenticación basada en roles es por esto que debemos revisar a profundidad los pro y los contra de cada uno de ellos con el fin de determinar en que momento cada una de estas soluciones presta un mejor servicio a los usuarios.

La autenticación de usuario y contraseña se basa como su nombre lo dice en una pareja de llaves en la cual el nombre de usuario es único y la contraseña es una llave secreta que al combinarse y autenticarse contra un repositorio de datos, permiten o niegan el acceso a determinado recurso. Este esquema es ideal en aquellos casos en los que tenemos muchos usuarios los cuales tienen características como: individualidad, aislamiento y privacidad en la forma en que acceden a los datos y los privilegios que tienen sobre los mismos.

La autenticación basada en roles se basa en agrupar un conjunto de usuarios que por medio de algún mecanismo, ya sea una autenticación previa, o una conexión de confianza, se les asigne permisos sobre uno o más recursos; la ventaja de la autenticación por roles radica en que aceptamos que los usuarios existen y que tienen acceso al servidor, restringiendo entonces su acceso de acuerdo a los privilegios que su rol posea y permitiendo manejar desde el mínimo privilegio hasta el máximo que permita el organismo ante el cual se está autenticando.

Teniendo en cuenta esto podemos resumir sus características de la siguiente forma:

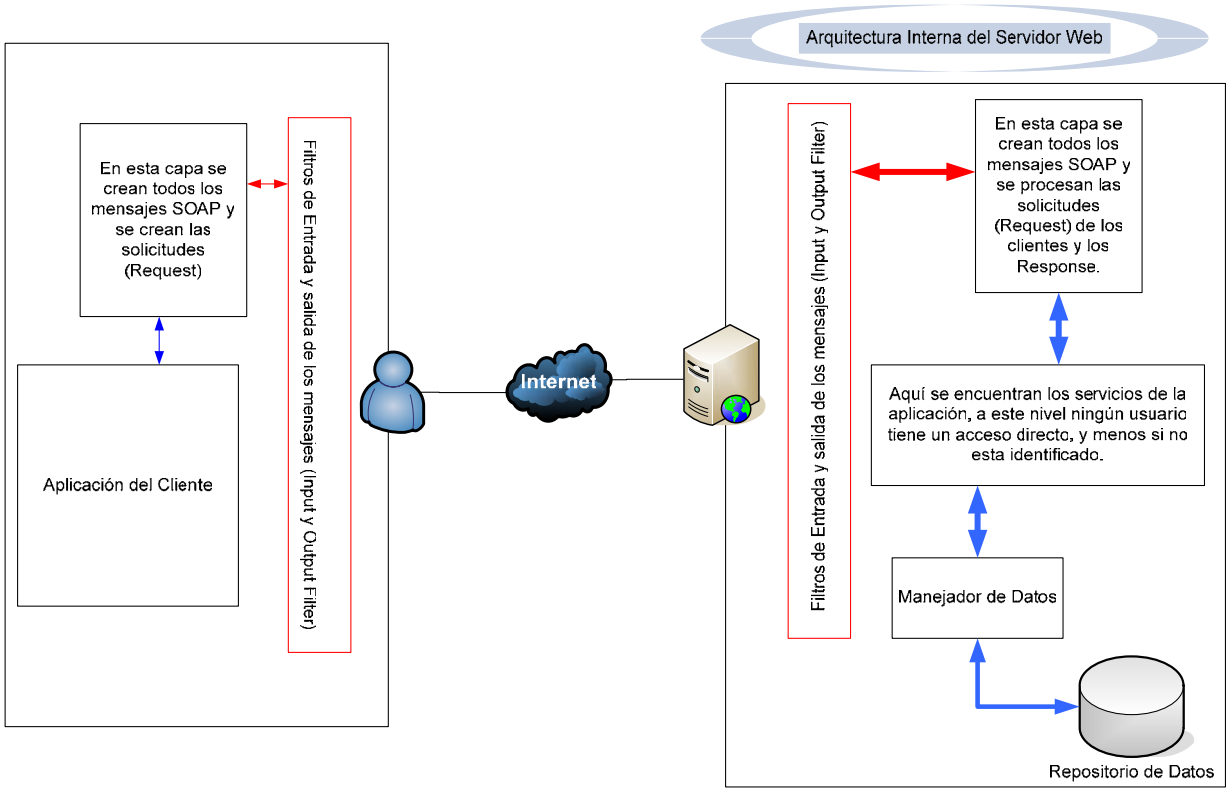
Seguridad Basada en Roles	Seguridad Usuario/Contraseña
Agrupa varios individuos que poseen los mismos privilegios	Un solo individuo
Se manejan roles y perfiles de acceso	Se manejan perfiles por cada usuario
Sencilla Administración	Administración compleja debido al crecimiento de los usuarios
Se pueden utilizar usuarios que no pertenecen a ningún rol (mínimo privilegio)	Se hace posible el uso de usuarios anónimos

Tabla 3. Comparación entre Seguridad por Roles y Seguridad de Usuario/Contraseña

Si tomamos en cuenta todo lo anterior y lo aplicamos a la teoría de los *Web Services*, encontramos que el uso de seguridad basada en roles es la mejor aproximación a solucionar el problema del acceso a un Web Service y sus métodos públicos, porque aunque existirán usuarios que pertenecen a determinado rol, y por lo tanto tendrán acceso a ciertos métodos, también alguien que no posea ningún rol, podrá acceder a aquellos métodos que no requieren ningún tipo de autenticación, siendo posible mantener la compatibilidad entre los usuarios públicos y los restringidos.

A continuación se describe la solución a la autenticación basada en roles la cual no atenta contra ninguno de los paradigmas anteriormente mencionados y con el mínimo impacto en el desarrollo existente.

