CENTRO DE CIENCIAS BÁSICAS
POSGRADO EN CIENCIAS EXACTAS, SISTEMAS Y DE LA INFORMACIÓN

TESIS

PARA OPTAR AL TÍTULO DE MAESTRO EN INGENIERIA DE SOFTWARE

MODELO DE AUTENTICACION DE SERVICIOS WEB MEDIANTE ASIGNACION DE IDENTIDADES VERIFICADAS

PRESENTA

Héctor Caudel García

DIRECTOR DE TESIS

Dr. Jaime Muñoz Arteaga

ASESORES

Dr. Francisco Álvarez Rodríguez
Dr. Ricardo Mendoza González
M.C. Eduardo Bautista Villalpando

Aguascalientes, Aguascalientes – Noviembre de 2010

Aguascalientes, Ags. Noviembre de 2010

Por medio de este conducto autorizamos al tesista:

L.I. Héctor Caudel García

La impresión de su documento de tesis titulado "MODELO DE AUTENTICACION DE SERVICIOS WEB MEDIANTE ASIGNACION DE IDENTIDADES VERIFICADAS", ya que cumple con los requisitos de contenido y de forma exigidos por la Universidad Autónoma de Aguascalientes.

Dr. Jaime Muñoz Arteaga
Director de Tesis

Dr. Francisco Álvarez Rodríguez
Asesor de Tesis

Dr. Ricardo Mendoza González
Asesor de Tesis

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Montreal, QC 3 de Noviembre de 2010.

A Quien Corresponda

Por medio del presente documento informo que la tesis titulada ***"Modelo de Autentificación de Servicios Web Mediante Asignación de Identidades Verificadas"***, ha sido revisada por un servidor haciendo las observaciones correspondientes y corregidas en su momento por el autor.

Dicha tesis se encuentra lista para su impresión así como para su defensa por parte del alumno **Héctor Caudel Garcia** como un requisito para la obtención del grado de *Maestro en Ingeniería de Software*.

Sin más por el momento y agradeciendo la atención que se sirva prestar al presente documento, reciba un cordial saludo.

Atentamente

M. en I. Luis Eduardo Bautista Villalpando
Profesor-Investigador
Depto. Sistemas Electrónicos
Centro de Ciencias Básicas

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES
Conmemoración del Bicentenario del inicio de la Independencia de México y del Centenario de la Revolución Mexicana

Centro de Ciencias Básicas

L.I. HÉCTOR CAUDEL GARCÍA
PASANTE DE LA MAESTRÍA EN CIENCIAS EXACTAS, SISTEMAS Y DE LA INFORMACIÓN
P R E S E N T E .

Estimado (a) Alumno (a) Caudel:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis y/o trabajo práctico titulado: **"Modelo de Servicios WEB mediante asignación de identidades verificadas",** hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 3 de noviembre de 2010
"LUMEN PROFERRE"
EL DECANO

DR. FRANCISCO JAVIER ÁLVAREZ RODRÍGUEZ

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES
CENTRO DE CIENCIAS BASICAS

c.c.p.- Archivo

AGRADECIMIENTOS

A DIOS, por todo lo que me ha dado, por haberme puesto en el camino la realización estos estudios, que nunca me dejo caer.

A mis asesores: El Dr. Jaime Muñoz Arteaga, El Dr. Francisco Álvarez Rodríguez, EL Dr. Ricardo Mendoza González y el Mtro. Eduardo Bautista Vallalpando por todo su apoyo, tiempo y consejo en la realización de esta investigación.

A los maestros que me impartieron clase a través de esta etapa.

Además a mis amigos de generación: Daniel, Erika, Fabio, Gustavo, Huizilopochtli, Irving, Luis Antonio, Ulises y Uriel por el su ayuda y apoyo durante este tiempo, así como el haberme brindado el regalo de su amistad.

A los profesores de Florida Atlantic University, Dr. Eduardo B. Fernández, Dra. Marie Petrie Larredondo y el Mtro. Ola Ajaj, los cuales me apoyaron durante mi estancia en el extranjero.

También se agradece enormemente a la Universidad Autónoma de Aguascalientes, por permitirme cursar mis estudios en ella, por los apoyos que me brindo tanto académicamente como económicamente, a Conacyt por otorgarme la beca con la que se curso esta maestría y el apoyo que me permitió viajar al extranjero.

A mi familia: Mi papas Juan Caudel y Ma. Estela García ejemplos de vida y amor, gracias por sus palabras de aliento en los momentos difíciles, a mis hermanas Claudia y Lupita por su apoyo y paciencia durante este tiempo.

A Goretti González, por amistad, su continuo apoyo y palabras de aliento.

RESUMEN:

La presente investigación fue realizada en la Universidad Autónoma de Aguascalientes, en Aguascalientes, México, en el periodo de 2008 a 2010 El propósito de la presente investigación consistió en el desarrollo de un modelo de autenticación orientado a servicios web enfatizando en la creación de identidades para los mismos. Para ello fue necesario el desarrollo del modelo que autentique a los servicios web como entidades, usando una flexibilidad en el uso de diversos tipos de credenciales, además de asignar cada servicio web involucrado una identidad independientes de la que posea el usuario que hace uso de él, así como el desarrollo del modelo mediante software y de las pruebas a aplicarse para su validación. El estudio se sustenta en los diferentes estándares de seguridad desarrollados para servicios web. El desarrollo de la investigación es llevada mediante un método descriptivo experimental. Se utilizo un caso de prueba basado en varios clientes y proveedores de servicios web, los cuales necesitan consumir mutuamente la información que ellos generan. Se planteo un caso de estudio basado en 3 los cuales intercambian información, a este caso de estudio se aplicaron los casos de prueba desarrollados, en los cuales por deducción el resultado fue negativo al no sobrepasar el modelo planteado.

Contenido

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

INTRODUCCION

La presente investigación está enfocada al tema de autenticación de servicios web, lo cual se podría definir en la comprobación de la identidad de servicios web que han sido desarrollados por diferentes proveedores, y que deben de interactuar entre si dentro de un proceso para un fin común.

El objetivo principal de este trabajo es definir un modelo que permita establecer la correcta autenticación entre servicios web, al definir como única, la identidad de cada uno de los involucrados de tal forma que en el momento en que alguien intente filtrar un servicio web no permitido, este pueda ser detectado y rechazado con facilidad, no importando que el intruso provea credenciales validas que pudieron ser substraídas a algún usuario autorizado.

Un servicio web se pude definir como una aplicación publicada en Internet, la cual para poder usarla se debe de hacer referencia al sitio donde se encuentra residente, con previa autorización de los propietarios, esto mediante el otorgamiento de credenciales [1]. Para hacer uso de esta se deberá adaptar correctamente la aplicación que quiera hacer uso de ella a las interfaces que este publica. Los servicios web tienen la cualidad se correr sobre cualquier plataforma y pueden ser desarrollados sobre una amplia variedad de lenguajes de programación, pueden tener un gran nivel de automatización, evitando que la intervención humana en varios procesos, además de que la mayoría de los protocolos sobre los cuales trabajan son estandarizados y abiertos al estar basados en XML [2]. Los servicios web trabajan a modo de componentes desplegados a través de la web.

Entre los ataques a la seguridad de los Servicios Web se encuentran; él "*pishing*", "*sniffing*", y "*middle man*", los cuales también son realizados sobre aplicaciones web, estos solo por mencionar algunos [3].

La relevancia en la creación de un modelo que permita la autenticación de servicios web, radica en la intensión de reducir el riesgo que se presenta ante las vulnerabilidades más comunes en un servicio web, las cuales pueden repercutir en la privacidad de la información que transita a través de la red. El modelo propuesto fomentaría la implementación de estrategias dirigidas a evitar la suplantación de la identidad propia de los servicios web, así como el copiado y modificación de la información intercambiada entre ellos. Estas situaciones de riesgo representan un aspecto importante a tratar, ya que la mayoría de los esquemas de seguridad tradicionales comprueban la identidad del usuario, dejando de lado la autenticación de los servicios web involucrados en un determinado proceso.

Descripción de capítulos

El proceso mediante el cual se desarrollara el trabajo de investigación, siendo el modelo de investigación, se basa en el propuesto en [4] mediante el método de investigación conceptual descriptiva.

La estructura general del protocolo de investigación se presenta gráficamente en la Figura 1. Cada sección será descrita brevemente a continuación:

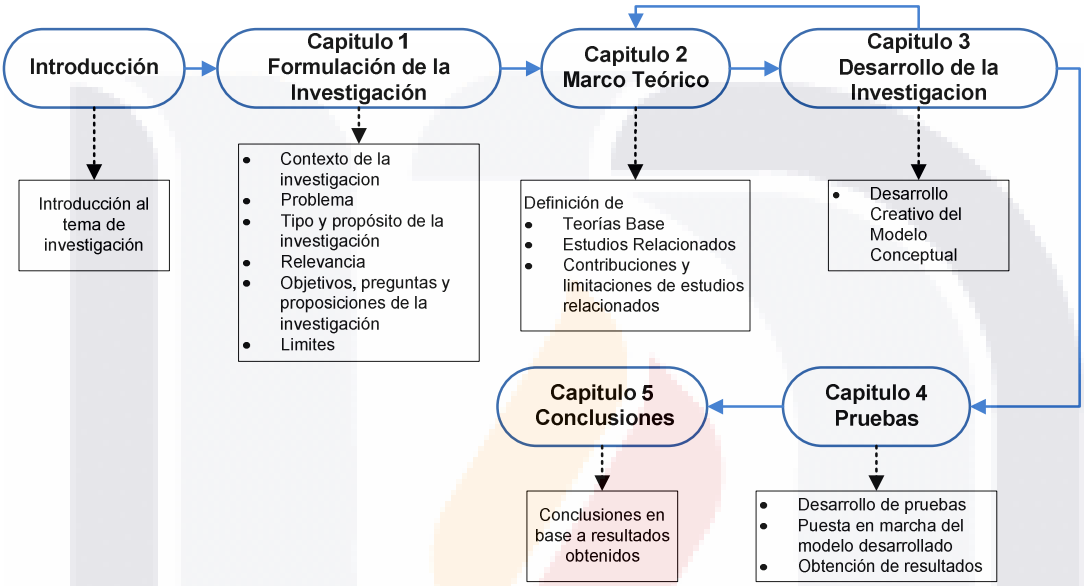


Figura 1: Diagrama de estructura de capítulos del trabajo de investigación

- **Introducción:**

En este capítulo se da una breve introducción de lo que será el tema de investigación y el objetivo a perseguir dentro de la misma.

- **Capitulo 1: Formulación de la Investigación:**

En este capítulo se definen los objetivos a perseguir, el contexto dentro del cual se desarrolla, se mencionara la problemática sobre la cual se abordara el tema de investigación y su estado actual. Además de definir la justificación por la cual se debe de realizar esta investigación, su estructura y finalmente las limitantes que esta tendrá.

  - Contexto de la investigación: Se describe de forma general los tópicos particulares en donde está ubicado el problema.
  - Problemática: es una declaración concreta de la situación no deseada, es decir, el problema de investigación que se pretende solucionar.
  - Tipo y propósito de la investigación.
  - Justificación.
  - Objetivos, preguntas y proposiciones de investigación.

- Limites.

- **Capitulo 2: Marco Teórico:**

En este capítulo se mencionaran las teorías base sobre las cuales esta investigación se desarrolla, además se describirán y comparan los diversos trabajos relacionados existentes, determinando lo que existe y se necesita, de tal forma que se puedan atacar las debilidades de estos y tomar las fortalezas con las que cuentan. Este capítulo cuenta con las siguientes secciones principales:

- Teorías Base.
- Estudios Relacionados.
- Contribuciones y limitaciones de estudios relacionados.

- **Capitulo 3: Desarrollo de la Investigación**

Aquí se describirá el modelo propuesto, se describirán sus componentes, las relaciones existentes entre ellos y su funcionamiento, la cual cuenta con el apartado:

- Desarrollo Creativo del Modelo Conceptual

- **Capitulo 4: Pruebas**

En este capítulo se planteara un escenario sobre el cual probar el modelo desarrollado, además de definir varias pruebas que podrán ser implementadas para poder validarlo.

- Desarrollo de pruebas.
- Obtención de resultados.

- **Capitulo 5: Conclusiones**

En base a los resultados obtenidos en el capitulo anterior, se elaboraran las conclusiones de esta investigación.

- **Capitulo 6: Referencias**

En este capítulo se muestra la bibliografía que da soporte a este trabajo de investigación.

- **Anexos:**

Los diferentes anexos que se obtuvieron en la investigación.

CAPÍTULO I:

1 FORMULACIÓN DE LA INVESTIGACIÓN

En este capítulo se expone el contexto dentro del cual se desarrollara la investigación incluyéndose también aquellos aspectos aun sin resolver dentro del trabajo. Asimismo, se describe la problemática identificada. Posteriormente se presentan los aspectos que justifican la realización de este trabajo de investigación.

En el Capitulo 1tambien se menciona el tipo y estructura de la investigación a desarrollar así como los objetivos que se persiguen en ella. Finalmente se presentan los límites que se establecieron, a fin de acotar el trabajo de investigación.

1.1 Contexto de la Investigación

El área principal de desarrollo a la que impacta, la presente investigación recae en el uso de servicios web seguros. Al ser tan amplio el campo de acción de estos, se enfocaron los esfuerzos al campo de la autenticación de servicios web.

Existe una gran cantidad de escenarios donde se puede visualizar la aplicación de la propuesta, por ejemplo organizaciones (e.g Instituciones educativas, de gobierno, privadas, etc), que pretenden integrar varios procesos vinculados a los servicios web que se encuentran aislados y que requieren compartir información confidencial entre si, ya sea por departamentos, áreas o interinstitucionalmente, por lo que es necesario homogeneizar el acceso a los datos, pero sin poner en riesgo la información compartida entre estos servicios.

Debido a que la seguridad implementada en los servicios web se realiza en varias formas, para esta investigación se considerara la implementación de la seguridad a nivel de mensajes y no a nivel de datos como es comúnmente implementada.

1.2 Problemática

El uso y aplicaciones que se hacen de los Servicios Web ha aumentado desde su creación, haciéndolos cada vez más eficaces y especializados en las tareas que estos realizan, pero a su vez también han crecido los problemas de seguridad que estos enfrentan debido a que conforme aumenta su uso, se han buscado formas más sofisticadas de burlar la seguridad, la mayoría de los problemas que enfrentan son similares a los que enfrentan las aplicaciones web puesto que estos son una evolución de las aplicaciones web estándar. En un inicio los desarrolladores solo se preocupaban de que el usuario ingresara sus credenciales, además de el correcto manejo y administración de permisos sobre los diferentes recursos a los cuales se intentaba, o se tenía permitido acceder a través de Servicios Web [1].

Posteriormente con el aumento en el uso de esta tecnología se crearon nuevos ataques a la seguridad, como lo es el *robo de identidad* o la *alteración de los mensajes SOA* que viajan a través de la red, para estos problemas ya se ha ideado soluciones como lo es la protección a nivel de transporte mediante el uso de HTTPS (HyperText Transfer Protocol Secure, es la versión segura del HTTPS), tickets Kerberos, SSL (Secure Socket Layer) entre algunos otros más.

Cuando se realiza el desarrollo de un sistema que hace uso de servicios web, es necesario modelarlo de tal forma que entre todos los involucrados pueden integrar en un proceso general y en conjunto conseguir un objetivo común. Regularmente el proceso de *orquestación* o *agregación* comienza con el servicio web, que valida al usuario mediante sus credenciales, y verifica que dicho usuario, acceda solo a los recursos que se permiten en base a sus credenciales.

Existen otros casos en las que un intruso (persona no autorizada que intenta acceder a un servicios web o a la información específica dentro de dicho servicio), logra apropiarse de las credenciales de otro usuario, e.g su nombre de usuario y contraseña, algún token de acceso o certificado, etc. De esta forma el intruso puede crear un servicio web que logre engañar al servicio o aplicación destinada a validar la autenticidad de la identidad de los usuarios, logrando así el intruso implementar el servicio web falsificado. Esto podría afectar información vital para la organización que haga uso de estos, o bien podría estar copiando y retransmitiendo los datos que circulan en la red mediante una redirección en las llamadas de servicios web. Otro ejemplo de ataque para este escenario puede ser dirigido mediante técnicas de "*sniffing*", o "*spoofing*" en el peor de los casos el intruso sería capaz de modificar la información.

Para poder realizar un ataque como el descrito anteriormente el intruso debería conocer la ruta en la cual se encuentra el servicio web objetivo. Este proceso es de por sí complicado ya que estos son expuestos en directorios UDDI, o bien implementar alguna técnica especial como el s*crawling*, adicionalmente, el atacante debería contar con las credenciales validas de algún usuario. Si el atacante cuenta con estos elementos, será capaz de crear un servicio web intruso el cual direccionará al servicio web objetivo, evitando las políticas de seguridad de otros servicios web que realizan un proceso de autenticación antes que el servicio web objetivo. Debido a que el intruso cuenta con credenciales validas, no se le negara el acceso a otros servicios web y recursos, puesto que la autenticación del usuario es válida. Aquí existe un fallo de confianza como se menciona en [5] dado que el intercambio de información entre un proveedor y un solicitante se debe de realizar en una forma esperada y entendida por ambas partes.

Actualmente no se ha encontrado un método que permita la protección desde el lado del servidor contra ataques de *Spoofing* [6]. Este problema ha sido mencionado en la literatura (ver ejemplo [7], en donde además se lista una seria de vulnerabilidades y problemas que los servicios web tienen actualmente [8], [9] Tomando en cuenta estas grietas en la seguridad de servicios web, específicamente en lo que respecta a la autenticación, es necesario un esquema tal que permita autenticar servicios web y no solo al usuario, esto contribuiría a evitar que intrusos impongan servicios web que no están autorizados en un entorno determinado.

1.3 Justificación

El constante avance en las tecnologías de la información permite generar nuevas alternativas para la distribución y manejo de los datos lo cual trae consigo múltiples beneficios visibles en varios entornos tales como: El empresarial, social, académico, militar, entre otros. Sin embargo dichos avances tecnológicos pueden ser utilizados con fines negativos o incluso criminales aprovechando las vulnerabilidades de otras tecnologías. Muchas de estas debilidades en la seguridad de los sistemas de información se vinculan fuertemente con la Autenticación tal como lo detalla Noor [10]. En este caso particular; los servicios web, mediante la adaptación de los ataques comunes que se realizan a las aplicaciones web, se pueden realizar varias acciones que comprometen la seguridad e integridad de los datos que son manejados por estas entidades.