8.2. ARQUITECTURA DE LA SOLUCION



8.3. IMPLEMENTACIÓN DE UN WEB SERVICE USANDO AUTENTICACION POR ROLES

Como se comentó en los primeros capítulos la solución planteada está desarrollada en el lenguaje C#. del .Net Framework de Microsoft, así que es necesario mencionar que requerimientos son necesarios para que el presente desarrollo funcione de manera adecuada.

8.3.1. REQUERIMIENTOS DE SOFTWARE

- .Net Framework (para que la solución pueda correr)
- IIS 5.0 o posterior (páginas .aspx de la solución)
- Windows 98 o superior (Windows 2000 Service Pack 3 preferible)

8.3.2. REQUERIMIENTOS DE HARDWARE

- 2 mb de espacio en disco duro
- Pentium II 500 mhz o superior
- 128 mb de RAM (256 preferible)

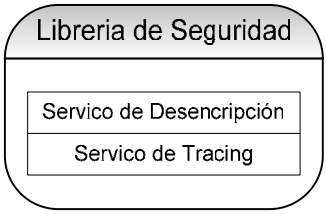
La solución consta de 3 partes, el cliente, el *Web Service,* y una librería que se encarga de la encripción y la autenticación de los roles.

8.3.3. LIBRERÍA DE SEGURIDAD

Figura 14. Diagrama de Bloques Librería de Seguridad para Servidor

Fuente: Autores

Figura 15. Diagrama de Bloques Librería de Seguridad para Cliente



Fuente: Autores

La librería de seguridad es el corazón de la solución planteada, se encuentra dividida en dos archivos, uno para el cliente y otro para el servidor, cada uno

de ellos posee métodos únicos para la manipulación de los archivos y habilita la capa de seguridad entre el cliente y el servidor.

La ventaja de tener estos dos archivos, es que el servicio puede estar asegurado en algunas partes, pero el cliente no necesita tener la librería si el solo desea acceder a los métodos públicos, adicionalmente, el cliente solo conocerá la forma de desencriptar los archivos, más no tendrá acceso a ninguno de los otros métodos que posee la librería en el lado servidor.

Para esta solución la librería de seguridad utiliza una llave simétrica de encripción, usando el algoritmo TripleDes, y unas tramas específicas para cada rol que tiene acceso al sistema, ellas se configuran por medio del administrador incluido en la solución.

Se presume entonces dos cosas: que el usuario ha estado en contacto con el proveedor del servicio y por lo tanto tiene o no una trama de seguridad que le dará un rol específico en el sistema y que adicionalmente tiene una versión similar de la librería de seguridad que le permitirá desencriptar los mensajes que se intercambian entre el y el *Web Service*.

8.3.4. WEB SERVICE

Los Web Services desarrollados pueden ser asegurados usando uno o más de los servicios que presta la librería de seguridad, estos son:

1. Método Administrador: Solo visible por aquellos clientes que tienen la trama de seguridad que los identifica como administradores.
2. Método Restringido: Solo visible por aquellos clientes que tienen la trama de seguridad respectiva, con esto intentamos mostrar como

podrían tenerse varios roles y por lo tanto diferentes niveles de acceso.

3. Método Público: Método que puede ser utilizado por cualquier cliente desde cualquier entorno.

Para lograr el uso de los servicios de encripción, seguimiento (*Tracing*) y manejo de roles, solo se necesita la referencia a la librería de seguridad y agregar un encabezado personalizado para manejar la trama, el cual también se encuentra en el mismo archivo de la librería de seguridad[23].

Una vez tenemos la referencia, lo único que se necesita para asegurar cada método web es agregar un atributo adicional al encabezado del método, dependiendo del servicio que deseamos usar, de la siguiente forma:

Ej:

Antes:

```
1    [WebMethod()]
2    Public string Metodo_Administrador()
3    {
4    //El código del método correspondiente
5    }
```

Después:

```
1    public Security.RoleBasedSecurityHeader roleBasedSecurity;
2
3    [WebMethod()]
4    [Security.RolesExtension("Administrador")]
5    [SoapHeader("roleBaseSecurity")]
6    [Security.ServerEncription()]
7    [Security.TraceExtension()]
8    Public string Metodo_Administrador()
9    {
10   //El código del método correspondiente
11   }
```

[23] Para más información ver el manual adjunto

En el ejemplo anterior tenemos un Web Service en su estado inicial, y como quedaría una vez se usen los mecanismos de seguridad planteados; tenemos entonces que se debe declarar una variable para manejar el header en el cual irá adjuntada la trama de seguridad; de igual forma necesitamos agregar que roles son permitidos para cada uno de los métodos(linea 4), estos nombres se compararán con los almacenados por el administrador de roles, para comparar que la trama coincida y poder dar una respuesta al cliente. La línea 6 y la linea 7 son opcionales ya que podemos o no utilizar encripción para los mensajes y el servicio de seguimiento para poder almacenar los mensajes enviados entre el cliente y el servidor.

8.3.5. CLIENTE

El cliente funciona de manera convencional, solamente debe agregarse la trama de seguridad y configurar el uso de los servicios de encripción si se desea tener acceso a los métodos restringidos por el sistema.

8.4. FUNCIONAMIENTO GENERAL

Lo que sucede dentro de la arquitectura anteriormente planteada es que el cliente usando la librería de encripción agregará la trama necesaria para poder acceder al método que el requiere. El servidor verificará la información tomando la trama y verificando si tiene o no acceso al método que esta pidiendo el cliente; de autenticarlo se le entregará la información que podrá o no estar encriptada dependiendo del encabezado de seguridad que se especifique, de lo contrario un error se devolverá al cliente.

Es claro que aunque el mecanismo de autenticación soporta roles, también soporta requerimientos en los cuales el usuario no ha entregado ningún tipo

de autenticación, soportando métodos totalmente públicos que no necesitan del uso de ningún mecanismo ni librería de seguridad, de esta manera logramos no afectar el paradigma de publicidad anteriormente planteado.

8.5. PLAN DE PRUEBAS PARA EL WEB SERVICE BASADO EN ROLES

Este Web Service se basa en el uso de tramas por parte del cliente en donde el servidor basado en éstas, identifica (autentica) al tipo de usuario, y con esto define el perfil especifico y los niveles de seguridad pegados a este.
El funcionamiento de este Web Service es el siguiente:

El cliente envía una petición a algún método, pero tiene la opción de enviar una trama de autenticación o no (esta trama se envía automáticamente por parte del cliente, debido a un proceso anterior en donde al cliente se le suministra una librería, la cual es generada automáticamente con la configuración específica para éste, que implementa este servicio y otros que mas adelante se especificarán.) dependiendo de esta trama el Web Service define el perfil de usuario y basado en esto le da acceso a ciertos métodos; en caso de que el servidor no reciba ninguna trama o una trama invalida, se le asignara automáticamente el perfil mas bajo o de usuario anónimo.

Un posible ataque que nace de la anterior descripción es el siguiente:

Intentar rastrear los mensajes enviados entre el cliente y el Web Service en el momento de iniciar la comunicación, ya que es en este momento en el que se envía la trama que le dará acceso al usuario con un determinado perfil. Para hacer esto se pueden probar varios intentos de ataque:

- Intentar interceptar los mensajes entre el cliente y el servidor para obtener la trama.

- Hacer diferentes tipos de peticiones a un método privado, con varias tramas generadas por una aplicación, este ejemplo tendría un funcionamiento muy similar al intento de averiguar la clave de un usuario.

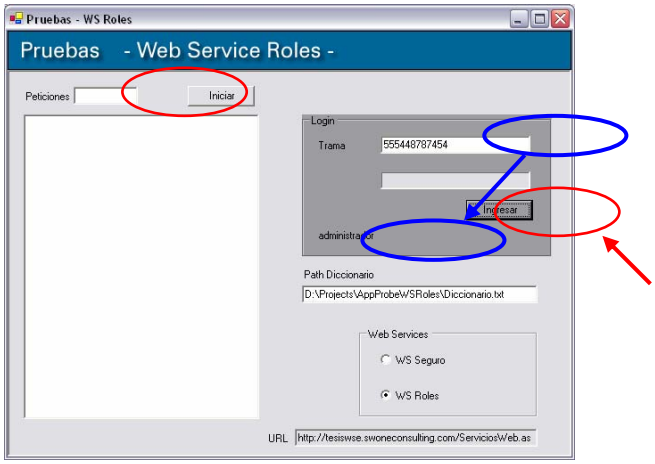
Para el caso del ataque anterior también se desarrolló una forma de reducirlo, ya que a los mensajes enviados por parte del servidor se les agrego un método de encripción en el formato (mensaje) XML enviado, así que si un atacante puede obtener una trama valida, todavía le haría falta la forma de desencriptar esta información, es aquí donde entra a discusión la funcionalidad de la librería enviada al cliente, ya que ésta no solo se encarga de enviar la trama respectiva, sino que tiene internamente todas las funciones de desencripción y encripción de los mensajes entre el usuario y cliente. Dejando así una capa de seguridad por parte del cliente, en donde es su responsabilidad ubicar esta librería en una zona segura y en lo posible de un mínimo de intentos de ataque.

8.6. PRUEBAS REALIZADAS AL WEB SERVICE DE ROLES

8.6.1. DOCUMENTACIÓN DEL SOFTWARE DE PRUEBAS DEL WEB SERVICE POR ROLES

8.6.1.1. PRUEBAS "DENIAL OF SERVICE"

Figura 16. Screen de la Aplicación



Fuente: Autores

En esta pantalla se puede probar los dos métodos que maneja el Web Service:

- Probar un numero de peticiones para ver hasta donde resiste el Web Service, esto consiste en ingresar un número aleatorio de peticiones y luego haciendo clic en el botón Peticiones, así el sistema se encarga de hacer n peticiones al método *getRol ()*.

- Ingresar, con este podemos probar el método *Metodo_Administrador()* configurando la trama que se va a enviar en el encabezado del mensaje SOAP con el fin de ver si se puede acceder al Web Service o no, luego de darle la información correspondiente a una trama que sea correcta o que sea incorrecta, o sus diferentes combinaciones, en el

resultado podremos observar cual es el rol que retorna este procedimiento. Aquí se debe tener en cuenta que no solo se envió la trama correspondiente sino que también se hizo la respectiva configuración del mensaje SOAP para tener la respuesta que se necesita.

8.6.2. FUNCIONAMIENTO DEL PROGRAMA

Tenemos dos eventos cuyo código es el siguiente:

1. Código del Evento de Peticiones

```
for (int I=0; i < numPeticiones; i++)
{
        if (this.rdbWSRoles.Checked)
        {
                WSRoles.CookieContainer = cr;
                WSRoles.getRol();
                test = i + " - " + wsr.getRol();
        }
}
```

2. Código del Evento de Login

```
WSRoles.RoleBasedSecurityHeader trama = new WSRoles.RoleBasedSecurityHeader();

trama.RoleBasedCode = this.txtMail.Text.GetHashCode().ToString();
_proxy.RoleBasedSecurityHeaderValue = trama;
```

8.6.3. ESTADÍSTICAS DE LAS PRUEBAS

Peticiones “Denial of Service”

Numero de Peticiones	Tiempo
5	2 segundos
50	19 segundos
500	2:58 minutos
5000	30:05 minutos

Pruebas de Diccionario para encontrar la trama

Suponiendo que tenemos una trama de 4 dígitos numérica y probándola con los números de 0 a 9, tenemos lo siguiente:

ABCD → Donde cada letra representa los 10 posibles números e iniciamos con la combinación *0000*, entonces, 10*10*10*10 = 10.000 posibilidades, y de acuerdo a los tiempos tomados, tenemos que por cada número el programa se toma un tiempo de 0,23077 segundos, podríamos encontrar el siguiente tiempo: 2.307,7

38:28 Minutos[24] tomaría en recorrer todas las posibilidades.