En este caso particular; los servicios web son susceptibles a los ataques comunes que se realizan a las aplicaciones web genéricas mediante ligeras adaptaciones a tales amenazas comprometiéndose con ello la seguridad e integridad de los datos transaccionados por estas entidades. Ya que, como se mencionó anteriormente, hay una tendencia a validar al usuario que accede al servicio y no al servicio como entidad lo cual puede repercutir en la aceptación de solicitud de entidades no autorizadas [11].

El desarrollo de este trabajo de investigación y su futura implementación podría ayudar a disminuir la vulnerabilidad de los servicios web ante posibles ataques de suplantación de identidades. Esto contribuiría a salvaguardar información delicada que es intercambiada entre servicios web que forman parte de un proceso. Además de ahorrar tiempo a empresas e instituciones que coordinan dichos servicios.

1.4 Tipo y descripción general de la investigación

Para la realización de esta investigación se tomara el proceso de trabajo de una investigación descriptiva exploratoria como se muestra a continuación:

- **Investigación Descriptiva**

Este tipo de investigación se ocupa de analizar como es y cómo se manifiesta un fenómeno y sus componentes. Permite detallar el fenómeno estudiando básicamente a través de la medición de uno o más de sus atributos [12].

- **Investigación Experimental**

Busca especificar propiedades, rasgos y características importantes de cualquier fenómeno que se analice [12].

1.5 Objetivo General y Específico de la Investigación

1.5.1 Objetivo General

Desarrollar un modelo para la autenticación de servicios web que implemente una correcta autenticación entre estos, permitiendo establecer una identidad a cada uno de ellos.

1.5.2 Objetivo Especifico

- Especificar un modelo de seguridad para Servicios Web que autentifique a estos como entidades.
- Incorporar en el modelo una flexibilidad en el uso de credenciales.
- Facultar al modelo desarrollado para proveer a los servicios web involucrados, una identidad independiente de la del usuario que haga uso de él.
- Probar la autenticación de entidades a través del modelo desarrollado

1.6 Preguntas de Investigación

Se plantearon las siguientes preguntas de investigación

- **Pregunta General:**
  - ¿Cómo especificar un esquema de seguridad para la autenticación de servicios web como entidad?
- **Pregunta Específica**
  - ¿Qué elementos deben de ser parte de un esquema de seguridad basado en la autenticación de servicios web?
  - ¿Cómo asignar una identidad única a un servicio web?

1.7 Premisas de la investigación

Las premisas de las cuales se parten para la realización de la investigación son las siguientes:

- La autenticación entre servicios web como entidades no ha sido bien desarrollada, para su implementación en un escenario donde interactúen varios servicios web, lo que representa una buena oportunidad de investigación.
- Los estándares actualmente disponibles para establecer relaciones de confianza entre servicios web son muy abstractos [11].

1.8 Área de pertenencia

El problema abordado en esta investigación, pertenece al área de Sistemas de Información en Ingeniería de Software.

1.9 Límites de la investigación

Uno de los límites que tendrá esta investigación será que no evitara el robo de las credenciales del usuario ya sea físicamente (e.g. mediante ingeniería social), o de manera electrónica, para este último existe una gran cantidad de software especializado dirigido a mitigar dicha amenaza (e.g. antivirus, herramientas *antipishing* entre otros). En cuanto al robo físico de credenciales, no se puede establecer un método de protección infalible ya que siempre estará a condición de la actitud que tome el usuario al respecto.

Otro aspecto importante radica en el hecho de que esta propuesta se concentra en la implementación de la seguridad a nivel de mensaje, no a nivel de datos, esto con la finalidad de hacer el modelo más robusto respecto al uso de diferentes tecnologías.

CAPÍTULO II:

2 MARCO TEORICO

Durante el desarrollo de este capítulo se expondrán las diferentes teorías base a las que pertenece el trabajo de investigación, se explicaran brevemente según el ámbito en el cual sean útiles para los resultados a los cuales se quiere llegar, posteriormente se analizaran los diferentes trabajos relacionados con la investigación, una vez descritos se obtendrán las fortalezas y debilidades que estos exponen, las cuales finalmente serán comparadas entre sí, lo cual ayudara a el desarrollo del modelo, al tomar todas las fortalezas que estos exponen y tratar de mitigar las debilidades que tienen.

2.1 Descripción de Teorías Base

Teorías base: arquitecturas orientadas a servicios, seguridad de servicios web, pruebas de software

Se hará una breve descripción de las teorías base que se utilizaran para el desarrollo de esta investigación, y la aportación que hacen a la misma:

Teoría	Que aporta esta teoría
Arquitectura orientada a Servicios	Provee la teoría base para el desarrollo de servicios web
Seguridad de aplicaciones web	Provee la teoría sobre la seguridad y ataques a los que están expuestos los servicios web así como la forma de evitarlos, y las bases para el desarrollo de un modelo seguro de autenticación.
Diseño de Patrones de software	Provee las bases para el desarrollo de patrones, a fin de dar una solución genérica a un problema recurrente dentro de la seguridad en servicios web.
Pruebas de software	Provee el sustento, para la realización de pruebas y verificación del modelo a fin de que pueda ser considerado seguro y confiable.

Tabla 1: Teorías Base

2.2 Arquitectura Orientada a Servicios

2.2.1 Arquitectura de software

La arquitectura de software [13] es la suma de los módulos no triviales, procesos y datos del sistema, su estructura y relaciones exactas de unos con otros, como estos pueden ser y se espera que sean extendidos y modificados, de cuales tecnologías ellos dependen, de la cual se pueden deducir sus capacidades exactas y flexibilidades del sistema, y en qué forma se planea la implementación o modificación del sistema. La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema.

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. Esta se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de cómputo, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

2.3 Arquitectura Orientada a Servicios y Servicios Web

2.3.1 Que son los servicios web

Existen varias definiciones respecto a lo que son los servicios web, una definición mencionada por Tartanoglu [14] establece que son una entidad de software desplegada en la Web cuyas interfaces están descritas en XML. Estos pueden ser implementados usando cualquier lenguaje de programación y ser ejecutados sobre plataformas heterogéneas. Lo cual permite a los Servicios Web la propiedad de interoperar entre diferentes entidades a través del intercambio de mensajes.

Otra definición mencionada por Curbera [15] establece que un servicio web es una aplicación en red que permite interactuar usando protocolos web estándar de aplicación a aplicación sobre interfaces bien definidas, las cuales son descritas usando un lenguaje estándar de definición funcional.

Por lo cual se puede concluir que un Servicio Web es una aplicación que esta publicada en la red, a la cual se puede acceder mediante las interfaces que expone las cuales están descritas en XML. Los Servicios Web se pueden utilizar a modo de componentes, esto para lograr orquestaciones y agregaciones para la creación de complejos procesos.

Los Servicios Web pueden utilizar varios protocolos como medio de transporte. El protocolo más usado es el HTTP, mas no es el único.

Los servicios web [1] pretenden ser bloques de construcción que colectivamente representan un entorno de aplicación, a diferencia de los componentes tradicionales, aunque, estos tienen un número de características que les permiten participar como parte de una arquitectura orientada a servicios. Una de estas características es la autonomía respecto de otros servicios, es decir cada servicio es responsable de su propio dominio, lo cual típicamente se traslada hacia limitar su alcance a una función específica de negocios. Este enfoque de diseño resulta en la creación de unidades funcionales aisladas que están unidas por un común acuerdo a un framework estándar de comunicaciones.

2.3.2 Alcances de los servicios web

Los Servicios web se desarrollan bajo diversas tecnologías como lo son Java, .Net o PHP, las cuales están apegadas a un estándar por lo cual tienen la capacidad de interactuar y cooperar entre sí para lograr un objetivo en común pese a sus diferencias en cuanto a diseño, lenguaje nativo o plataforma sobre la cual se ejecuta, algunos ejemplos a mencionar sobre el uso de servicios web destacan la compra de artículos por internet, ejecución de transacciones bancarias, invocar motores de búsqueda, ligar aplicaciones de empresas, reserva de boletos para avión, u hoteles, la creación de widgtes, el computo científico, redes sociales y más recientemente como interfaces para acceso a la “cloud computing” (computo en nube).

El computo en nube permite a los recursos de infraestructura de TI y a las plataformas de aplicación ser expuestas y consumidas como servicios [16]. Esta es el progreso natural de una arquitectura orientada a servicios. Este progreso es debido a que las capacidades de computo (hardware y software), son expuestas como servicios. La web como plataforma: datos con Web 2.0, programando y desarrollando con mashups, y desplegando la provisión de recursos con cómputo en nube. El éxito de las arquitecturas orientadas a servicios es aparentemente debido a la proliferación de servicios 2.0 que exponen sus APIS. La “infraestructura como servicio” provee recursos de computo (CPU, memoria, almacenamiento y ancho de banda) en demanda vía web services o APIs. Usando tecnologías de virtualización de hardware, los proveedores de nube son capaces de afrontar un rápido aprovisionamiento de recursos [17].

2.3.3 Limitaciones de los servicios web

Los servicios web como cualquier entidad de software cuentan con varias limitantes, las cuales conforme pasa el tiempo y avanza la tecnología se van subsanando, algunas por su misma naturaleza permanecerán inamovibles. Estas son:

- Los servicios web actualmente no tienen estandarizado el almacenamiento de estado, es decir tal como reciben datos, los transforman y los emiten a quien lo solicito, pero no mantienen

datos sobre el proceso previamente realizado, la forma en la que lo mantienen es enviando entre mensajes los estados de los datos que transformaron.

- Como cualquier aplicación basada en web, su disponibilidad está condicionada a la disponibilidad del servidor en el cual se encuentran alojado.
- Su rendimiento es más bajo respecto a otras tecnologías como lo es RMI.
- Debido a que el principal medio de transporte usado en servicios web es el HTTP, se pueden eludir ciertas políticas de ruteo y de firewall, lo cual podría ocasionar ciertos problemas de seguridad [16].
- Al momento de buscar un servicio web mediante UDDI al no estar todos los servicios unificados en uno solo, los clientes no pueden localizarlos de manera eficiente y los proveedores de servicios deben de hacer esfuerzos extra para poder publicar sus servicios de tal forma que sean usados [17]. Incluso algunos motores de búsqueda de internet solo almacenan el archivo WSDL del servicio web pero no publican o muestran como se debería.

2.3.4 Ventajas de los servicios web

Dentro de las ventajas con las que cuentan los servicios web se encuentra:

- **Utilizan estándares abiertos:** al usar tecnologías abiertas, no están ligadas a ningún fabricante como lo es DCOM con Microsoft y RMI con Java.
- **Permiten integración "Just in Time":** para utilizarlos solo se debe de hacer referencia a ellos.
- **Dan vida a aplicaciones legadas:** aplicaciones creadas en lenguajes antiguos como lo es COBOL, solo necesitan ser "envueltas" en XML y exponerse como servicio web para que puedan seguir siendo usadas.
- **Son reutilizables:** pueden ser utilizados como componente estándares que es están dispersos a través de la red, por lo cual pueden ser reutilizados por varias partes en la red.
- **Se pueden utilizar en diversos entornos de hardware / software:** estos están desarrollados en varias tecnologías como lo es Java, tecnologías .Net o PHP y correr sobre sistemas operativos como lo es Microsoft y UNIX por mencionar algunos.

2.4 Arquitectura SOA

La arquitectura SOA [18] (Service Oriented Architecture) es una arquitectura para sistemas distribuidos, esta define como componentes de software a los servicios, estos son organizados en estructuras para soportar requerimientos de negocios. SOA y los servicios web permiten a las organizaciones la automatización de procesos de negocios al incrementar la velocidad y efectividad del intercambio de información. Esta arquitectura viene a dejar atrás las arquitecturas anteriores para sistemas distribuidos como lo es CORBA y DCOM las cuales son propietarias y se limitan solo a su entorno empresarial. Esta arquitectura está caracterizada por las siguientes propiedades:

- **Vista lógica**: el servicio es una abstracción (vista lógica) de los programas, bases de datos, procesos de negocios etc., definido en términos de lo que hace (llevando a cabo una operación de negocio).
- **Orientación a Mensajes:** El servicio se define en términos de los mensajes intercambiados entre agentes proveedores y solicitantes, y no está basado en las propiedades de los agentes. La estructura interna del agente (lenguaje de bases de programación, BD, proceso) se abstrae en SOA. Esto permite incorporar cualquier componente o aplicación a esta arquitectura "decorando" estos componentes con software de gestión y conversión.
- **Orientación a la descripción:** Un servicio se describe con metadatos procesables. La descripción da soporte a la naturaleza pública de SOA: solo se incluyen en la descripción aquellos detalles que se exponen públicamente y son importantes para el uso del servicio. La semántica de un servicio debe documentarse, directamente o indirectamente, por su descripción.
- **Granularidad:** Los servicios tienden a usar un pequeño número de operaciones con mensajes relativamente complejos.
- **Orientación a la red**: Los servicios tienden a usarse a través de la red, aunque este no es un requisito absoluto.
- **Neutral a la plataforma**: Los mensajes se envían en un formato estándar y neutral a la plataforma, distribuido a través de interfaces (XML).

SOA y los Servicios Web son apropiados para aplicaciones [14]:

- Que deben de operar a través de internet, donde la fiabilidad y la velocidad no se pueden garantizar.
- Donde no existe habilidad de gestionar la instalación de forma que todos los solicitantes (clientes) y proveedores se actualicen a la vez.
- Donde los componentes de un sistema distribuido se ejecuten en distintas plataformas y distintos productos.
- Donde una aplicación existente necesite exponerse para ser usada a través de la red y pueda "decorarse" como un Servicio Web.

La arquitectura de un servicio web se basa en la separación de funciones en base al modelo de componentes, en la cual agrega la capa de integración la cual permite un único punto de conexión esto mediante las interfaces que se exponen en XML lo cual permite la implementación de varios bloques de servicios web adecuando únicamente las interfaz mediante la cual se comunica con otro servicio web lo que permite una baja dependencia uno de otro.

2.4.1 Componentes SOA

Los servicios web generalmente trabajan de la mano con los siguientes 3 componentes que a continuación se mencionan:

SOAP

Es un protocolo de mensajes para solicitar alguna operación provista por el servicio, este es un protocolo que está basado en XML el cual es usado, para codificar los mensajes, respuestas y errores de las peticiones a los servicios web que han sido enviados a través de la red, estos mensajes al estar basados en XML son independientes de los Sistemas Operativos, además pueden ser transportados a través de una gran variedad de protocolos de internet como lo es HTTP y SMTP.

Los Servicios web son en gran parte usados por el hecho que su medio de transporte más usado es el HTTP el cual trabaja sobre el puerto 80, lo cual permite pasar los firewall de las empresas en las cuales por lo general se cierran todos los puertos excepto este ya que es el puerto estándar para conexión a Internet al ser el puerto de HTTP.

WSDL

El IDL para Servicios web es WSDL, este es un documento de definición de tipos XML que controla un conjunto de documentos XML, este documento actúa como la especificación de un Web Service [19]. WSDL es un lenguaje XML usado para describir la interfaz de programación de servicios Web [15]. Esta descripción puede ser vista como el contrato entre el proveedor del servicio y el cliente, mediante el cual el proveedor del servicio le especifica al cliente que métodos puede innovar, el tipo de dato esperado para cada método así como también como se accederá a los servicios.

UDDI

UDDI es un directorio distribuido de Servicios Web, el cual permite a los proveedores de servicios web dar a conocer los servicios web que ellos implementan esto de una forma estándar de tal forma que los clientes puedan invocarlos y consultarlos [20]. El UDDI normalmente es conocido como la "sección amarilla", en la cual se puede consultar los servicios web disponibles.

Un registro UDDI es un archivo XML que describe una empresa y los servicios que ofrece. Dicho registro consta de tres partes. Las páginas blancas describen a la empresa que ofrece el servicio: nombre, dirección, contactos, etc. El UDDI es una pieza importante en el área de los servicios Web, sin él, las organizaciones que tienen servicios Web no podrían dar a conocer fácilmente a sus posibles clientes que es lo que ofrecen, y los clientes no podrían conocer los datos que necesitan para acceder a esos servicios [21].

2.4.2 Roles de un servicio web

Los servicios pueden asumir diferentes roles cuando interactúan en varios escenarios. Dependiendo del contexto en el cual sean vistos, será la tarea que esté realizando en el momento, el servicio web tomara diferentes roles al mismo tiempo [1].

Proveedor de Servicios

Cuando actúa como un proveedor de servicios, el servicio web expone una interface a través de la cual los solicitantes pueden invocar el servicio. Un proveedor de servicios promueve su interface para publicar la descripción de un servicio. En un modelo cliente-servidor, el proveedor seria comparable con el servidor. Un proveedor también puede actuar como solicitante del servicio.

Solicitante de Servicio

Un solicitante de servicio es el servicio web o programa que envía un mensaje de solicitud a un servicio web especifico. En el modelo cliente .servidor este tomaría el papel del cliente. Un servicio web también podría actuar como proveedor del servicio.

Intermediario

El rol de intermediario es asumido por el servicio web cuando este recibe un mensaje del solicitante del servicio, y entonces este envía el mensaje al proveedor del servicio. Este también puede actuar como proveedor y solicitante. El intermediario puede existir en diferentes formas, pueden ser pasivos y simplemente pueden distribuir o enrutar mensajes, mientras que en estado activo procesa el mensaje antes de volver a enviarlo. Típicamente, a los intermediarios solo se les permite procesar o modificar el encabezado del mensaje.

2.5 Seguridad de aplicaciones web

En un principio los requerimientos tradicionales de un sistema seguro son la integridad, confidencialidad y disponibilidad, cualquier acción que se enfoque a violar cualquiera de estos conceptos es llamada ataque, mientras que la posibilidad de que ocurra un ataque es llamado vulnerabilidad [22]. Dentro del contexto de seguridad referente a SOA existen varios aspectos los cuales se pueden dividir en dos grupos [23]:

2.5.1 Aspectos funcionales de seguridad

Estos aspectos son estándar en el sentido de que ellos incluso aparecen en las aplicaciones tradicionales:

- **Autenticación:** verifica que los solicitantes (usuarios humanos, entidades de sistemas registrados y componentes) son quien dicen ser. El resultado de la autenticación es el establecimiento de credenciales, las cuales describen los atributos (por ejemplo, identidad, rol, grupo, espacio) que puede estar asociado con la autenticación del solicitante.
- **Autorización:** Brinda permiso a los solicitantes autorizados para accesar a recursos, previendo las bases para el control de acceso, el cual aplica restricciones para prevenir el acceso sin autorización a recursos. El control de acceso asegura que solo los demandantes autorizados puedan accesar y modificar los recursos.
- **Criptografía:** provee algoritmos criptográficos y protocolos para proteger datos y mensajes de ser leídos o alterados. La encriptación provee la confidencialidad para codificar datos en una forma inteligible mediante un algoritmo reversible, el cual permite mantener una llave de desencriptacion para decodificar los datos. Una firma digital provee la integridad al aplicarse,

esto para asegurar que los datos son auténticos y no han sido modificados durante el almacenaje o transmisión.