Si tomamos ese cálculo para probar cuanto tiempo tarda esta aplicación en llegar a encontrar la palabra combinación 5471, tenemos:

6*5*8*2 = 480 posibilidades en un tiempo de **1:51 minutos**[25]

Como podemos observar que es mucho mas rápido poder lograr averiguar una combinación de solo números, pero al ver los resultados obtenidos con la aplicación de pruebas (Ver Figura 16), nos podemos dar cuenta que solo nos esta retornando el rol a la que pertenece esa trama, ya que si se desea obtener información adicional no solo bastaría con obtener la trama correcta, sino que también se necesitaría tener la información suficiente que se debe adicionar en el mensaje SOAP y aun así, el usuario que recibiera este mensaje no podría conocer la información ya que este llega codificada. En el caso de las pruebas esta trama retorna el rol administrador, pero al hacer el

[24] Este tiempo es basado en cálculos estadísticos.
[25] ídem.

intento de probar algún método de este rol obtenemos una respuesta nula ya que no hay forma de decodificar la información.

8.7. ANÁLISIS DE DEBILIDADES Y FORTALEZAS DEL MECANISMO DE SEGURIDAD BASADO EN ROLES

CARACTERISTICA	DEBILIDAD	FORTALEZA
Encriptación	El algoritmo usado por ser TripleDes Simétrico, se usa la misma llave para encripción y desencripción, esto expone un riesgo de seguridad si se averigua la llave.	Los mensajes usan un algoritmo robusto de encripción lo que aumenta la seguridad del mensaje SOAP.
Log de Eventos	No tiene impacto	No tiene impacto
Bajo Impacto en el desarrollo Existente	Al ser sencillo de implementar, esto le resta maleabilidad por parte del usuario.	No se requiere mucha intervención ni horas ingeniero para volverlo funcional.
Facilidad de Implementación	Ninguna Disponible	Al reducir tiempo y costos, los Web Services pueden volverse más robustos en menos tiempo.

9. TECNOLOGIAS DE SEGURIDAD EXISTENTES VS SEGURIDAD BASADA EN ROLES

9.1. COMPARACION BASADOS EN SUS CARACTERISTICAS

Características / Tecnologías Existentes	Web Service Tecnologías Existentes	Web Service Con seguridad basada en Roles
Encriptación	SI	SI
Log de Eventos	SI	SI
Autenticación	SI	SI
Autorización	SI	SI
Impacto en el desarrollo Existente	SI	NO
Dificultad de Implementación	SI	NO
Firmas Digitales	SI	NO
Fácil Administración	NO	SI
J2EE	WEBSPHERE	NO DISPONIBLE
.Net Framework	WSE	PROTOTIPO DESARROLLADO

El anterior cuadro comparativo ilustra las diferencias puntuales que existen entre el desarrollo realizado con tecnologías existentes y el desarrollado por los autores, haciendo especial énfasis en sus debilidades y fortalezas, así como las plataformas que pueden cumplir con sus especificaciones.

Como se puede observar, la solución plateada, es decir el mecanismo de seguridad basado en roles, cumple con todas las características mencionadas, menos el uso de firmas digitales, el cual no se descarta desarrollar, pero que para los fines de la autenticación por roles básica no es necesaria, adicionalmente como se menciona al final del libro en los trabajos futuros, la implementación de la librería de seguridad para web services desarrollados en la plataforma J2EE no está desarrollada, planteando un tema interesante para aquellos que pretendan dar continuidad a este texto.

Adicionalmente se muestra que la solución planteada tiene ciertas ventajas sobre las existentes en el mercado, entre ellas la fácil administración al no necesitar guardar tantos usuarios y contraseñas para manejar la autorización de acceso a los Web Services.

10. CONCLUSIONES

Aunque los Web Services, han evolucionado de ser una teoría, hasta el punto de tener dos gigantes del Software, como son Microsoft y Sun Microsystems, diseñando frameworks, que los soporten, falta mucho por mejorar en el aspecto de la seguridad y control de acceso, porque si bien fueron diseñados para ser métodos públicos, no hay un estándar específico para poder controlar el acceso en el momento en que la información que se intercambia es sensible.

Un de los grandes inconvenientes de los Web Services es su falta de estandarización en muchos aspectos, cada empresa diseña su propia arquitectura y por lo tanto diferentes métodos de conseguir la información, esto desembocará en un problema cuando se trate de conciliar todas estas soluciones, para crear nuevas y más escalables aplicaciones.

Las conclusiones del presente proyecto se dividen de la siguiente manera:

10.1. IMPLEMENTACIÓN

Debido a que el presente desarrollo presenta las dos soluciones, es decir, la propuesta por los autores y la realizada usando las tecnologías existentes, es necesario aclarar que en el momento de una implementación, es decir aplicar estas librerías de seguridad a un Web Service desarrollado, se resalta que al usar la librería de seguridad de autenticación basada en roles, el tiempo de implementación se reduce dramáticamente, esto debido a que no se necesita instalar librerías desconocidas o configurar algoritmos innecesarios, solo se

debe referenciar la librería de seguridad y seguir unos pocos pasos para alcanzar un modelo robusto y seguro.

10.2. IMPACTO EN LOS DESARROLLOS EXISTENTES

La librería de seguridad ofrecida en el presente libro no impacta en lo absoluto el código ya desarrollado, permitiendo mantener la lógica del negocio existente, al solo afectar los atributos iniciales del Web Service.

10.3. COSTO DE IMPLEMENTACIÓN

El costo de una solución segura no solo se mide por las licencias que se deban adquirir sino también por el tiempo ingeniero y el esfuerzo en los cambios que se deben hacer a la infraestructura actual, es por esto que se demuestra que el mecanismo de acceso basado en roles, no solo es sencillo de implementar sino también el tiempo ingeniero necesario para lograr la seguridad es minimizado.

10.4. IMPACTO EN EL FUTURO

Teniendo en cuenta que se da un desarrollo puntual para asegurar los web services usando roles en el cual no se toma en cuenta la arquitectura de seguridad de la plataforma (al estar montada en un servidor windows, nos referimos entonces a las cuentas del directorio activo por ejemplo), se puede extender fácilmente para soportar la arquitectura de autenticación que haya diseñado el arquitecto de software responsable del Web Service a asegurar.

Con estas conclusiones no se pretende hacer una comparación para demostrar que algún método es mejor que el otro, es solamente una opción

diferente que se da a los usuarios con el fin de ser más concientes de que día a día las necesidades empresariales varían y es necesario dar en el mercado una amplia gama de opciones en la medida de lo posible, que sean extensibles, sencillas y fáciles de implementar.

11. PROYECTOS A FUTURO

Una vez terminada cualquier investigación es común que se encuentren proyectos o desarrollos faltantes, los cuales pueden ser interesantes de desarrollar, y que pueden servir a cualquier lector interesado en el tema en la búsqueda de áreas de investigación y profundización.

Los trabajos propuestos a continuación no pretenden cubrir todo el espectro de investigaciones pendientes en esta área, sino solamente mostrar los intereses de los autores a medida que este texto fue desarrollado.

11.1. DESARROLLO DE UN ADD-IN PARA IMPLEMENTAR LA SEGURIDAD EN UN PROYECTO DE WEB SERVICES EN VISUAL STUDIO .NET

Teniendo como base el presente desarrollo de las librerías de seguridad y las estrategias de seguridad utilizadas por los autores para lograr mejorar los niveles de acceso en Web Services, es apenas natural que se busque facilitar aún más el trabajo de aquel que sea responsable de aumentar la seguridad en un Web Service. Es por esto que los autores concientes de que no es suficiente con una librería de seguridad y una guía para su implementación, proponen que se debería realizar un Software a manera de wizzard que se encargara de hacer esto con la intervención mínima del desarrollador.

11.2. DESARROLLO DE LA LIBRERÍA DE SEGURIDAD PARA LA AUTENTICACION BASADA EN ROLES EN UN AMBIENTE J2EE

Los autores son concientes que aunque el desarrollo realizado puede ser consumido por cualquier cliente desarrollado en cualquier plataforma, es necesario crear librerías de seguridad que permitan manejar los servicios adicionales como es el de encripción. Adicionalmente se cree necesario desarrollar esta misma librería para asegurar servicios web desarrollados en el .Net Framework, para que también se pueda usar el mismo sistema en aquellos desarrollados en Java.

12. BIBLIOGRAFÍA

[1] World Wide Web Consortium. Dirección: http://www.w3c.org/. Fecha de Consulta: 30 Mayo 2003.

[2] Sun Microsistems. Dirección: http://es.sun.com/aprender_sobre/webservices/. Fecha de Consulta: 17 de Enero 2003.

[3] Autor Personal: CHAPARRO LOPEZ, Hilda Cristina
Título: Los web services como herramienta generadora de valor en las organizaciones / Hilda Cristina Chaparro López . Ingeniería y Universidad Vol. 8, no. 1 (ene.-jun 2004), p. 49-68

[4] Is SSL enough security for first-generation Web services?. Dirección: http://www.webservices.org/index.php/article/articleview/529/1/24/. Fecha de Consulta: 5 Febrero

[5] Web Services – Axis. Dirección: http://ws.apache.org/axis/java/security.html. Fecha de Consulta: 30 Marzo de 2003.

[6] CLABBY, Joe. Web Services Explained. Prentice Hall, 2003.

[7] Defending Your XML Web Service against Hackers, Part I. Dirección: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service09052001.asp, Fecha de Consulta: 29 de Enero de 2004.

[8] Programming with Web Services Enhancements 1.0 for Microsoft .NET, Dirección: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/progwse.asp, Fecha de Consulta: 23 de Febrero de 2004.

[9] Inside the Web Services Enhancements Pipeline. Dirección: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/insidewsepipe.asp, Fecha de Consulta: 23 de Febrero de 2004

[10] Encrypting SOAP Messages Using Web Services Enhancements. Dirección: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wseencryption.asp, Fecha de Consulta: 8 de Febrero de 2004.