- **Confidencialidad de datos:** Proteger el grado de ocultación de datos sensibles.
- **Integridad de los datos:** Detectar la manipulación de datos y asegurarse que únicamente el remitente de los datos pueda obtener acceso a ellos de forma íntegra.
- **Protección contra ataques:** asegurarse que hackers no tomen el control de las aplicaciones.

2.5.2 Aspectos no funcionales de seguridad

Estos aspectos son no funcionales en el sentido de que no están directamente relacionados con la seguridad:

- **Interoperabilidad:** Este se refiere directamente SOA, donde diferentes soluciones de seguridad no deben de romper la compatibilidad entre los servicios que de otra forma serian compatibles.
- **Gestión:** Esta preocupación es la más grande para SOA, en la cual una única solución de seguridad debe de proteger a diferentes servicios.
- **Facilidad de desarrollo:** Esta inquietud es común para cualquier solución de seguridad, bien en SOA o en el desarrollo de una aplicación tradicional, la complejidad reduce la adopción de cualquier solución de seguridad.

2.5.3 Autenticación

Las opciones de autenticación en servicios web están divididas en dos categorías: orientadas a la conexión y orientadas al documento.

Autenticación Orientada a la Conexión

Los sistemas con autenticación orientados a la conexión identifican quien o que esta el otro punto final de la conexión. Incluso donde el protocolo de comunicación una conexión prolongada, algunos sistemas orientados a la conexión, mantiene el estado de sesión usando cookies o extensiones URL en las que los usuarios que no han sido autenticados no pueden hacer ninguna petición al server.

Todos los sistemas de autenticación orientados a la conexión están construidos sobre un pasword o bien mediante reto-respuesta. Las categorías de cada sistema de autenticación son las siguientes:

- **Autenticación basada en sistema operativo:** depende de mecanismos subyacentes que son soportados por el sistema operativo, como lo es Microsoft Windows.
- **Autenticación basada en servidores web:** usa las capacidades que están disponibles en HTTP, como lo son las contraseñas.
- **Autenticación basada en tokens:** requiere que el usuario posea un token físico, como lo es una smartcard, la cual juega un papel en el proceso de autenticación, o bien un archivo llave como es el caso de Lotus Notes. Algunas veces el token muestra un valor que debe de ser verificado por un servidor de autenticación, el token en otras ocasiones puede tener una interfaz eléctrica.

- **Autenticación Web SSO:** soporta diferentes métodos de autenticación y, en adición, mantiene una sesión autenticada que puede ser usada para accesar a un grupo de servicios web dentro del dominio.

La autenticación SSO cliente/servidor define la autenticación de la sesión basada en un protocolo de autenticación criptográfico como lo es Kerberos.

La autenticación SSO al momento cuenta con dos grandes propuestas: la .Net Passport de Microsoft y la SAML la cual es desarrollada por OASIS [16].

Autenticación Orientada a Documentos

Los sistemas con autenticación orientada a documentos, adjuntan o incrustan un token de autenticación o tokens con un mensaje. Los mensajes y sus tokens de autenticación pueden ser transportados usando varios tipos de protocolos entre los cuales se encuentra HTTP, SMTP y FTP. El significado de la información de autenticación contenida en los tokens debe de ser negociada por el remitente y el receptor del mensaje.

Los sistemas basados en autenticación orientada a documentos incrustan información alrededor de una entidad dentro del cuerpo del documento. Esta información permite al destinatario autenticar el creador del documento o alguna otra parte confiable que de fe de la identidad de la entidad que está relacionada con el documento. La relación exacta de la entidad en el documento puede variar y debe de ser agregada previamente, existen dos enfoques principales [16]:

Firmas Digitales

Con las firmas digitales, una o más partes firman todo el documento o partes del mensaje usando un algoritmo de firma digital como lo es RSA o DSA. Para servicios web la opción ideal es XML Signature y WS-Security. Estos están adaptados a SOAP, están basados en documentos XML y tiene la flexibilidad que es necesaria para el uso de mensajes SOAP.

Otra opción para firmar documentos de servicios web basados en XML es la sintaxis de mensajes criptográficos (Cryptographic Message Sintax CMS). CMS es usado con Secure Multipart Internet Message Extension (S/MIME). CSM y S/MIME fueron desarrollados por la IETFS como una forma de proteger el e-mail. Estos asumen que las entradas son texto pero no XML. También incluyen estándar para la firma y desencriptación de mensajes basados en texto así como también se incluye una especificación para codificar una firma y certificados que pueden ser manejados por sistemas de e-mail.

Tokens

Esta consiste en insertar un token a el encabezado, tal como lo hace SAML [16] al agregar SAML Assertions, las cuales están específicamente diseñadas para llevar información de seguridad relevante. Otra opción es WS-Security la cual está adaptada a SOAP y reconoce diversos tipos de tokens como lo son Passaport, Kerberos y X.509.

Protección de datos

La protección de datos se enfoca en proteger los datos de espías que los podrían ver si autorización así como de la alteración de los mismos. Los servicios web deben de ser protegidos de mensajes inapropiados.

Para proteger los datos se deben de proveer dos servicios de seguridad: confidencialidad e integridad. Se debe de asegurar que el contenido del mensaje no será revelado a personal que no está autorizado así como también el mensaje no debe de ser alterado. Para estos casos es usada la criptografía. Para lo cual se pueden tener dos enfoques: orientados a la conexión u orientados al mensaje. El enfoque orientado a la conexión protege los mensajes mientras estos son transmitidos entre sistemas y las soluciones a este enfoque incluyen SSL e IPSec. Los orientados a mensaje protegen los datos en tránsito o en almacenamiento, este enfoque incluye soluciones como los es XML Encryption y S/MIME.

Un mecanismo de seguridad puede proveer más de un servicio. Por ejemplo una forma digital puede autenticar el origen de un mensaje pero también puede proteger la integridad del mensaje. SSL e IPSEc son usados para transmitir de forma segura datos de un punto a otro de forma conjunta. Con los servicios web, el mensaje podría pasar por manos de varios servicios web los cuales quizás solo lo retransmitan o solo enruten motivo por el cual deben de leer su contenido, siendo esto una desventaja puesto que si el mensaje contiene datos que no deben de ser leídos, crearía un agujero de seguridad, motivo por el cual para datos de suma importancia SSL e IPSEc son la opción adecuada, aunque en cada punto de enrutamiento se debe de negociar una nueva sesión.

XML Signature y XML Encryption trabajan juntos para proveer protección de datos orientados a mensaje. Una ventaja de XML Encryption es que porciones del mensaje SOAP pueden ser encriptadas. Las porciones encriptadas del mensaje podrían o no coincidir las partes firmadas del documento. Esto implica que se puede encriptar por partes, algunas de las cuales solo pueden ser vistas por ciertas personas con los permisos necesarios.

2.5.4 Criptografía

La criptografía es la ciencia de aplicar matemáticas complejas para aumentar la seguridad de las transacciones electrónicas. Esta se basa en el concepto de que algunos cálculos pueden ser fáciles en un sentido, pero extremadamente difíciles en el sentido contrario [24]. Los algoritmos criptográficos son algoritmos matemáticos que están diseñados de tal manera que se pueden llamar con diferentes conjuntos de datos para entrar en funcionamiento

Criptografía simétrica

Los algoritmos criptográficos simétricos toman el texto claro como entrada. Después usando una clave simétrica, sacan una versión cifrada del texto, también llamada texto cifrado. Los algoritmos simétricos tienen dos versiones: cifrador en bloque y cifrador de flujo. Dentro de los algoritmos existentes se encuentra: DES, 3-DES, RC2, RC5, RC6 y Rijndael.

Por lo cual tiene las siguientes características:

- En la criptografía simétrica, se utiliza la misma clave para cifrar y descifrar.
- El cifrado simétrico es rápido.
- El cifrado simétrico es seguro.
- El texto cifrado que resulta de un cifrado simétrico es compacto.
- Dado que la clave simétrica debe de llegar al receptor, el cifrado simétrico está sujeto a la intercepción.
- El número de claves en la criptografía simétrica es, aproximadamente el cuadrado del número de participantes y, por tanto, no tiene una buna escalabilidad en poblaciones muy numerosas.
- La criptografía simétrica requiere de una administración compleja de claves.
- La criptografía simétrica no se ajusta a las firmas digitales o a la aceptación.

Criptografía asimétrica

Dentro de la criptografía asimétrica se requiere una longitud de clave mucho más grande, para lograr el mismo nivel de seguridad que usando una clave simétrica. Los algoritmos asimétricos se llaman de esta forma, porque en lugar de usar una sola clave para realizar la codificación y decodificación, se utilizan dos claves diferentes: una para cifrar y otra para descifrar. Estas dos claves independientes, pero matemáticamente relacionadas, siempre se generan juntas

Dentro de los algoritmos existentes se encuentra el RSA

Entre las principales características de la criptografía asimétrica se encuentra:

- Con la criptografía asimétrica está cifrada con una clave (pública o privada) solo se puede descifrar con la otra clave (privada o pública).
- El cifrado asimétrico es seguro.
- Dado que no se necesita enviar la clave al receptor, la codificación asimétrica no sufre por la interceptación de claves.
- El número de claves que se necesita distribuir es el mismo de participantes, de ahí que la criptografía asimétrica sirve muy en escalas de poblaciones grandes.
- La criptografía asimétrica no tiene los problemas complejos de distribución de claves
- La criptografía asimétrica no exige una relación previa entre las partes para hacer el intercambio de claves.
- La criptografía asimétrica soporta firmas digitales y aceptación.
- El cifrado asimétrico es relativamente lento.
- El cifrado asimétrico expande el texto cifrado.

2.6 Pruebas de software

Dentro de los servicios web existen numerosas pruebas a realizar según el tipo de aplicación que se esté desarrollando en el caso especifico de los servicios web se pueden aplicar las siguientes, las cuales están basadas en las pruebas que se aplican comúnmente a aplicaciones web:

En la actualidad existe un proyecto llamado OWASP (Open Web Aplication Security Project) [24], la cual es un proyecto abierto el cual está enfocado, en mejorar la seguridad de las aplicaciones web. Dentro de este proyecto existe un apartado llamado Testing Guide, donde se definen varias pruebas que se pueden realizar a las aplicaciones web, entre la cuales también se pueden incluir a los servicios web:

2.6.1 Injection

Los ataques de inyección involucran la subversión de un subsistema para desarrollar acciones que benefician al atacante. Pasando directamente parámetros de entrada a un query del subsistema u otro constructo que tenga una estructura en especial, los caracteres especiales pueden ser usados para realizar cambios. Esto al alterar los parámetros que se envían hacia la aplicación y alterar el comportamiento que el sistema debería de tener. Dentro de las vulnerabilidades más comunes están las siguientes:

- SQL injection
- Command Injection
- LDAP injection
- XPtah Injection
- Code Injection
- XML Injection

2.6.2 Confidencialidad e Integridad

La confidencialidad provee la habilidad para que las comunicaciones entre cliente-intermediario-servidor se den sin que partes no autorizadas puedan estarlas “escuchando”. La integridad provee la habilidad para que las comunicaciones sean protegidas contra la falsificación o alteraciones de partes no autorizadas. Estas dos características son críticas en un entorno de servicios web puesto que sus comunicaciones viajan a través de una red poco segura.

Dentro de las vulnerabilidades y pruebas más comunes a realizarse en aplicaciones web son las siguientes:

- Cipher choise
- Encryption coverage
- Ataques de repetición
- Integrity check coverage
- Invalid XML
- XML canonización
- Algoritmos no soportados
- Falla ante los requerimientos de las políticas

2.6.3 Autenticación

La autenticación es típicamente provista por un nombre de usuario y su contraseña, dependiendo de los requerimientos de las aplicaciones, se pueden agregar múltiples factores de autenticación como lo serian: tarjetas inteligentes, tokens, o elementos biométricos que también pueden ser usados para asegurar la identidad. Dentro de los típicos casos de prueba se incluyen la falsificación o modificación de credenciales, así como la perdida de estas o un uso erróneo de la misma.

Un aspecto más de la autenticación consiste en la captura de una comunicación que involucre el intercambio de credenciales e intentar extraer las credenciales de este para poder accesar a la aplicación. Al realizarlo se puede repetir una petición hacia la aplicación o bien acceder como un usuario ilegitimo. Dentro de las vulnerabilidades y pruebas más usadas se encuentran las siguientes:

- Ataque de fuerza bruta
- Credenciales perdidas
- Credenciales falsificadas
- Ataques de repetición
- Manipulación de la autenticación
- Hombre en medio
- Manipulación de la sesión de autenticación
- Almacenamiento de las credenciales de autenticación
- Confidencialidad en el intercambio de autenticación
- Verificación de certificados

2.6.4 Autorización

La autorización y control de acceso va de la mano con la autenticación, esta es muy comúnmente implementada en listas de control de acceso y roles. Esta debe de ser medida para asegurar el correcto acceso, o bien si un rol es asignado a un usuario autenticado y que este no pueda ser falsificado para acceder a información desautorizada o alguna funcionalidad especifica.

Dentro de las pruebas más usuales se encuentran las siguientes:

- Token falsificado
- Ataques hijacking
- Manipulación entre limites de confianza
- Ataques en filtración de IP
- Archivo temporales

2.6.5 Disponibilidad

Esta implica determinar la disponibilidad del servicio o aplicación ante diversos ataques que este podría sufrir, básicamente se basa en tratar de saturar o desactivar la aplicación de tal forma que no pueda seguir procesando peticiones y deje de funcionar. Como lo muestran las siguientes pruebas:

- SOAP demasiado grande
- Referencia a entidades
- Envenenamiento de esquema
- Desvíos de ruteado
- Inundación de autenticación

2.6.6 Pruebas Estándar

Las siguientes son pruebas estándar que se realizan sobre servicios web, en las cuales se enfocan en tratar de obtener información acerca el servicio web y por consecuencia encontrar vulnerabilidades dentro de estos de una forma más sencilla, en lugar de estar a prueba y error.

- WS Information Gathering
- Testing WSDL
- XML structural testing
- XML content level testing

2.7 Patrones de Diseño

Un patrón de diseño como lo menciona [25] "describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución al problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces". En general un patrón tiene 4 elementos esenciales:

- Nombre del patrón: permite describir, en una o dos palabras, un problema de diseño junto con sus soluciones y consecuencias.
- Problema: describe cuando aplicar el patrón.
- Solución: describe los elementos que constituyen el diseño, sus relaciones, responsabilidades y colaboraciones.
- Consecuencias: son los resultados así como las ventajas e inconvenientes de aplicar el patrón

Los patrones de seguridad expuestos por Gamma [25], se clasifican según su propósito como lo es:

- **De Creación:** Factory Method, Abstract Factory, Builder, Prototype, Singleton
- **Estructurales:** Adapter, Bridge, Composite, decorator, Façade, Flyweight, Proxy
- **De Comportamiento:** Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor.

Un patrón de seguridad como lo menciona Kienzle [26] es una bien entendida solución a un problema recurrente de información.

Existen dos grandes categorías sobre patrones de seguridad:

Patrones estructurales: Estos patrones pueden ser implementados en el producto final, estos abarcan los descritos por Gamma [25]. A menudo incluyen diagramas de estructura y descripciones de interacción.

Patrones de procedimiento: estos patrones pueden ser usados para mejorar el proceso de desarrollo de software en donde la seguridad es un objetivo crítico, estos a menudo impactan en la organización o la administración del proyecto de software.

Dentro de los patrones orientados a la seguridad se encuentran los desarrollados por Kienzle [26], los cuales se mencionan a continuación:

Patrones Estructurales:

- Account LockOut
- Authenticated Session
- Client Data Storage
- Client Input Filters
- Directed Session
- Encrypted Storage
- Hidden Implementation

Estos solo por mostrar algunos de los que están disponibles

Patrones de procedimiento:

- Build The Server from The Ground Up
- Choose The Right Stuff
- Document The Security Goals
- Document The Server Configuration
- Enroll By Validating Out Of Band

2.8 Trabajos relacionados

A continuación se describen los trabajos relacionados con esta investigación, se describirán sus fortalezas y debilidades y al final se hará una comparación entre ellos.

2.8.1 Client Certificate and IP Address Based Multi-factor Authentication for J2EE Web Applications

Este trabajo [27] está basado en la autenticación de sesiones basadas en navegadores web. Persigue la misma problemática de esta investigación pero aplicada a clientes web basados en un navegador, en el cual su problemática radica en la correcta autenticación del cliente mediante certificados, para posteriormente autenticar al usuario.

Para ello utiliza certificados y cifrado basado en SSL para comunicarse con el servidor de la aplicación a la cual se desea tener acceso todo esto basado en Java. La forma en que lo implementa es de la siguiente manera:

En cada uno de los clientes que deben de tener acceso a la aplicación protegida, se instala un certificado de cliente SSL, el cual provee identidad a cada uno de los puntos de acceso.

En el servidor de la aplicación se cuenta con un Servlet a modo de proxy el cual se encarga de filtrar todas las peticiones que se realicen.

Inicialmente se realiza una autenticación inicial llamada de “dos direcciones”, en la cual el servidor solicita al cliente su certificado para comprobar su identidad, una vez verificada la identidad se genera un token de sesión el cual es enviada al cliente para que proceda a comunicarse de esta forma se crea un “canal seguro”, este token será destruido al final de la sesión.