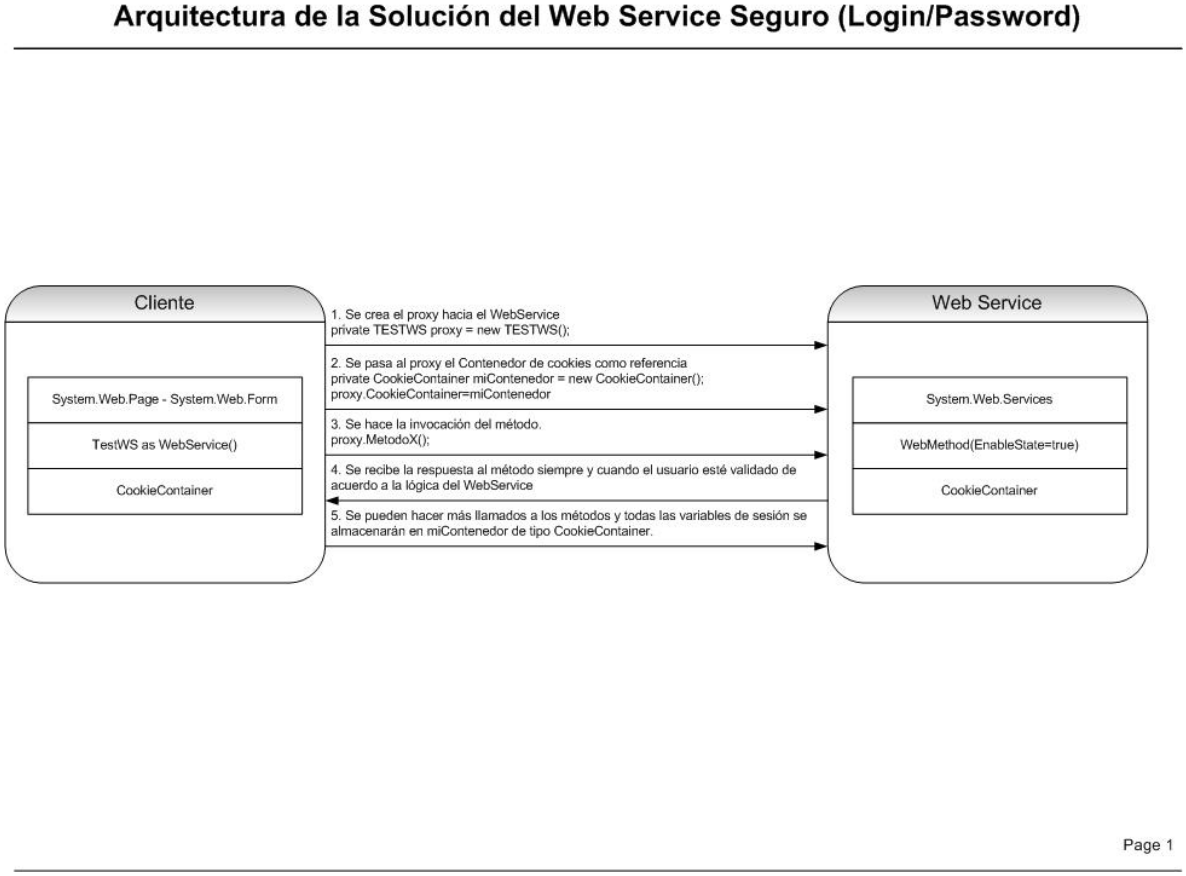
Still waiting for the Web Services Miracle. Dirección: http://www.business2.com/articles/web/0,1653,44928,FF.html, Fecha de Consulta: 6 de Febrero de 2004.

The Past, Present and Future of Web Services, part 2. Dirección: http://www.webservices.org/index.php/article/articleview/679/1/24/. Fecha de Consulta: 6 de Febrero de 2004.

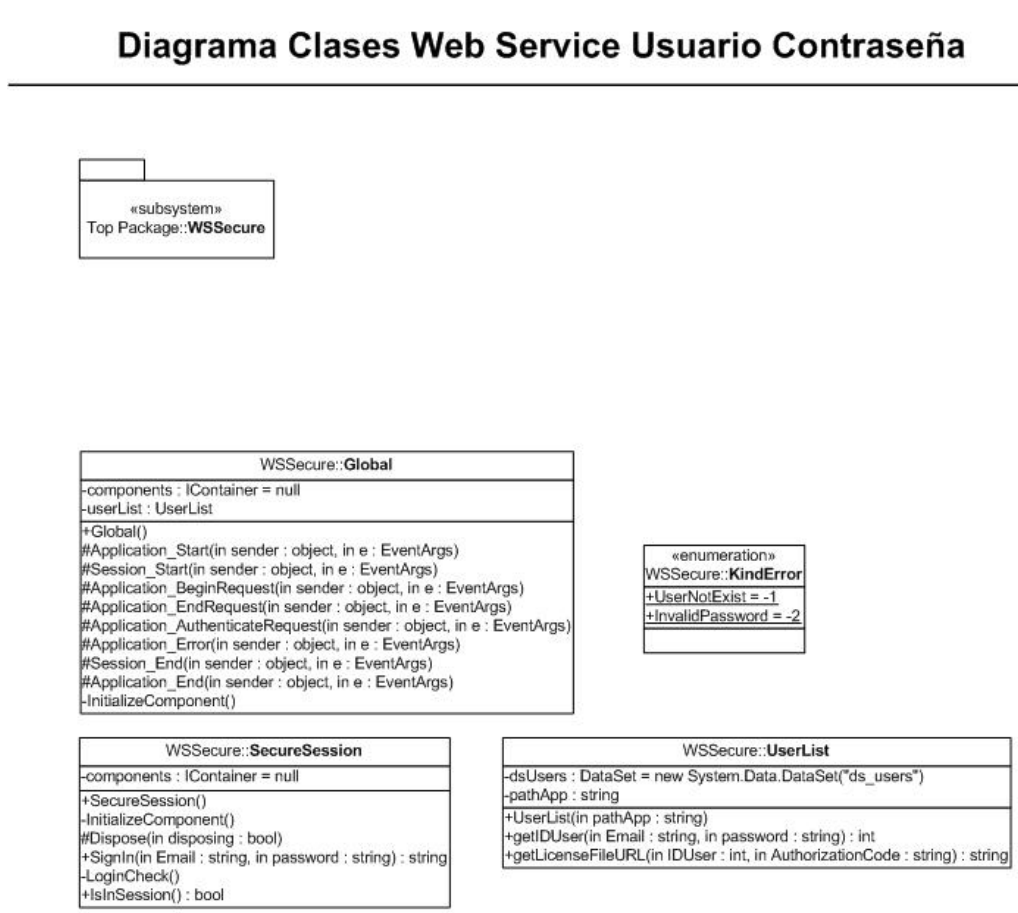
RSA grants royalty-free licenses on SAML. Dirección: http://www.webservices.org/index.php/article/articleview/353/. Fecha de Consulta: 8 de Febrero de 2004.

Security Evaluation, Microsoft Windows Server 2003 with .Net Framework and IBM Websphere. Dirección: http://www.atstake.com/research/reports/eval_ms_ibm/analysis/2.1.1.html, Fecha de Consulta: 8 de Febrero de 2004.

13. ANEXOS

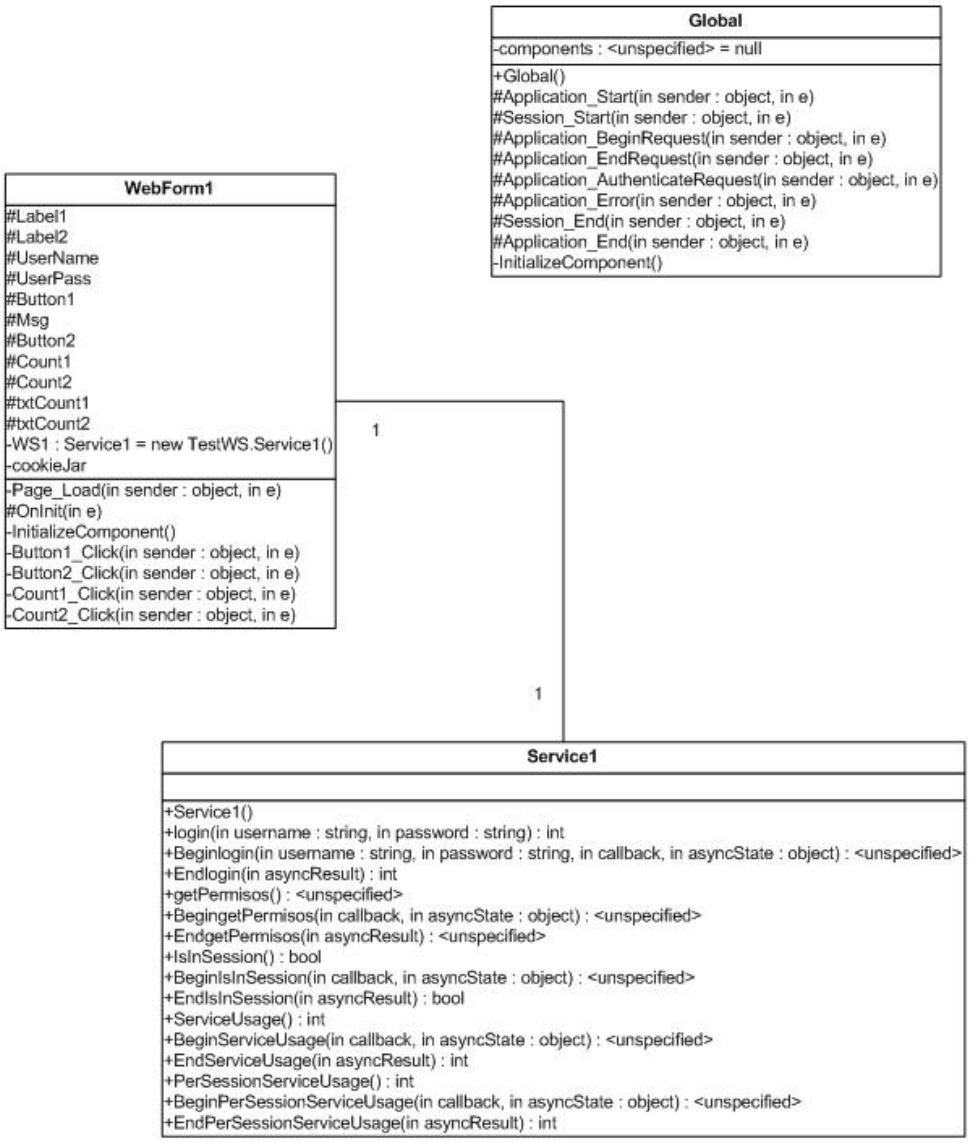


Anexo 1. Arquitectura de la Solución del Web Service Seguro (Login/Password)



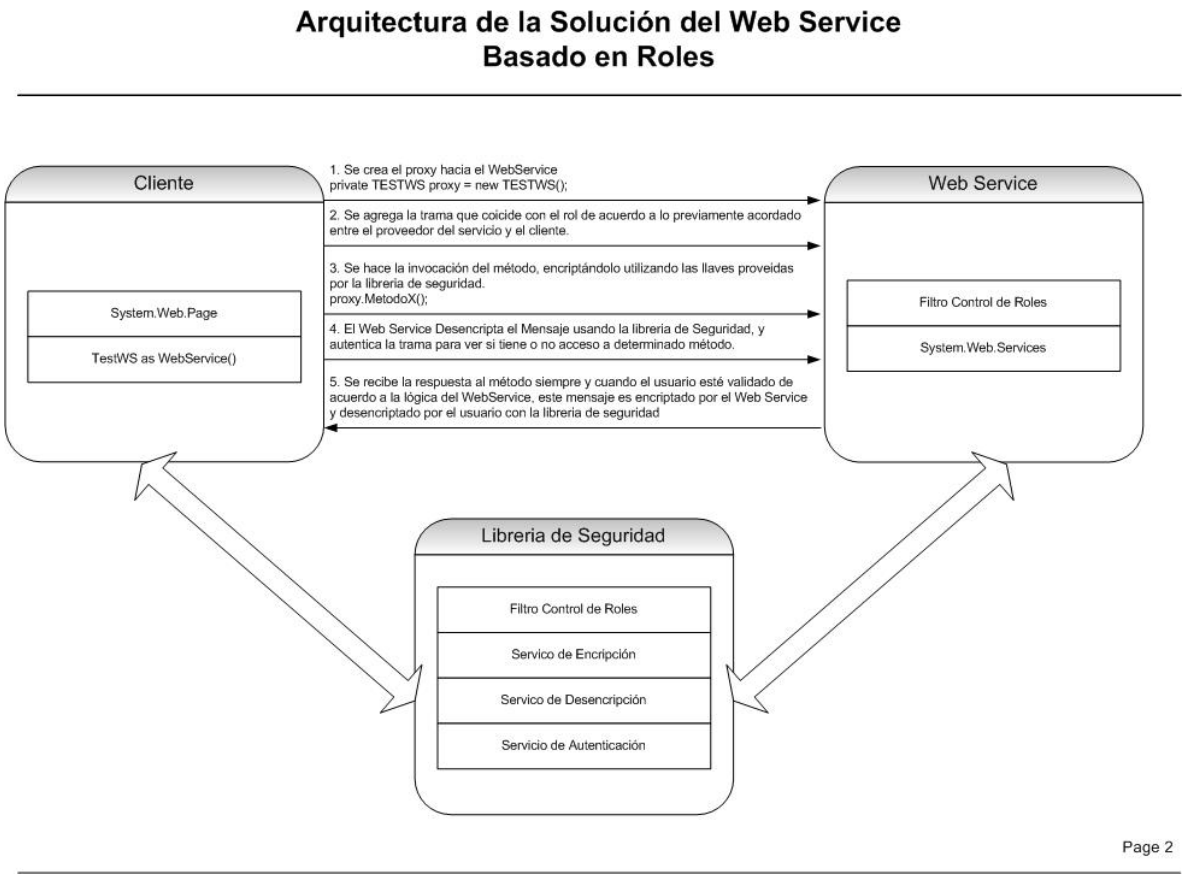
Anexo 2. Diagrama de Clases Web Service Seguro (Login/Password)