El cliente realiza una petición con el token recién adquirido, esta petición es interceptada en el servidor por un Servlet residente ahí, el cual verifica el encabezado del mensaje, buscando específicamente la IP de la cual proviene la petición. La IP obtenida es verificada contra una lista de IP’s autorizadas, en caso de que la IP este permitida, le permite acceder a la aplicación, caso contrario regresa una página de error

Presenta las siguientes desventajas:

- Necesariamente se deben de instalar certificados en cada uno de los clientes, los cuales desean acceder la aplicación web protegida, por lo cual la implementación se reduce a entornos de acceso pequeños y medianos.
- Prevee que los certificados proveídos serán robados, pero confía en que la IP usada como segundo medio de autenticación no podrá ser falsificada.
- No contempla el acceso desde atrás de un proxy, por lo cual menciona que una debilidad será que alguien podría utilizar el certificado desde un cliente no autorizado dentro de la misma subred.

- Concentra toda la toma de decisión en un único punto, en un servlet que filtra a modo de proxy todas las peticiones entrantes, de tal forma que para poder modificar los permisos de acceso a las aplicaciones protegidas este es el único punto de modificación.
- Existe una gran carga al crear, instalar y mantener los certificados en cada uno de los clientes.
- No es fácil detectar cuando se hizo acceso con un certificado valido desde otro cliente no autorizado.

Presenta la siguiente ventaja:

- Su uso es muy poco invasiva tanto para el servidor como para los clientes, puesto que en el servidor solo se debe de instalar el serlvet y en los clientes el certificado, proceso que ya se encuentra estandarizado.

2.8.2 Microsoft IIS

Microsoft IIS el cual es un servidor de Servicios Web implementa HTTP Authentication, la cual está basada en IP/DNS. De tal forma que este puede ser configurado para aceptar solicitudes de IP´s y DNS específicos, pero al mismo tiempo debe de hacer un loopback al DNS para comprobar la autenticidad de la llamada, esto repercute en la carga extra del servidor al estar comprobado las peticiones, además en el caso de que un servicio web se encuentre detrás de un proxy no podrá acceder ya que quizás la IP de proxy no está registrada y esta sea rechazada [16].

Otra política de seguridad que implementa IIS, está basada en Windows Security: Windows User Account, la cual limita el acceso a peticiones que provengan de dominios Windows los cuales están configurados por una cuenta Windows, este tiene la limitante de que solo está disponible en entornos Windows, por lo cual pierde la cualidad de heterogeneidad [16].

2.8.3 SOMA: Mutual Approval for Included Content in Web Pages

Este trabajo desarrollado por T. Oda [28] muestra un método para la estructuración de páginas web basadas en la autorización para la publicación de contenido, está basado en 3 componentes principales: Add On para navegador a modo de elemento autenticador, archivos manifiesto para la especificación de permisos de publicación y un script en cada cliente para validar las solicitudes, como a continuación se describe en la Figura 2: Diseño SOMA:

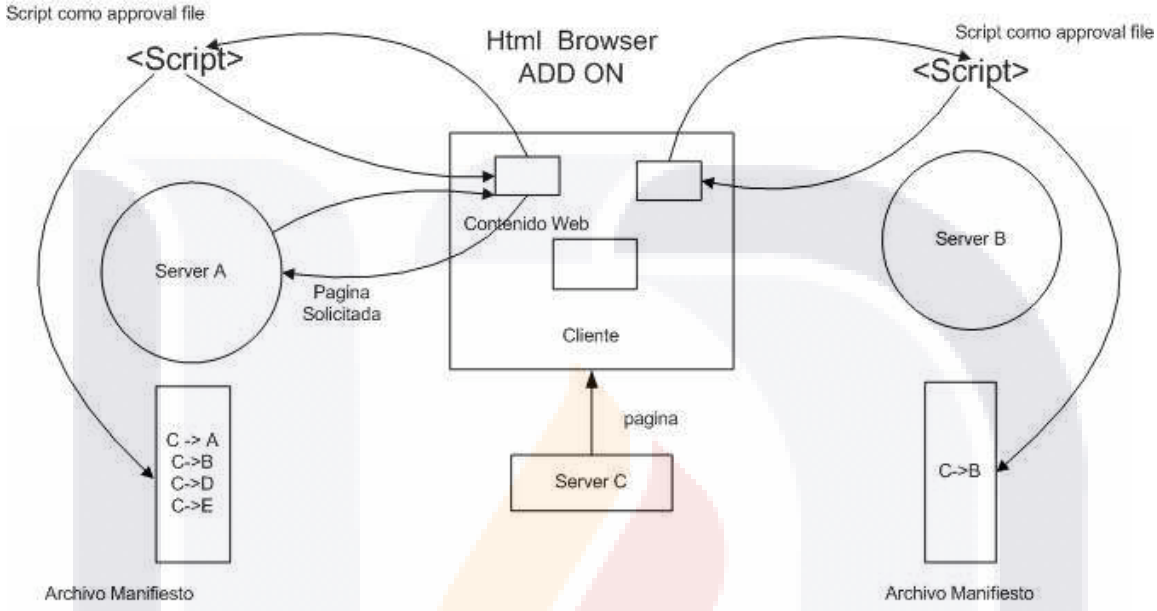


Figura 2: Diseño SOMA

El diseño se basa en la composición de páginas web basadas en permisos otorgados mediante archivos manifiesto, los cuales residen dentro de cada servidor involucrado. Al intentar obtener información desde algún otro servidor, este evalúa mediante un script, si el servidor solicitante esta registrado en su archivo manifiesto, si este está aprobado, entonces le permite acceder a la información que este expone permitiendo estructurar la página web.

2.9 Contribuciones y limitaciones de trabajos revisados

A continuación se describen brevemente las contribuciones y limitaciones de los trabajos realizados:

Titulo de Investigación 1	**SOMA: Mutual Approval for Included Content in Web Pages**
Desarrollado por	T. Oda [28]
Contribución	Permite un establecimiento de permisos entre los clientes, permitiendo evitar una centralización en la toma de decisiones.
Limitante:	Esta desarrollado para aplicaciones web. Se necesita un Add On extra que se instala en cada navegador. No evita ataques como Middle Man ya que no protege la información mientras esta viaja en la red.
Trabajo	**Microsoft IIS**
Desarrollado por	Microsoft [16]
Contribución	Permite establecer una identidad única para cada uno de los involucrados
Limitante	Esta desarrollado únicamente para tecnologías basadas en Microsoft. Agrega carga extra al momento de la autenticación, debido a que debe de hacer una llamada de comprobación a el destino que origino la petición.
Titulo de Investigación 2	**Client Certificate and IP Address Based Multi-factor Authentication for J2EE Web Applications**
Desarrollado por	H. Park y S. Redford [27]
Contribución	Provee un servlet a modo de proxy en el servidor de la aplicación a proteger, el cual selecciona las peticiones autorizadas a acceder a la aplicación web en base a la IP de procedencia.
Limitante	• Esta desarrollado para aplicaciones web. • Depende en gran medida de los certificados, lo cual repercute en un gran esfuerzo en su implementación, desarrollo y mantenimiento. • Concentra toda la toma de decisiones en un punto. • Solo esta ideado para SSL.

Tabla 2: Trabajos Relacionados

Comparación de Trabajos Relacionados

Estrategia de Autenticación / Fortaleza	SOMA [28]	Microsoft IIS [16]	Client Certificate and Ip Address Based Multi-factor Authentication [27]	Propuesta
Seguridad en el medio de transporte	NO	SI	SI	SI
Identidad propia para cada uno de los participantes	NO	SI	NO	SI
Heterogeneidad sobre la tecnología de implantación	SI	NO	NO	SI
Autonomía en la compartición de información	SI	NO	NO	SI

Tabla 3: Comparación de fortalezas y debilidades de trabajos relacionados

En la Tabla 3 previamente mostrada, se observan las fortalezas y debilidades de los diferentes trabajos relacionados en estos cabe destacar que cada uno de ellos presenta una fortaleza en un área específica, dejando otra desprotegida. Las categorías en las cuales entre todos los trabajos relacionados coincidieron son:

- **Seguridad en el medio de transporte:**

Este tópico fue tomado en cuenta debido a la seguridad que forzosamente debe de implementar cualquier medio de transmisión de datos, puesto que bien quizás no alteren los datos, pero pueden copiar la información y obtener la credenciales que en su momento está siendo transmitidas, lo cual implica un gran riesgo para el usuario que hace uso de la aplicación.

- **Identidad propia para cada uno de los participantes**

Esta categoría es una de las bases de este trabajo de investigación, se deberá de asignar a cada uno los participantes una identidad, de tal forma que no solo se haga uso de la identidad que presenta el usuario que hace uso de la aplicación sino que también cada entidad de software tenga una propia.

- **Heterogeneidad sobre la tecnología de implantación**

La heterogeneidad sobre la tecnología de implantación es una de las esencias sobre las cuales los servicios web están basados, el permitir que cualquier modelo o esquema de autenticación que haga uso de servicios web la implemente es base para su mejor aceptación.

- **Autonomía en la compartición de información**

Este tópico puede ser visto desde varios puntos de vista, en el caso de servicios web que sean solo usados en corporaciones grandes, y consumidos en áreas especificas quizás lo usuarios necesitarían que la compartición de de información sea decidida en un único punto, pero en el caso de servicios que son creados por diversas empresas, las cuales son muy diferentes unas de otras,

cada una debería tener el derecho de solo compartir la información que quiere con quien quiere, sin que en un punto intermedio alguien supuestamente lo haga.

CAPÍTULO III:

3 DESARROLLO DE LA INVESTIGACIÓN

Dentro de este capítulo se desarrollara el modelo propuesto, será descrito en cada uno de sus componentes tanto en estructura, como en comportamiento, además se definirán las diversas relaciones existentes entre ellos.

3.1 Introducción

Un servicio web es una aplicación a modo de componente en la cual sus funcionalidades e interfaces están expuestas a potenciales usuarios a través de la web, mediante tecnologías como lo es XML, SOAP, WSDL y HTTP [29]. Estas aplicaciones se caracterizan por ser tecnológicamente independientes entre sí, además de permitir su continuo rehusó y agregación “al vuelo” dentro de una aplicación.

Los requerimientos típicos dentro de un sistema seguro son integridad, confidencialidad, disponibilidad [22] y autenticación. Cualquier acción que se enfoque en la violación de alguna de estas propiedades es llamada “ataque”, la posibilidad de que exista un ataque es llamada “vulnerabilidad”.

La seguridad dentro de los sistemas de software continuamente ha sido vulnerada por diferentes ataques, en este caso considerando aplicaciones web, concretamente servicios web. Dentro de este contexto tomando en consideración el enfoque hacia los servicios web estos pueden ser susceptibles a diversos ataques como lo menciona Jensen et al [22]: oversize payload (sobrecarga dañina) , Coervice Parsing (parseo coercitivo), SOAPAction Spoofing (falsificación de contenido SOAP), XML Injection (Inyección de XML), WSDL Scanning, Metadata Spoofing, ataque de ofuscación y sobrecarga criptográfica solo por mencionar algunos ataques de los que pueden ser victima los servicios web.

La autenticación como lo menciona Hartman [16] es el método mediante el cual una entidad demuestra que es quien dice ser, en un entorno de software, esta es implementada mediante el uso de credenciales las cuales son muy variadas como lo son: passwords, llaves públicas y privadas certificados digitales [30], tokens de acceso, entre otros.

3.2 Problemática

La problemática a resolver dentro de este trabajo de investigación se encuentra dentro del contexto de la autenticación de un servicio web hacia otro, es decir el proceso mediante el cual un servicio web expone su identidad hacia otro servicio web, para poder hacer uso de las funcionalidades que este expone. Por lo general a quien se valida es al usuario que accede al servicio mediante las credenciales que presenta y no al servicio como entidad, lo cual repercute en la aceptación de solicitudes realizadas con credenciales validas por entidades no permitidas [11].

La problemática está basada en la situación en la cual un servicio web expone sus interfaces para que otros puedan consumir la información que este proporciona. Para lo cual este expide credenciales (passwords, certificados, llaves) para que usuarios autorizados puedan accesar a la información.

En ocasiones a estos usuarios les son substraídas sus credenciales, lo cual es reportado y estas inmediatamente revocadas o bien hacen uso indebido de ellas. Pero en situaciones en las que el usuario no lo ha notado, la persona que las substrajo, puede crear un servicio web en el cual, el utilice las credenciales del usuario y pueda realizar peticiones al servicio web que emitió las credenciales. Estas peticiones pueden estar estructuradas de tal forma que no contienen validaciones creadas desde el lado del cliente, que quizás el servidor aun no haya implementado como lo describe Jensen et al [22], lo cual puede repercutir en la integridad del servidor, así como la integridad de la información que expone.

Este tipo de problemas pueden ser atacados mediante el uso de patrones, concretamente security patterns y misuse patterns, los cuales muestran en una forma genérica como provenir y como es que un ataque se lleva a cabo, ejemplo de misuse pattern en el cual se muestra una forma general de cómo atacar el problema puede ser visto en el apartado de anexos

3.3 Acotamiento del problema

El problema será tratado respecto de los siguientes límites que a continuación se exponen:

- Los servicios web no estarán dentro de un entorno federado.
- Los servicios web a tratar no estarán dentro de un mismo entorno de software ni estarán desarrollados bajo un mismo lenguaje de programación.
- No se abarcara el SSO (Simple Sign-On).
- Se toma por cierto que los canales de comunicación son seguros.
- Se toma por cierto que las credenciales del usuario pueden ser robadas.
- No se evitara el robo de las credenciales del usuario, puesto que esto está directamente relacionado con las actitudes que tome el usuario al respecto, así como también ya existen herramientas expresamente desarrolladas para ese fin como lo son los antivirus y diversas herramientas de seguridad disponibles.

3.4 Fundamentos del enfoque

Este enfoque está basado en la poca información existente acerca de cómo proporcionar una identidad a una entidad de software, más concretamente a un servicio web, así como la forma de establecer confianza entre clientes y proveedores, actualmente existe una especificación que describe este problema pero como ya se menciono anteriormente es demasiada abstracta en su implementación.

Este enfoque está basado en el modelo de intermediación de confianza llamado confianza directa intermediada (Direct Brokered Trust) mencionada en WS-Trust [17], debido a la continua comparación de las credenciales contra un listado de entidades aceptadas, además también está basado en la topología de confianza expuesta por WS Federation [31], en la cual describe como un servicio web puede interactuar con los recursos que se desean consumir y el proveedor de tokens.

Al asignar una identidad a una entidad no basta simplemente con asignar un nombre, un identificador, o resaltar una de sus características, también se deben de proveer los medios con los cuales esta entidad puede ser verificada. Proceso que es un poco más complicado en el entorno del software, puesto que por su naturaleza, estos no tienen una característica tangible o bien, se le puede asignar alguna característica, pero con estas, existe la posibilidad de que sea robada y usada por un atacante. Por lo cual en este modelo se plantea dar a un servicio web una identidad, pero a su vez describir los medios y procesos mediante los cuales se pueda verificar su identidad, tratando de que esta no pueda ser robada o falsificada, y en caso de que así fuera, esta debería de ser comprobada como una falsificación por parte del atacante que realice tal acto.

3.4.1 Introducción

Como introducción se establece el contexto sobre el cual se desarrollara el modelo a ser detallado, este está basado en servicios web, dirigiéndose hacia el área de seguridad aplicada a estos, pero la seguridad se puede aplicar a varios aspectos como lo es:

- Integridad
- Autenticación
- Confidencialidad

Este modelo está enfocado al área de autenticación de los servicios web. Dentro de este modelo se asume como verdadero que:

- Las credenciales del usuario han sido robadas.
- Un servidor Autenticador (Intermediario), al ser un servidor dedicado, es seguro.
- No se puede acceder al código de un servicio web, puesto que este se encuentra compilado.

Para lo cual se toma autenticación como lo define Hartman [16] “es el método mediante el cual una entidad demuestra que es quien dice ser”.

Objetivo:

Integrar un mecanismo que autentique la identidad entre Servicios Web

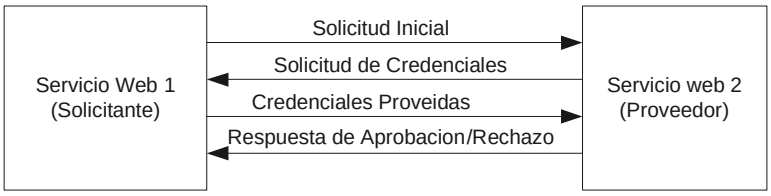


Figura 3: Interacción de autenticación básica entre Servicios Web

En la Figura 3: Interacción de autenticación básica entre Servicios Web, se muestra el proceso básico de autenticación entre servicios web, en ellos se realiza una comunicación, la cual puede ser cliente/servidor en ambas direcciones.

Por lo cual debe de existir un mecanismo el cual permita asegurar a ambas partes que las peticiones que están recibiendo provienen de la entidad que ellos creen, no importando las credenciales de usuario que presente.

Estas peticiones deben de ser evaluadas por una entidad externa, esta puede ser un ente de alguna de las partes, dedicado únicamente a este proceso, o alguna entidad externo, de la cual se tenga una confianza demostrada. Con los avances actuales en robo de información es posible crear peticiones desde otros puntos de la red, en las cuales integren un origen igual al que inicialmente se confía.

Para la realización de este modelo se hace uso un intermediario el cual evalúa las peticiones, este modelo de intermediario está basado en la arquitectura provista por SAML [31], además de ser utilizado por WS-Trust [32] en los diferentes escenarios de relaciones de confianza que expone.

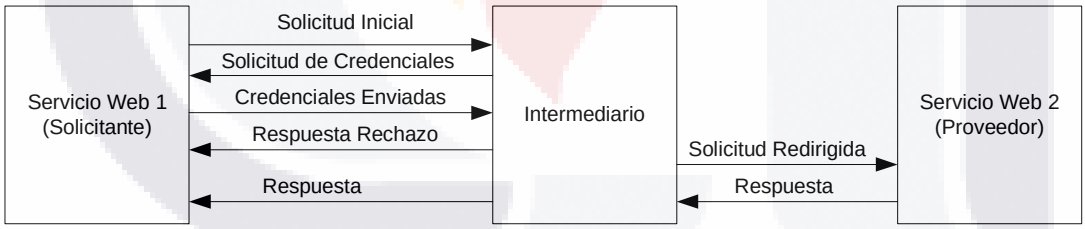


Figura 4: Interacción entre Servicios Web e Intermediario

Como se muestra en la Figura 4 basada en [33] y [32], entre ambos servicios web existe un intermediario, el cual se encarga de evaluar a los servicios web involucrados, emitiendo rechazo a las solicitudes o aprobándolas, lo cual permite la comunicación entre ellos. Al realizarlo de esta manera se evita la sobrecarga y replicación de programación en cada servicio web al concentrar la autenticación en un único punto especialmente desarrollado para ello.

Para el desarrollo del modelo se contemplan los siguientes componentes principales los cuales interactuaran entre si, como lo muestra la Figura 5 .

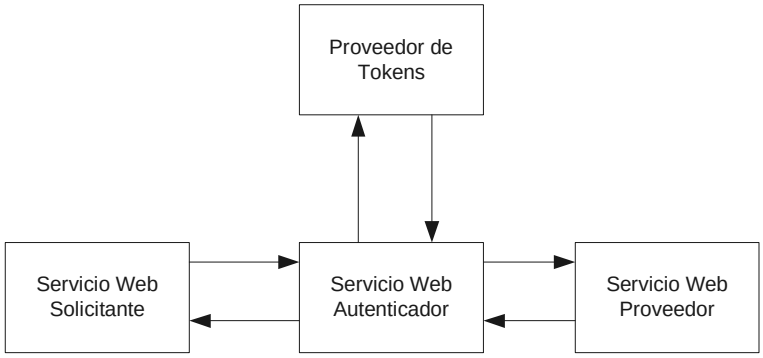


Figura 5: Modelo de Interacción básica entre Servicios Web

Estos componentes comprenden la vista general del modelo, cada uno de ellos, será explicado brevemente a continuación y detallado en las siguientes secciones del capítulo.

Servicio Web Solicitante

El Servicio Web Solicitante es un servicio web que necesita comunicarse con otro servicio web para solicitar información o la ejecución de un proceso. Este servicio también puede asumir el rol de proveedor, cuando reciba peticiones de procesamiento o información. Detalles de su funcionamiento y estructura serán detalladas en los siguientes apartados.

Servicio Web Proveedor

El Proveedor es un Servicio Web el cual expone funcionalidades para permitir que otros consuman las funcionalidades que este tiene o bien para proveer de información. Este Servicio web tiene la necesidad de asegurarse de que las peticiones que recibe son de quien él cree, a su vez, este servicio también puede asumir el rol de solicitante, cuando realice solicitudes de información. Detalles de su funcionamiento y estructura serán detalladas en los siguientes apartados.

Servicio Web Autenticador

El intermediario es un servicio web donde está alojada la mayoría de la lógica de autenticación, el intermediario se encarga de verificar la identidad del solicitante así como también , proveerle a este, credenciales que le permitan comprobar al proveedor que su identidad ya ha sido verificada.

Proveedor de Tokens

El proveedor de token es una entidad que provee de tokens a el Intermediario a cada petición de él, para lo cual, en cada generación de token, se guarda un registro del token y su tiempo de caducidad, para futuras comparaciones. El funcionamiento y estructura de este componente será descrita en las siguientes secciones.

Archivo Manifiesto

El archivo manifiesto es un archivo XML, el cual cuenta con una estructura basada en XML Encryption. Este archivo contiene los siguientes campos:

- IdWS
- TipoToken
- Token
- Fecha Emisión
- Fecha Caducidad
- URL Permitidas

Las URL permitidas indican las URL a las cuales el servicio web puede tener acceso [28].

Estos son los campos principales, siguiendo con las especificaciones provistas por XML Encryption de parte de WS-Security, en el mismo archivo se deberán de especificar las partes encriptadas y el algoritmo utilizado para realizar esto. Este archivo permanece encriptado para evitar, en caso de ser accedido o robado por un extraño, la revelación de la información que contiene.

Este archivo se encuentra residente en cada uno de los servicios web que utilicen el modelo que se propone. Un archivo manifiesto contiene una lista de dominios los cuales a los cuales se les permite el acceso a la información o funcionalidad que pública, como lo muestra la Figura 6.

```
<ServWeb id="468300000">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
        Type="http://www.w3.org/2001/04/xmlenc#Element">
        <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"
            xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" />
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <xenc:EncryptedKey
                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                <xenc:EncryptionMethod
                    Algorithm="http://www.w3.org/2001/04/xmlenc#kw-tripledes"
                    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" />
                <xenc:CipherData
                    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                    <xenc:CipherValue
                        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                        gMp/3ZuYVyHn74JDKr3WCLDrf7H+S6wLqGEdRdgqQGw=
                    </xenc:CipherValue>
                </xenc:CipherData>
            </xenc:EncryptedKey>
        </ds:KeyInfo>
        <xenc:CipherData
            xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
            <xenc:CipherValue
                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                Wr1njyJlYYOM9lAYqcwGCWkw2L4pUjQD2GGVoU9lVZ0wKqHY8y3l3GY8FY4i5K3G8grIe2xN4u7x
                7RtkFiXZgGMeYnQp6oB6ckKp3KFKHVqtucc9AVzOgC7XAw/oe61HRFqe6RRVzXjNMLU5TaV7lJF1
                I8NVPQmUSDX7NRtnR68=
            </xenc:CipherValue>
        </xenc:CipherData>
    </xenc:EncryptedData>
    <URL>"http://dominio.com/funcion"</URL>
    <URL>"http://dominio2.com/funcion"</URL>
    <URL>"http://dominio3.com/funcion5"</URL>
    <URL>"http://dominio.subdominio/funciones"</URL>
</ServWeb>
```

Figura 6 : Archivo Manifiesto

Token

Un token puede ser un certificado digital, un token binario, o un password alfanumérico, estos pueden cambiar según sea necesario, para poder utilizarlos se deberá de construir el servicios web en base a WS-Policy, de tal forma que el servicios web involucrado tenga información acerca de cómo comunicarse con el otro servicio web.

3.5 Modelo de Autenticación

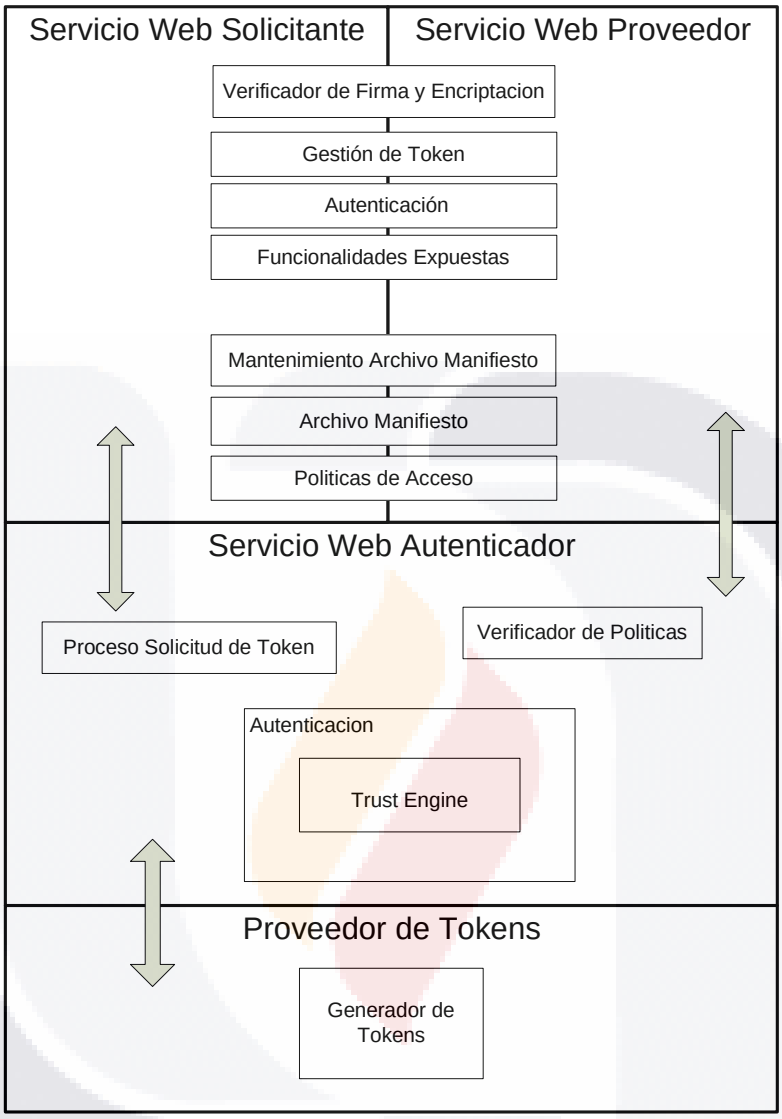


Figura 7: Modelo De Autenticación

Como lo muestra la Figura 7, se presenta el modelo de autenticación propuesto, este está basado en los componentes anteriormente descritos, cada uno a su vez cuenta con varios procesos internos como se muestra:

- Servicio Web Solicitante/Proveedor:
  - Gestión de token
  - Proceso de autenticación
  - Funcionalidades expuestas
  - Actualización de archivo manifiesto
  - Archivo manifiesto
  - Políticas de acceso

- Servicio Web Autenticador
  - Proceso solicitud de token
  - Mantenimiento archivo manifiesto
  - Archivo manifiesto
  - Proceso de autenticación
- Proveedor de tokens

A continuación se describe el proceso general de autenticación utilizado en el modelo

3.5.1 Proceso General de Autenticación

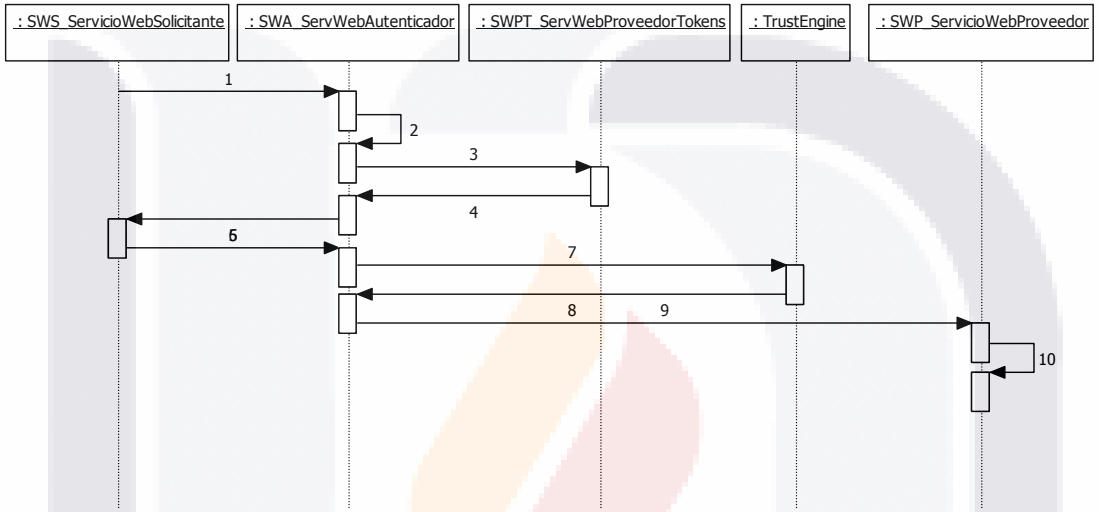


Figura 8 Proceso General de Autenticación

El proceso general de autenticación se realiza conforme a la Figura 8, como se muestra a continuación:

1. El servicio Web Solicitante, realiza una solicitud de autenticación, al intermediario para que provea los medios necesarios para comunicarse con el servicio web proveedor. Cabe destacar que las peticiones se realizan usando el estándar WS-Security para proteger la confidencialidad e integridad del mensaje.
2. El intermediario, al verifica la identidad del servicio web solicitante.
3. El intermediario genera una solicitud al Proveedor de tokens, el cual le regresa un token, este token se registra.
4. El servicio proveedor de tokens genera un token y lo envía a el autenticador (servicio web Autenticador)
5. El intermediario al obtener el token, regresa el token generado al servicio web que lo solicito.
6. El Servicio web solicitante almacena el token y crea una petición hacia el intermediario con la intención de comunicarse con un servicio web proveedor, enviando el token que previamente había recibido.
7. El intermediario verifica el token recibido con el Trust Engine para verificar la autenticidad del mensaje

8. El Trust Engine regresa la respuesta de la validación del token.
9. El mensaje es enviado al servicio web Proveedor
10. El servicio web proveedor verifica el mensaje y procesa la petición.

Lo anteriormente mencionado es una descripción a grandes rasgos, los siguientes apartados mostraran, el comportamiento a detalle de cada uno de los componentes involucrados así como sus relaciones con otros.

Infraestructura PKI

Dentro del modelo propuesto, es fundamental el papel que ocupa una infraestructura basada en PKI, es decir en llaves públicas y privadas, las cuales como se menciono anteriormente, la información que se encripta en base a estas, no puede ser obtenida sin su correspondiente par de llave [30] .

Para lo cual, inicialmente se deberá de generar un par de llaves, las cuales funcionaran para brindar una autenticación al mensaje respecto de su procedencia. Este par de llaves puede ser generado con software diverso como lo es OpenSSL, el cual permite generar un par de llaves: una publica y una privada.

Distribución

La llave privada deberá de permanecer en un lugar seguro dentro del servidor de autenticación, ya que en ella reside la clave del éxito en la integridad de los mensajes intercambiados. Por otra parte, la llave publica, puede ser distribuida a todos los servicios involucrados, sin que para ello exista una necesidad de mantener esta llave en secreto, puesto que sin la llave privada, es poco lo que se puede hacer.

Identidad de Servicios Web

Debido a la propia naturaleza del software, un difícil proveer una identidad a ese basada en sus rasgos, puesto que no contiene rasgos que lo hagan único. Por lo cual se plantea el asignar un Id único al servicio web, el cual se mantendrá dentro de la aplicación que haga uso de esta. Podrían cuestionarse otras opciones como la asignación de un certificado digital a cada aplicación, firmar las mismas solicitudes que genera la aplicación, pero estas estarían sujetas a lo que otra entidad crea acerca del software. Por ese motivo se toma en cuanta un factor de autenticación 2 el cual es mencionado en [30].

Un factor de autenticación 1 implica “algo que usted sabe”, como lo sería una contraseña o un Pin, el cual desde el punto de vista de esta investigación, es posible que la entidad que hace uso de los servicios web, le hayan sido substraídas.

“Un factor de autenticación 2 implica “algo que usted tiene”. Las señales de autenticación son autenticadores de *factor dual* o *factor 2*. Para usar una señal de autenticación, se debe tener la misma señal (algo que se tiene) y se debe de tener una contraseña correcta (algo que se sabe). Si un intruso lograra acceder a la señal de autenticación, no podría utilizarla a menos que también conociera el PIN, de forma similar si el atacante tuviera la seña de autenticación, no podría utilizarla puesto que no tiene la contraseña” [30].

Por lo cual para poder asignar una marca única a un servicio web, se plantea asignarle un ID, en su interior, es decir embebido en su programación. De tal forma que incluso si se vulnera la seguridad del servidor donde el servicio web se encuentre residente, no exista algún registro o prueba de posesión que permita robar la identidad que podrá tener el servicio web, como podría ser un archivo "cookie" donde estuvieran sus credenciales, un certificado digital o bien su registro en una BD en el lado de cliente.

La estructura del Id se generara en base a la notación de Año+Mes+Dia+Hora+Minuto+Segundo+Milisegundo, lo le permitiría ser único e irrepetible, puesto que el momento de su generación solo ocurre una vez.

Este Id estará registrado en una base de datos residente en el Servicio Web **Autenticador**, cabe destacar que no estará el Id en texto plano, puesto que esto implicaría un riesgo, en su lugar estar el resumen del ID después de haber aplicado un algoritmo Hash, como se muestra en la siguiente Tabla 4.

Id	Algoritmo	Resumen
2010102503184599	SHA-1	ba46044101f3f999cf8f8d5472ba01d2acfa912f

Tabla 4: Ejemplo de Hash

La única forma en la que se tendrá acceso al ID será mediante los métodos internos del Servicio web, en el modulo de autenticación como se describirá más adelante. Incluso al accesar a este ID, solo se obtendrá una reseña de hash, esto al realizar las comunicaciones con el **Autenticador**. Por otra parte El ID interno también servirá como clave de encriptación del archivo manifiesto Tk.xml, en el cual se almacenan tanto tokens como URL´s con las que se comunica el Servicio Web.

3.5.2 Servicio Web Autenticador (SWA)

El Servicio Web Autenticador es la entidad de software en la cual se aloja la mayoría de la lógica de autenticación. Dentro de sus principales funciones se encuentran:

- Verificar la identidad de los participantes.
- Actuar como entidad aseguradora de confianza entre los servicios web involucrados.
- Gestionar la asignación y creación de tokens a usarse entre servicios web como prueba de identidad.

Para lo cual el Servicio Web Autenticador cuenta con varios componentes como lo muestra la Figura 9.

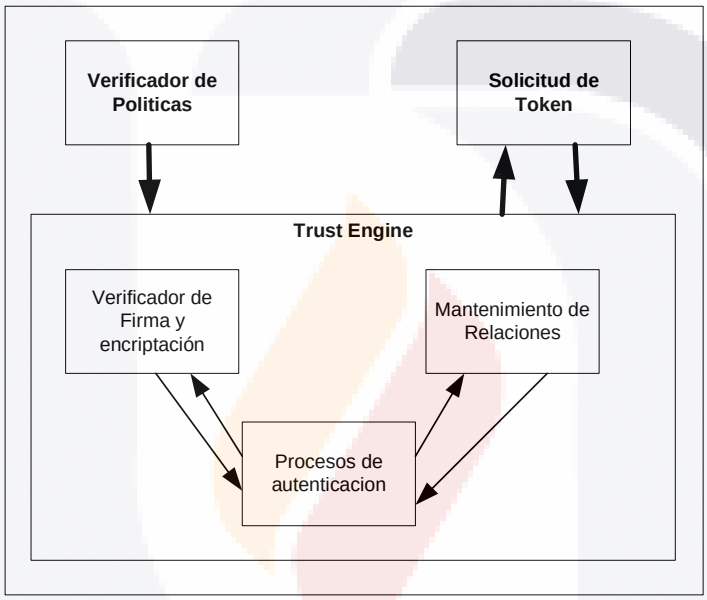


Figura 9: Componentes de Intermediario (SWA: Servicio Web Autenticador)

Los componentes que lo integran se describen a continuación:

Modulo Verificador de Políticas

Este componente es el encargado de verificar que los mensajes que recibe el servicio web cumplan con las políticas estipuladas. Tanto en formato como en contenido

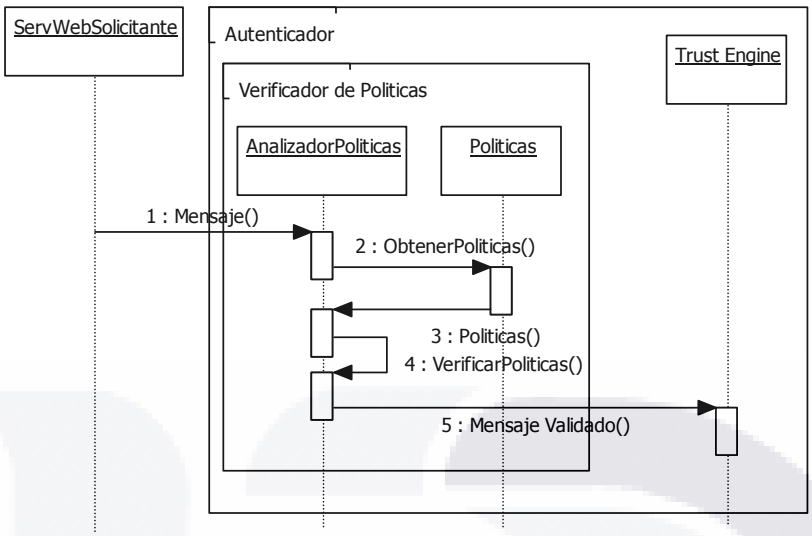


Figura 10: Verificador de Políticas

Conforme a la Figura 10, se describe su comportamiento a continuación;

1. El servicio web solicitante envía un Mensaje
2. Se solicitan las políticas.
3. El **Analizador de políticas,** obtiene las políticas del servicio web de **Políticas**.
4. Se verifican las políticas contra el mensaje enviado, verificando que cumpla tanto en formato como en contenido.
5. Si el mensaje es válido es enviado a el **Trust Engine**

Trust Engine

El Trust Engine es un componente conceptual que evalúa los aspectos relacionados con la seguridad de un mensaje [32] basándose en el Trust Engine definido en [34] se describen los siguientes componentes.