Diagrama Clases Cliente Web Service Usuario Contraseña



Page 2

Anexo 2. Diagrama de Clases Web Service Seguro (Login/Password)



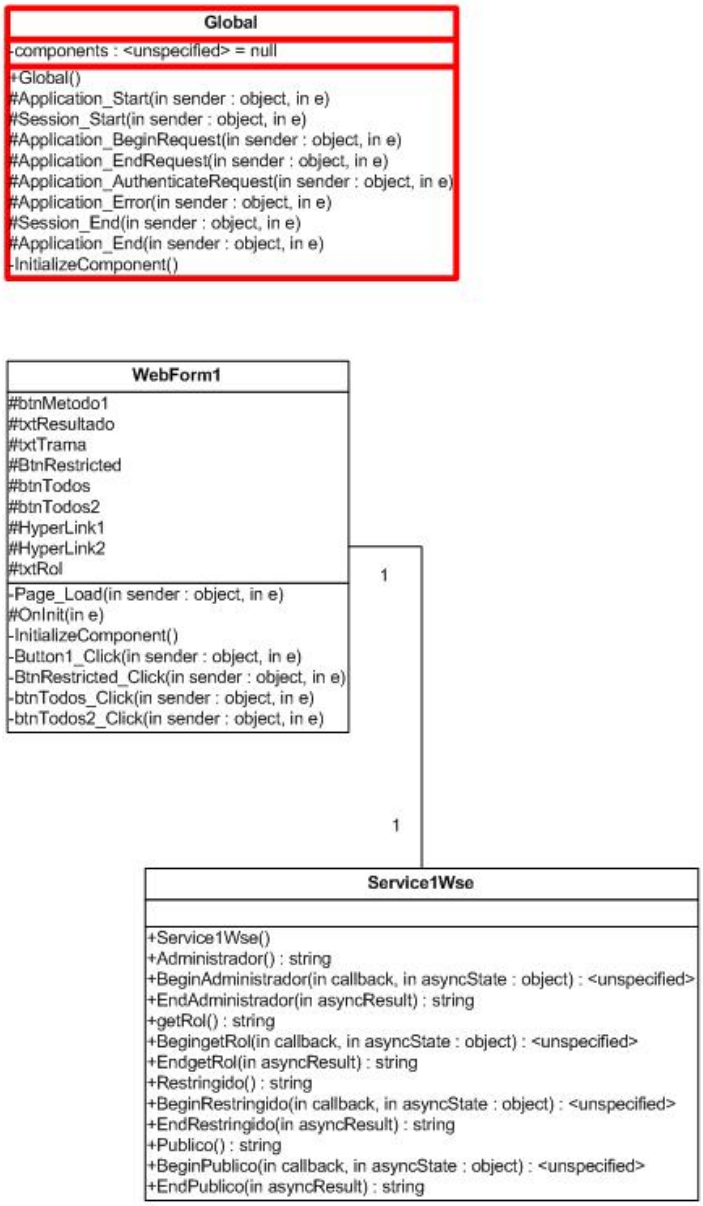
Anexo 3. Arquitectura del Web Service Basado en Roles

Diagrama Clases Libreria Seguridad (WebService Basado en Roles)

Page 1

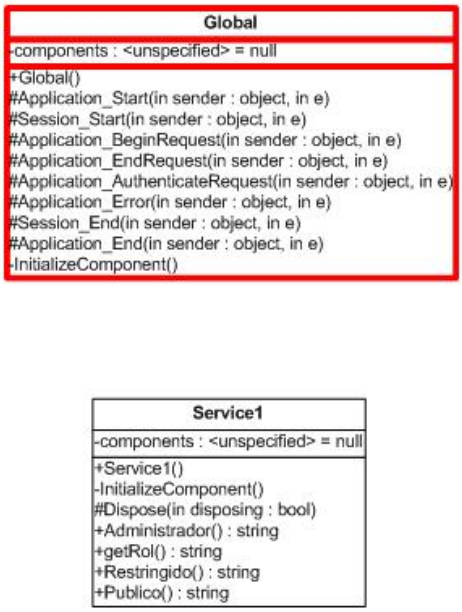
Anexo 4. Diagrama de Clases del Web Service Basado en Roles

Diagrama Clases Cliente Web Service basado en Roles



Anexo 4. Diagrama de Clases del Web Service Basado en Roles

Diagrama Clases Web Service Basado en Roles



Anexo 4. Diagrama de Clases del Web Service Basado en Roles