Verificador de firma y encriptación

Este componente es el encargado de gestionar la integridad y confidencialidad de los diferentes mensajes intercambiados entre los servicios web involucrados. Dentro de sus tareas se encuentran las siguientes:

- Verificar la firma digital incluida dentro de los mensajes.
- Desencriptar y encriptar la información contenida dentro de los mensajes entrantes y salientes, en base a la llave privada residente dentro del **Autenticador**.

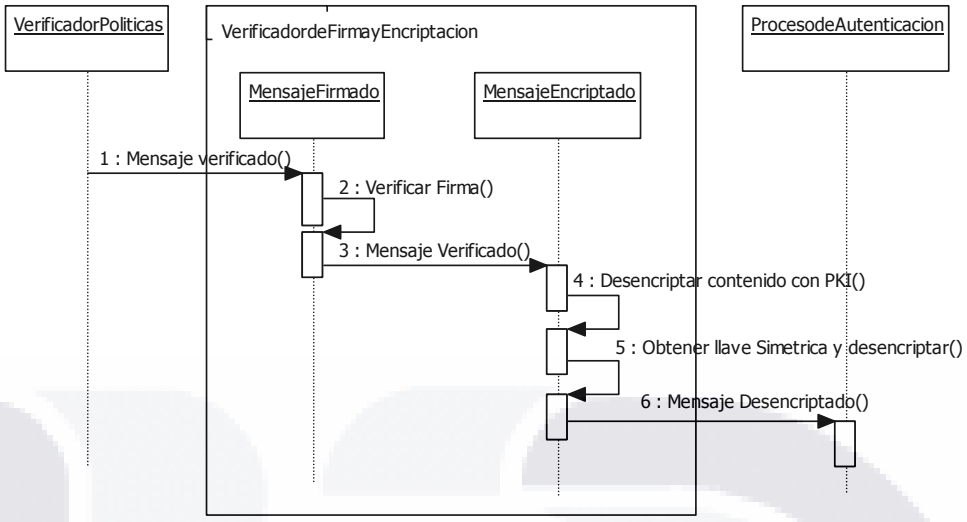


Figura 11: Verificador de Firma y Encriptación Entrantes

Este modulo como se muestra en la Figura 11 tiene como meta el obtener los datos que le son enviados al **Proceso de Autenticación**. Esto mediante la comprobación de la firma digital presente dentro de los mensajes que recibe. Este modulo obtiene los mensajes directamente del modulo **Verificador de Políticas.** Este modulo contiene dos comportamientos como se describe a continuación:

Verificación de Firma y Encriptación Entrantes

Este modulo como se muestra en la Figura **11** procesa los mensajes entrantes, a través del **Verificador de Políticas** como se muestra a continuación:

1. El mensaje verificado correctamente en **Verificador de Políticas** es enviado al modulo **Verificador de Firma y Encriptación.**
2. El submodulo **Mensaje Firmado** verifica con la llave privada residente dentro del **Autenticador** el contenido del mensaje a fin de que se tenga la certeza de que este no ha sido modificada su contenido durante su envió.
3. Una vez que el mensaje ha sido verificado, este es enviado al submodulo **Mensaje Encriptado**, para su procesamiento.
4. Dentro del submodulo **MensajeEncriptado** se procede a obtener mediante el uso de la llave privada residente dentro del **Autenticador**, la llave simétrica con al cual el contenido del mensaje fue codificado.
5. Una vez obtenida la llave simétrica se decodifica el mensaje y se obtiene el contenido del mensaje.
6. El contenido del mensaje es enviado al modulo **Autenticación** donde se evaluara la información que contiene el mensaje para determinar la identidad y los privilegios con los que cuenta el remitente del mensaje.

Como se menciono anteriormente este modulo cuenta con dos comportamientos, siendo el segundo descrito a continuación y mostrado en la Figura 12:

Verificación de Firma y Encriptación Salientes

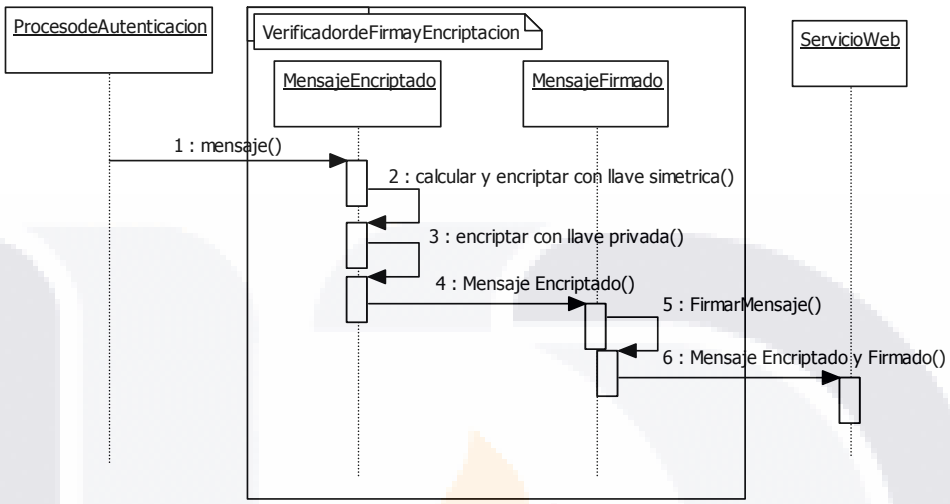


Figura 12: Verificador de Firma y Encriptación Salientes

Este submodulo al contrario del **Verificador de Firma y Encriptacion Entrante** es el encargado de agregar firma y encriptación a los mensajes que son enviados como respuesta, esto con la misión de poder asegurar al usuario, la identidad del remitente y proteger los datos enviados mediante técnicas de encriptación. El comportamiento del modulo se describe a continuación:

1. El mensaje procedente de **Proceso de Autenticación** es enviado al modulo **Verificador de Firma y Encriptación** para ser encriptado y firmado
2. El mensaje es obtenido por el submodulo **MensajeEncriptado**, en base a el contenido del mensaje una llave simétrica es generada, y con esta se encripta el mensaje.
3. La llave simétrica con la cual fue encriptado el mensaje, es encriptada con la llave privada la cual se encuentra residente dentro del **Autenticador,** una vez que la llave simétrica esta encriptada, esta es adjuntada al mensaje.
4. El mensaje es enviado el submodulo **MensajeFirmado.**
5. Se firma el mensaje con la llave Privada.
6. El mensaje es enviado como respuesta al servicio web que lo solicito

Lo anteriormente descrito permite crear un mensaje más ligero tanto en tamaño como en tiempo de procesamiento, puesto que al encriptar un mensaje en base a una llave simétrica, generara un contenido encriptado más ligero que el que generaría el uso de una llave asimétrica. Para cubrir una de las características de las llaves simétricas que implica cierta debilidad, se uso la llave asimétrica, de esta forma la llave simétrica es encriptada con la llave asimétrica, lo cual permite una mejora en tiempo de procesamiento al desencriptar el mensaje.

Procesos de Autenticación

Dentro de esta tarea se encuentra la lógica mediante la cual se determina la identidad del solicitante y las relaciones que tiene con otros servicios web.

Dentro de sus tareas se encuentran las siguientes:

- Verificar la identidad de los mensajes entrantes.
- Determinar las relaciones con otros servicios web.
- Gestionar la generación de tokens.

Cada uno de los servicios web involucrados deberá de obtener un token del **Autenticador**, esto como forma de agilizar el proceso de autenticación, ya que la siguiente vez que quieran acceder a un servicio web en específico, bastara con presentar su token.

Dentro del **Trust Engine** se cuenta con una base de datos dedicada únicamente para el **Autenticador**, en este caso el esquema base de la base de datos cuenta con las siguientes tablas y los siguientes campos sugeridos:

- Id: Id de registro
- IDWS: Id del servicio web, este Id es único e irrepetible para cada uno de los servicios web involucrados. Este Id estará registrado como la reseña generada por el Id original en base a la aplicación de un algoritmo de Hash como lo es SHA-1 o MD5. El incluir solo la reseña del IDWS permite que nadie conozca el Id del servicio web, incluso si se lograra el robo de la base de datos, no se podría obtener el Id de los servicios web, puesto que los algoritmos Hash no permiten su cálculo a la inversa
- URI: Indica la dirección en la cual se encuentra residente el servicio web

Además cuenta con tablas detalle cómo se muestra a continuación **TablaDetalle1**:

- Id: Id de registro.
- IDWS: campo de relación con identificador único del servicio web. El contenido no tendrá el IDWS asignado a el servicio web, en su lugar contendrá el resumen generado por un algoritmo hash, esto con la finalidad de proteger los datos en caso de que la tabal sea sustraída
- Hash: Indica el tipo de algoritmo hash usado

Y una más con la siguiente estructura **TablaDetalle2**:

- Id: Id de registro.
- IDWS: campo de relación con identificador único del servicio web.
- Tipo de Token: tipo de token en uso.
- Token: Token que se está usando.
- FechaExpedicion: Fecha en la cual el token fue expedido.
- FechaCaducidad: Fecha en la cual el token caduca.

Como se puede observar, la primera estructura descrita actuaria como tabla maestra, y las 2 tablas siguientes actuarían como tabla detalle, en las cuales, en cada cambio las transacciones realizadas

por el servicio web quedarían reflejadas. En las tablas nunca se hace referencia a la URI que va a asociada con el servicio web. Esto es debido a que en caso de de que la tabla sea substraída, no se podría saber a cual URI pertenecería el hash residente en la tabla.

Para la implementación de esta BD, se pueden tomar cualquiera de las presentes en el mercado como lo será MySQL, SQL Server entro otras. Cabe destacar que dentro del modulo de autenticación, a la BD solo se puede tener acceso desde estos módulos, en las funcionalidades de registrar y eliminar tokens que sean usados por el servicio web.

Este modulo cuenta con dos comportamientos los cuales serán descritos a continuación:

Autenticación con Solicitud de Token

Este modulo específica la forma en la cual un servicio web debe de solicitar un token a fin de poder comunicarse con otros servicios web, como lo muestra la Figura 13.

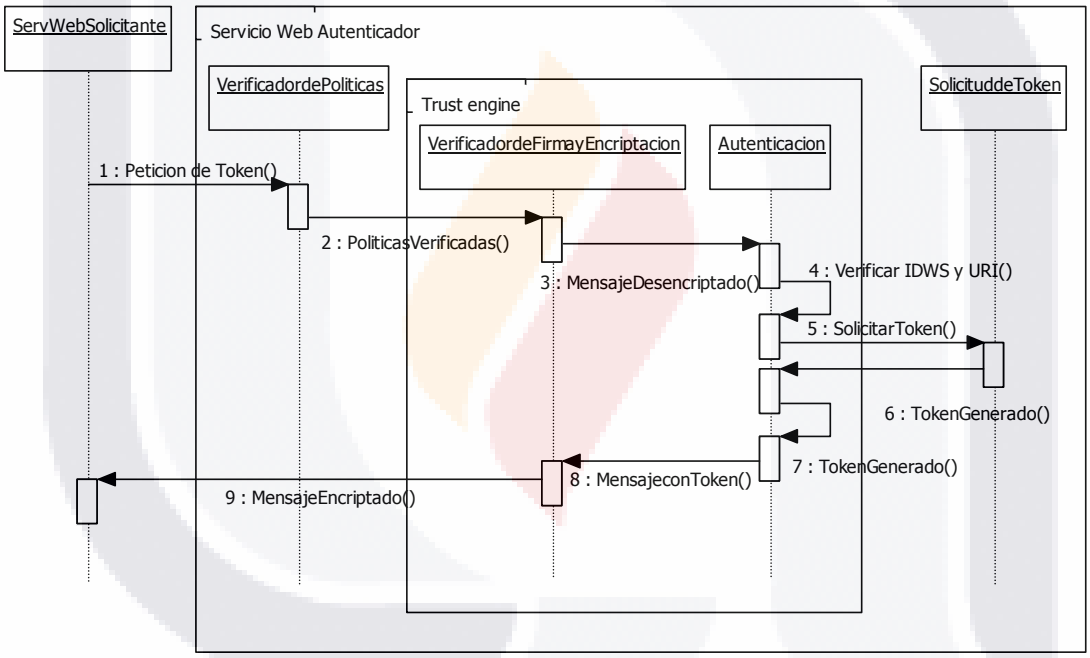


Figura 13: Autenticación con Solicitud de Token

El proceso mediante el cual se lleva a cabo la autenticación se describe a continuación tomando como referencia la Figura 13

1. El Servicio Web Solicitante genera una petición de solicitud de token, en esta petición incluye su URI, y su IDWS (la reseña generada por la aplicación de un algoritmo de hash), este mensaje ha sido encriptado y firmado con un proceso similar al descrito en **Verificador Firma y Encriptacion Saliente**, con la diferencia que el mensaje ha sido encriptado con la llave publica que previamente le fue provista.
2. El mensaje es revisado por el verificador de políticas.

3. El mensaje es verificado y decodificado en **VerificadordeFirmasyEncriptacion**.
4. Dentro de **Autenticación** se verifica el IDWS y la URI de procedencia, que venían en el mensaje contra las registradas previamente en la BD.
5. Una vez verificada, se realiza una solicitud a **SolicituddeToken** para que genere un token para el mensaje.
6. El token es regresado de **SolicituddeToken** a **Autenticación**.
7. El token es registrado.
8. Se crea un mensaje de respuesta donde se incluye el token recién generado, este mensaje es encriptado como de describió en **Verificador de Firma y Encriptación Salientes**.
9. El mensaje es enviado al Servicio Web que lo solicito.

El Servicio Web que origino la solicitud al recibir el mensaje, al estar encriptado o solo firmado por el **Autenticador**, le da la confianza de que el mensaje proviene de quien él cree, ya que la única forma que tiene para verificar la firma o desencriptar el mensaje es utilizando la llave publica que se le fue provista.

Autenticación con Token

El siguiente comportamiento dentro del modulo de autenticación, es aquel en el cual, el Servicio Web Solicitante ya cuenta con un token, el cual previamente ya ha negociado con el **Autenticador**, como se describió anteriormente en **Autenticación con Petición de Token,** este proceso se muestra en la Figura 14.

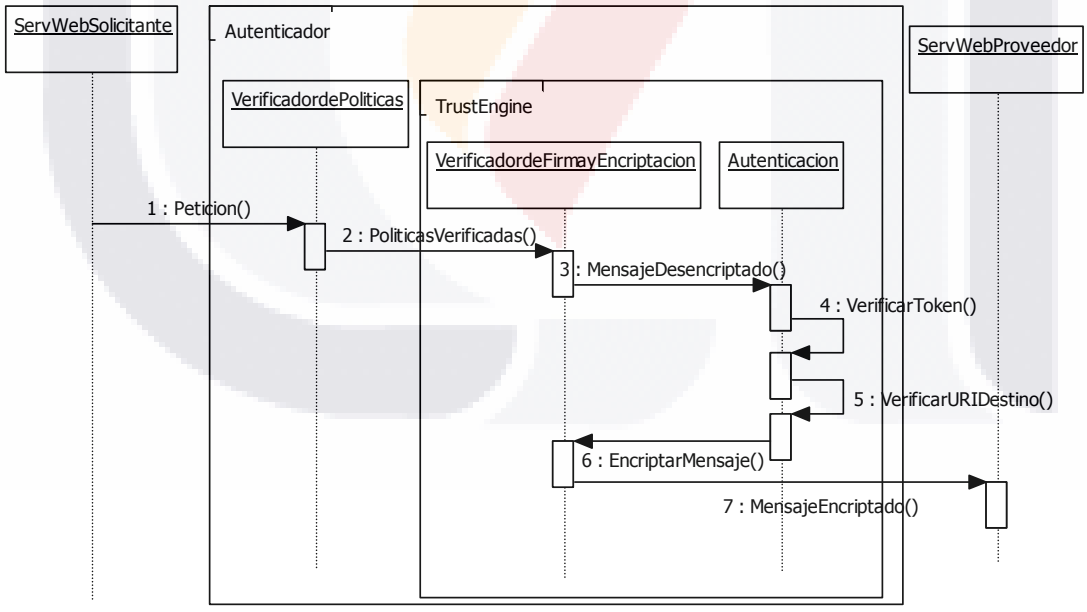


Figura 14: Autenticación con Token

El proceso mediante el cual se lleva a cabo la autenticación es el siguiente:

1. El Servicio Web Solicitante, genera una petición, en la cual incluye el token que previamente debió de haber negociado, su IdWS, su URI, y la URI del servicio web con el cual él desea

comunicarse, este mensaje previamente deberá de estar encriptado con un proceso similar al que se describió en **Verificador de Firma y Encriptacion Salientes,** pero del lado del cliente

2. La petición es verificada por el **Verificador de Políticas,** para determinar si el mensaje respeta las políticas que previamente se establecieron acerca del servicio Autenticador, a fin de estandarizar el formato de las comunicaciones.
3. El mensaje es comprobado y desencriptado en el **Verificador de Firma y Encriptación.**
4. Una vez que el mensaje ha sido desencriptado, se extrae el contendido del campo IDWS y la URI de procedencia, se verifica que estas se encuentran registradas dentro de la BD, Además se verifica el token que venía incluido en el mensaje, se revisa la fecha de expedición y al caducidad del mismo.
5. Una vez comprobado lo anterior, se verifica si el servicio web puede tener acceso a al URI destino, esto al compararlo contra los registro de la BD.
6. Si el servicio puede tener acceso, se envía la petición original a **VerificadordeFirmayEncriptacion,** para que lo encripte y firme con la llave privada residente en el **Autenticador**.
7. El mensaje encriptado es enviado al Servicio Web **Proveedor**, cabe destacar que no se envía el token que originalmente había sido enviado por el Servicio Web Solicitante, si el Servicio Web Proveedor necesita responder el mensaje, deberá de negociar un token con el **Autenticador** de la misma forma en que el solicitante lo realizo.

Mantenimiento de Relaciones en Autenticador

El submodulo de mantenimiento de relaciones, tiene como función, el gestionar las relaciones existentes en entre los servicios web involucrados. Este submodulo nace de la necesidad de permitir a cada uno de los involucrados, el seleccionar con quien compartir la información o funcionalidades que exponen, ya que se evita que la toma de decisiones se centralice en un punto, a la espera de que algún administrador aplique realice los cambios solicitados. El proceso mediante el cual se realiza como lo muestra la Figura 15

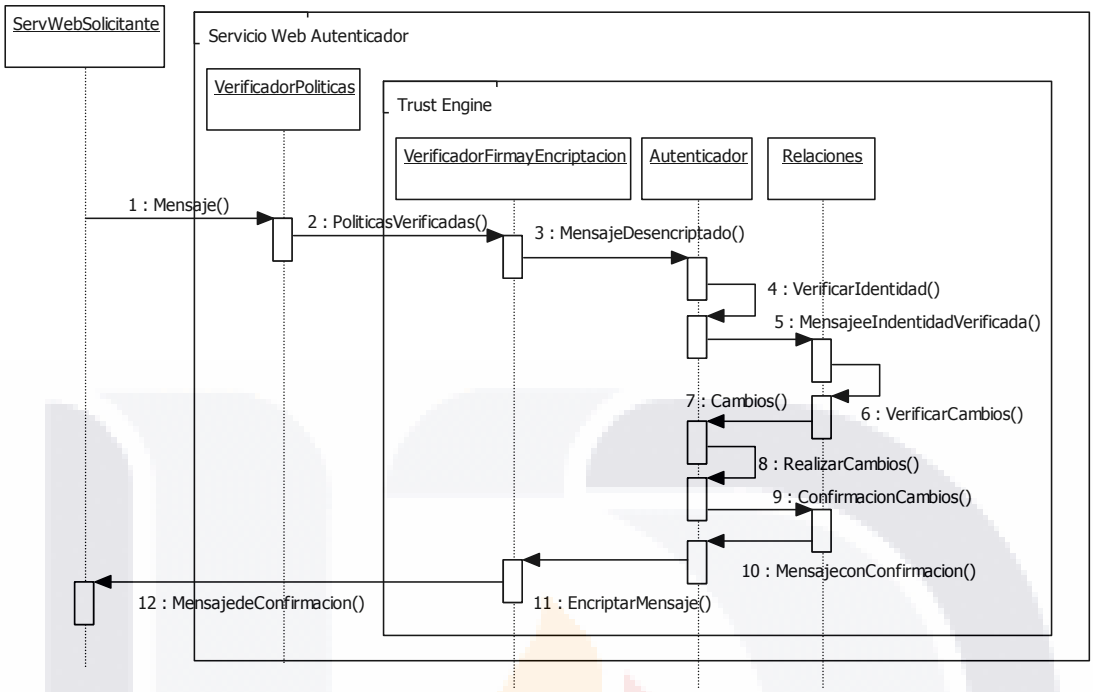


Figura 15: Mantenimiento de Relaciones en Autenticador

1. El Servicio Web Solicitante envía un mensaje con la petición de cambios a realizarse. Cabe destacar que en el mensaje deberá de incluir las URI, que desea agregar o eliminar. Además deberá de agregar su token de sesión el cual previamente deberá de haber negociado, así como su IDWS, a fin de que Autenticador lo pueda reconocer como tal. Este mensaje estará encriptado en base a una llave simétrica incluida dentro del mensaje la cual estar encriptada con la llave publica.
2. El mensaje deberá de ser analizado por el verificador de Políticas.
3. El verificador de Firma y Encriptación obtiene los datos contenidos dentro del mensaje.
4. El **Autenticador** verifica la identidad del solicitante como se describió anteriormente en **Autenticacion con Token**.
5. El contenido del mensaje es enviado al submodulo de relaciones.
6. Se verifican las relaciones existentes contra los cambios que se solicitan dentro del mensaje.
7. Se envían los cambios a realizarse dentro del **Autenticador**.
8. Dentro del **Autenticador**, el cual es el único que tiene acceso a la BD, realiza los cambios solicitados.
9. La confirmación de los cambios realizados es enviada al submodulo **Relaciones**.
10. El mensaje es enviado al submodulo de **Verificacion y Firma** para que sea encriptado y enviado al solicitante.
11. El mensaje es Encriptado.
12. El mensaje encriptado y es enviado al solicitante con la confirmación de los cambios realizados

Tarea SWA: Proceso de Mantenimiento de Archivo Manifiesto

El Proceso de Mantenimiento del Archivo Manifiesto residente en el intermediario se realiza como lo muestra la Figura 16: Proceso de mantenimiento en intermediario:

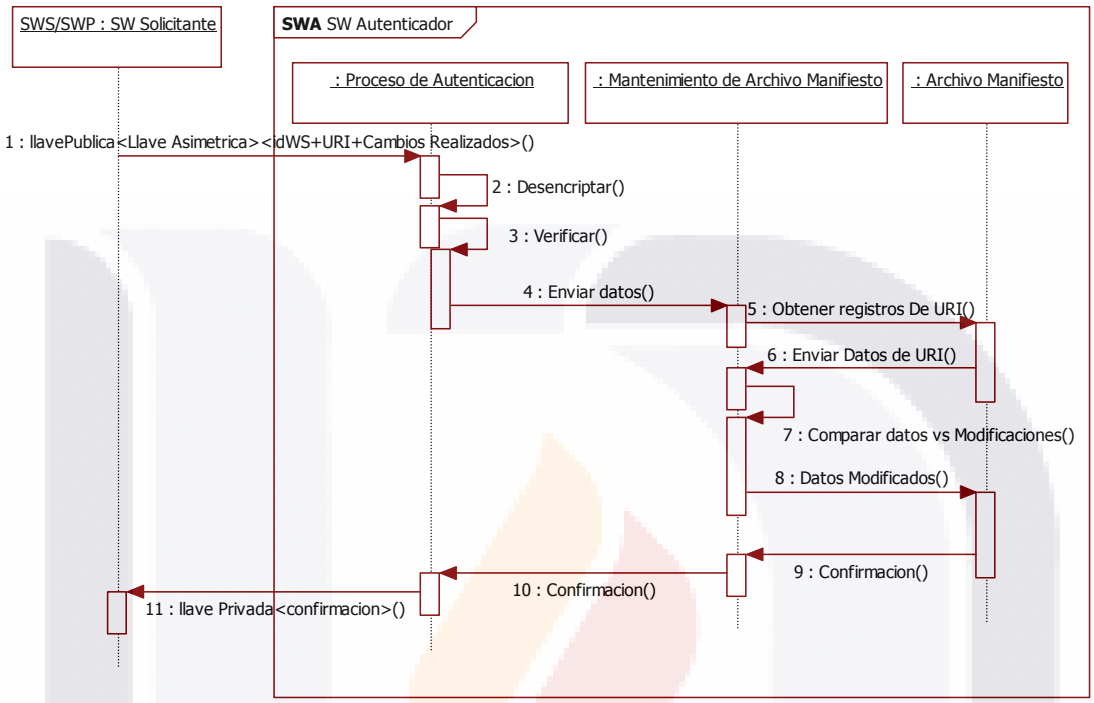


Figura 16: Proceso de mantenimiento en intermediario

En el proceso de mantenimiento del archivo manifiesto intervienen dos componentes: el intermediario (SWA) y un servicio web, el cual pude ser tanto un solicitante como un proveedor (debido a que puede tener los dos roles) (SWP/SWS).

1. Un servicio web (proveedor/solicitante), al realizar cambios en su archivo manifiesto, envía una petición de actualización al intermediario (SWA), la petición es enviada con los siguientes datos: Petición de Actualización + IDWS + Datos a ser actualizados + URL de Procedencia, estos van encriptados mediante una llave simétrica generada, la cual deberá de ir protegida con la llave publica provista previamente.
2. Al recibirlos el Intermediario (SWA) mediante el **Proceso de Autenticación** verifica la autenticidad del mensaje, en caso de que el mensaje sea correcto, procede a actualizar el archivo manifiesto en base a los datos recibidos.
3. Una vez actualizados los datos, se procede a verificar la identidad del solicitante mediante el proceso de autenticación.
4. Una vez verificado se envía una mensaje al proceso de mantenimiento para que realice los cambios necesarios
5. Al recibir la petición, procederá a recuperar los registros dentro de los archivos manifiesto almacenados, respecto del IDWS que lo está solicitando.

6. El conjunto de datos es enviado al proceso de mantenimiento de archivo manifiesto.
7. Este compara la información de los registros vs los cambios que se solicitan realizar.
8. Los cambios son realizados sobre los datos y se graban en el archivo manifiesto.
9. Se procede a enviar una confirmación de cambios realizados.
10. El proceso mantenimiento de archivo manifiesto, envía una confirmación al proceso de autenticación.
11. El proceso de autenticación genera un mensaje de confirmación el cual es enviado al servicio web solicitante (SWS) que envió la solicitud de modificación, la confirmación la envía protegida mediante su llave privada.

Tarea SWA: Proceso de Solicitud de Token

El proceso de solicitud de token, es el encargado de solicitar la generación de un token a el Servicio Proveedor de Tokens (SPT), este servicio al recibir la petición lo genera y regresara al intermediario, el cual lo registrara dentro de su archivo manifiesto respecto el servicio web que lo solicito inicialmente. Esto como lo muestra Figura 17: Proceso de Solicitud de Token

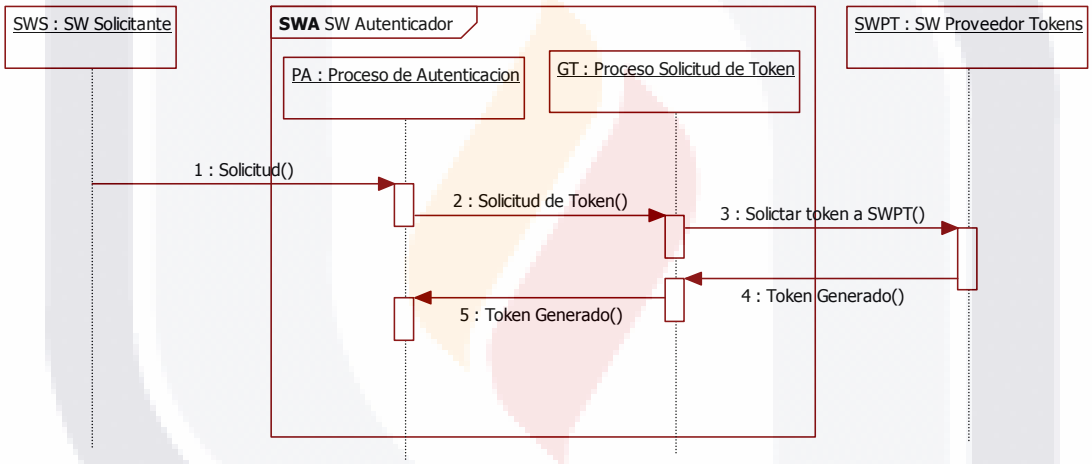


Figura 17: Proceso de Solicitud de Token

El proceso de solicitud de token es el encargado de solicitar tokens hacia el proveedor de los mismos, siendo este el único punto desde el cual se pueden hacer peticiones, como se muestra a continuación

1. El Servicio Web Solicitante (SWS), realiza una petición al Servicio Web Autenticador (SWA), dentro de los procesos de este, primero se verifica el mensaje en el Proceso de Autenticación.
2. Si el mensaje es correcto, se envía una solicitud al Proceso de Solicitud de token a fin de que solicite uno y se lo provea
3. Al recibir la petición, el Proceso de Solicitud de Token, envía un mensaje al **Servicio Proveedor de Tokens** (SPT), para lo genera.
4. El token una vez generado, es reenviado al **Proceso de Solicitud de Token**.

5. Una vez que este lo recibe lo reenvía a el proceso de autenticación, el cual lo había solicitado originalmente.

3.5.3 Servicio Web Solicitante / Proveedor (SWS / SWP)

Los servicios web, solicitantes/proveedores, cuentan con varios componentes como se muestra a continuación en Figura 18, los cuales serán descritos:

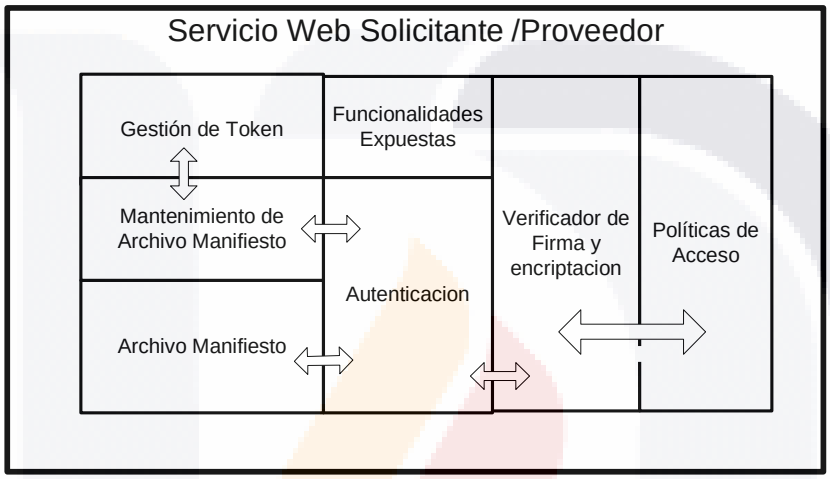


Figura 18: Componentes de Servicio Web Solicitante/ Proveedor (SWS/SWP)

Tarea SWS/SWP: Archivo manifiesto

Este archivo como se mostro previamente en Figura 6, es un archivo con una estructura XML, el cual contiene las URI´s permitidas a las cuales puede acceder el servicio web, además contiene el token de sesión, el cual debe de ser previamente negociado con el **Autenticador**.

El archivo se encuentra permanentemente encriptado, siguiendo el modelo recomendado por XML Encryption, de tal forma que solo las partes más delicadas del mismo estarán ocultas. Así que no se pueda acceder a las URI´s ni al token que resguarda. El método de encriptación se basa en una llave simétrica la cual tiene como semilla el IDWS, contenido dentro del servicio web, usando de preferencia los algoritmos TDES, AES o RSA, para encriptación simétrica.

Este archivo es mantenido mediante el **Proceso de mantenimiento de archivo manifiesto** el cual será descrito en las siguientes fases y usando por el proceso **Gestión de Token**.

La estructura del archivo será como se muestra en la Figura 19.

```
<ServWeb id="468300000">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
        Type="http://www.w3.org/2001/04/xmlenc#Element">
        <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"
            xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" />
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <xenc:EncryptedKey
                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                <xenc:EncryptionMethod
                    Algorithm="http://www.w3.org/2001/04/xmlenc#kw-tripledes"
                    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" />
                <xenc:CipherData
                    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                    <xenc:CipherValue
                        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                        gMp/3ZuYVyHn74JDKr3WCLDrf7H+S6wLqGEdRdgqQGw=
                    </xenc:CipherValue>
                </xenc:CipherData>
            </xenc:EncryptedKey>
        </ds:KeyInfo>
        <xenc:CipherData
            xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
            <xenc:CipherValue
                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                Wr1njyJlYYOM9lAYqcwGCWkw2L4pUjQD2GGVoU9lVZ0wKqHY8y3l3GY8FY4i5K3G8grIe2xN4u7x
                7RtkFiXZgGMeYnQp6oB6ckKp3KFKHVqtucc9AVzOgC7XAw/oe61HRFqe6RRVzXjNMLU5TaV7lJF1
                I8NVPQmUSDX7NRtnR68=
            </xenc:CipherValue>
        </xenc:CipherData>
    </xenc:EncryptedData>
    <URL>"http://dominio.com/funcion"</URL>
    <URL>"http://dominio2.com/funcion"</URL>
    <URL>"http://dominio3.com/funcion5"</URL>
    <URL>"http://dominio.subdominio/funciones"</URL>
</ServWeb>
```

Figura 19: Archivo Tk.XML

Al cifrar el documento se realizara un cifrado completo del mismo.

Tarea SWS/SWP: Mantenimiento de Archivo Manifiesto

Este proceso está encargado de dar mantenimiento al archivo manifiesto, cada vez que se realice un cambio en este, se procederá a informar al intermediario acerca de los cambios ocurridos para que los refleje en la copia que se encuentra residente dentro de él. Este proceso se detalla en la Figura 20: Proceso de Actualización de archivo manifiesto.

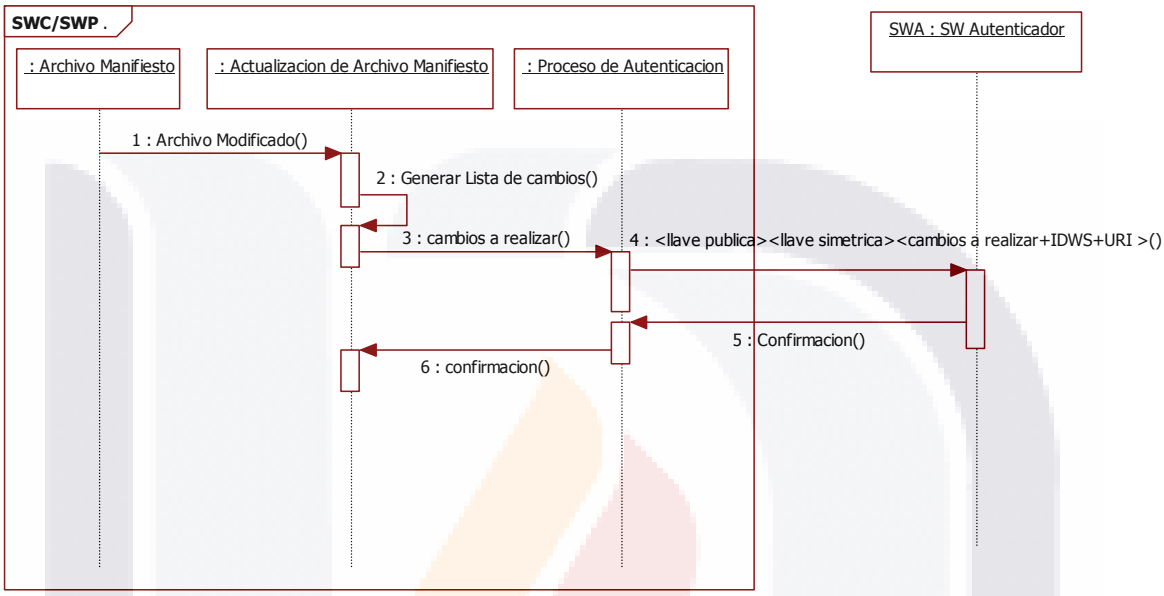


Figura 20: Proceso de Actualización de archivo manifiesto

1. Al realizarse cambios en el archivo manifiesto, se llama al proceso de mantenimiento.
2. El proceso de mantenimiento genera una lista de los cambios realizados en el archivo.
3. La lista de cambios generados es enviada al proceso de autenticación.
4. El proceso de autenticación genera un mensaje en cual se envía hacia el Servicio Web Autenticador (SWA), protegido por una llave simétrica y esta a su vez por la llave publica provista previamente.
5. El Servicio Web Autenticador (SWA) envía una confirmación al servicio web origino la petición, protegida por su llave publica.
6. Una vez recibido el mensaje este es verificado por el proceso de autenticación, extrae el contenido y envía la confirmación al proceso de actualización de archivo manifiesto.

Tarea SWS/SWP: Proceso de Autenticación

Este proceso es otro componente dentro de servicio web solicitante/proveedor, realiza la función de autenticación de las peticiones que recibe.

Las peticiones que recibe, están cifradas total o parcialmente con la llave privada del intermediario, por lo cual utiliza la llave publica que le fue provista para poder acceder a la información contenida en el mensaje, como se muestra a continuación la Figura 21: Proceso de Autenticación SWS/SWP.

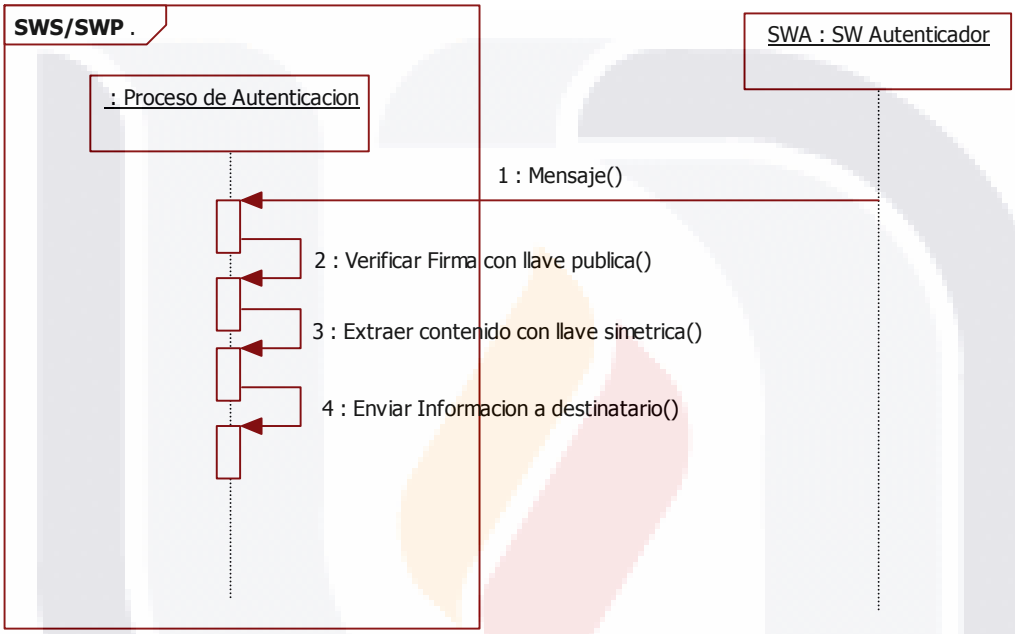


Figura 21: Proceso de Autenticación SWS/SWP

1. El servicio web solicitante/ proveedor (SWS / SWP), recibe un mensaje
2. En el proceso de autenticación se procede a verificar la firma del mensaje con su llave publica, si es correcto se extrae la llave simétrica que este contiene.
3. Con la llave simétrica obtenida se obtiene el contenido del mensaje
4. El contenido del mensaje es enviado al proceso que lo necesita.

Cabe destacar, que el método de encriptación utilizado, esta contenido dentro del mensaje, como lo puede ser TDES, RC5, AES, dependiendo del método a utilizar es que el que se debe de implementar dentro de **Proceso de Autenticación.**

Tarea SWS/SWP: Gestión de Token

El proceso de gestión de token especifica la forma en la cual un token que es negociado con el autenticador (SWA), y este es mantenido dentro del servicio web. Dada la naturaleza de los servicios web, estos no pueden mantener un estado por lo cual el token negociado se guardara en un archivo encriptado, al cual solo se podrá tener acceso desde el servicio web como se muestra a continuación:

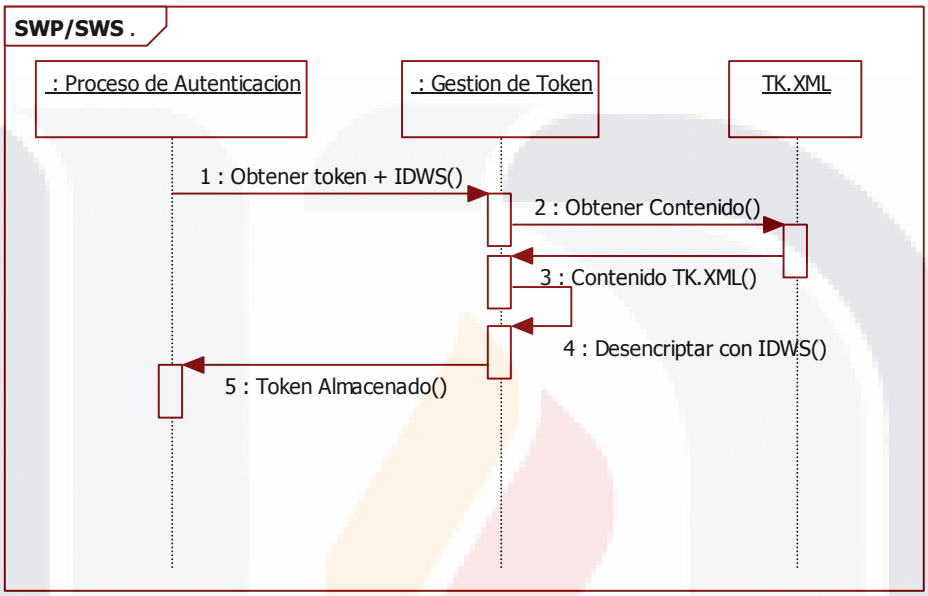


Figura 22: Proceso de Gestion de Token SWS/SWP

1. El proceso de autenticación envía una solicitud al proceso de gestión de token, en la cual también el envía el IDWS que se encuentra residente dentro de la aplicación.
2. El proceso de gestión de token, al recibir la petición obtendrá el contenido del archivo TK.XML.
3. El contenido del archivo TK.XML es enviado al proceso de gestión de token.
4. Este lo desencripta en base a la el IDWS que le fue proporcionado, el cual fue utilizado como clave para la encriptación del archivo.
5. El token que reside dentro de este archivo es enviado al proceso de autenticación que previamente lo había seleccionado.

Cada vez que se negocia un nuevo token, este queda residente en el archivo TK.XML, para protegerlo este se encripta con en base a el IDWS que se encuentra residente en el código compilado del servicio web.

Tarea SWS/SWP: Funcionalidades expuestas

Funcionalidades expuestas comprende todos los procedimientos que expone el servicio web, pero que solo son accesibles una vez que el proceso de autenticación ha sido desarrollado exitosamente. Estas pueden o no estar protegidas por el modelo planteado, esto en función de las funciones que desempeñe. Cuando estas funciones están protegidas, forzosamente se debe de pasar por el proceso de autenticación, dentro de las llamadas a los métodos expuestos.

Tarea SWS/ SWP: Políticas de Acceso

Las políticas de acceso (WS-Policy) definidas por la W3C [35], son empleadas para exponer la foma en la cual se debe de tener acceso a las diferentes funcionalidades que expone un servicio web. Esta especificación permite, el uso de diversos mecanismos para asegurar que la información que se esta recibiendo o enviando esta en los formatos deseados tanto por el solicitante como por el proveedor. En caso de enviar un mensaje que no cumpla con las políticas que el servicio web expone, simplemente se rechaza, evitando el proceso de autenticación.

Esta especificación va de la mano con WS-Security [36], de tal forma que permite asegurar la información que está comunicando, mediante el uso de diversos formatos como lo es: tokens, aserciones SAML, passwords binarios entre otros.

Al incluir WS-Policy dentro del modelo planteado se da flexibilidad de trabajar con diversos formatos de datos para poder lograr la autenticación de un mensaje.

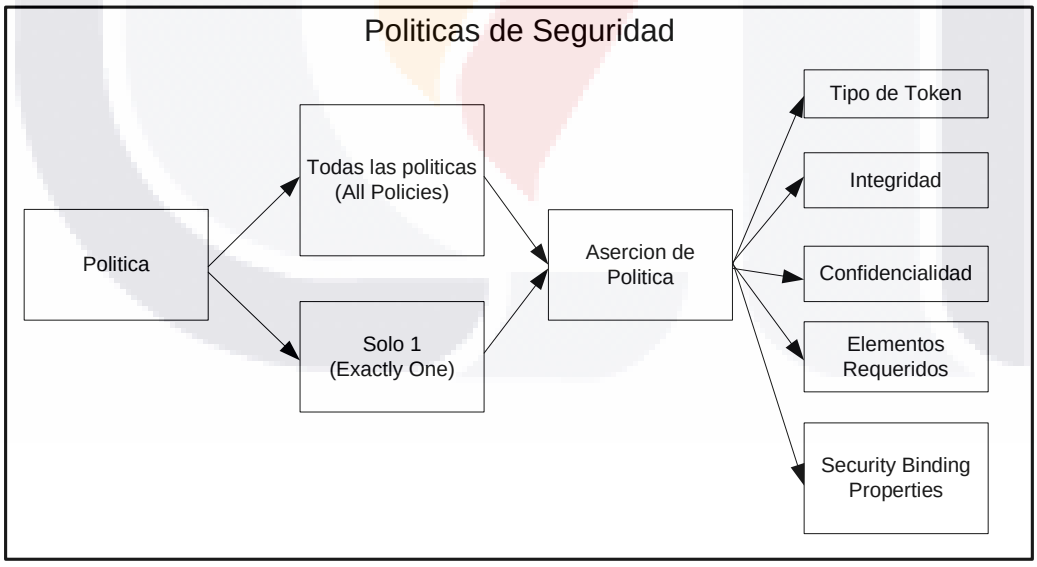


Figura 23: Políticas de Seguridad

Las políticas a usar en el modelo como se define en [37] y [35] se usaran en base a la creación de los diversos apartados de expuestos en la Figura 23: Políticas de Seguridad, en la cual cada una de las políticas que expondrá cada uno de los servicios web contara con lo siguiente:

- **Politica**: Comprende el Id de la política
- **Tipo de política**:
  - All Policies: En este caso todas las políticas publicadas deben de respetarse, existiendo al excepción de que quizás algunas no se usaran
  - **Exactly One**:Este caso solo expone una politica la cual debe de ser respetada
- Aserción de Politica: Comprende el nodo principal de la estructura a sobre a cual se basa cada una de las políticas a exponer
  - Tipo de Token: Identifica el tipo de token a usar, como lo puede ser una aserción SAML, un token alfanumérico, etc.
  - Integridad: define las partes firmadas y los elementos firmados de la aserción
  - Confidencialidad: Define las partes y los elementos encriptados de la aserción
  - Elementos Requeridos: Define los elementos y partes requeridas dentro de la aserción.
  - Security Binding Properties: define elementos tales como el uso de timestamp dentro de la política, los diferentes algoritmos utilizados dentro de esta en la encriptación de las diferentes partes y los tokens que la comprenden.

Cada uno de estos elementos deberá de ser expuesto por cada uno de los servicios involucrados de tal forma que agilice el proceso de selección de los mensajes, tanto lo emitidos como los recibidos.

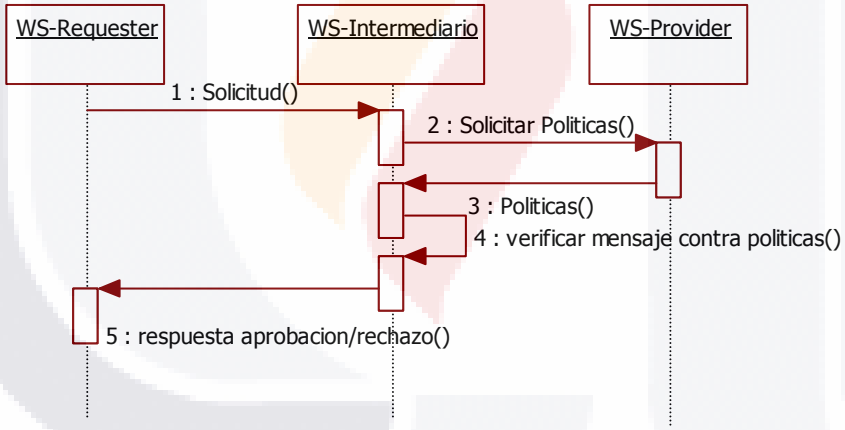


Figura 24: Proceso de validación de Políticas

El proceso de validación de políticas como se muestra en la Figura **24**: Proceso de validación de Políticas, se efectúa según los siguientes pasos:

1. El servicio web solicitante (WS-Requester) emite un mensaje con alguna petición, este mensaje puede o no ir de acuerdo con las políticas expuestas por el servicio web proveedor (WS-Provider)
2. El servicio web intermediario (WS-Intermediario), solicita las políticas a el servicio web Proveedor (WS-Provider).
3. El WS-Provider envía las sus políticas al WS-Intermediario

4. El WS-Intermediario, verifica las políticas recibidas contra las expresadas en el mensaje recibido y determina si estas son validas
5. En caso de ser validas, procesa la solicitud, en caso contrario puede o no emitir un mensaje de rechazo

3.5.4 Proveedor de Tokens

El proveedor de tokens es un servicio el cual únicamente tiene acceso a el servicio web autenticador, aquí se generan los tokens que se necesitan durante cada una de las sesiones, y se gestiona la caducidad de los mismos.

Este servicio se apoya de las especificaciones SAML y WS-Trust para la continua negociación, validación y expedición de tokens, entre él y el servicio Autenticador.

Los tokens generados son regresados al solicitante, envueltos en un mensaje SOAP, de tal manera que indique el tiempo de duración y fecha de creación, para que este pueda ser registrado y administrado tanto por el servicio Autenticador como por el Solicitante.

El proveedor de tokens, puede existir dentro del Servicio de Autenticación o puede estar externo a él, para lo cual este solo responderá a solicitudes que provengan del Autenticador.

Los tipos de tokens que se puede generar son diversos, para el desarrollo de este modelo, se tomara un token binario, el cual comprende un número creado en forma aleatoria.

CAPITULO IV:

4 PRUEBAS

En el capitulo anterior, se detallo la estructura y comportamiento del modelo de autenticación, ahora basados en este, se procederá comprobar que cumpla con su función para lo cual se deberán de hacer varios test, a fin de que el modelo compruebe que cumple el fin para el cual está diseñado.

Como lo menciona Balasbramaniam [38] debido a la naturaleza de los servicios web estos son como “cajas negras”, presentan un gran problema al momento de realizar las pruebas al respecto. Un consumidor no puede usar un análisis estático y técnicas de “caja blanca”, para probar la ausencia de vulnerabilidades.

Las siguientes pruebas están basadas en la guía de pruebas OWASP [24]

Escenarios:

Un escenario es una descripción parcial y concreta del comportamiento de un sistema en una determinada situación. Es una descripción parcial, porque solo necesita describir las características de las entidades involucradas, solo se describe aquello que está relacionado con un comportamiento particular del sistema analizado.

Un escenario es una situación en la cual mediante las relaciones existentes entre los participantes, el modelo desarrollado se puede aplicar, total o parcialmente, dentro de este desarrollan ciertos casos de prueba para verificar la validez de la implementación del modelo.

En este trabajo se tomara como escenario a una situación en la cual el modelo pueda ser probado, en esta se presentaran los participantes los cuales interactuaran entre ellos, en esta situación se tratara de acceder a la información que entre ellos están compartiendo o bien acceder a recursos a los que no se tiene acceso.

Para la prueba del modelo desarrollado anteriormente se plantea el siguiente escenario:

4.1 Escenario de prueba 1: Compartición de datos entre diversos clientes

Descripción:

Se tienen varias empresas las cuales requieren colaborar de forma conjunta mediante la compartición de información, estas necesitan tener la certidumbre de que los datos que proveen y aceptan, provienen de las entidades en las que ellos confían, por lo cual en el caso de una perdida de credenciales o mal uso de las mismas por parte de algún usuario, no afectara la credibilidad del origen de los datos, respecto a los involucrados.

Integrantes:

- Mitsubishi Motors
- Análisis Estadísticos SA de CV
- Fondo de Inversiones Xen

Requisitos:

- Desean compartir información entre ellos mediante servicios web
- Desean tener la certeza de que la fuente de datos es fidedigna así como a quien ellos proveen
- Cada uno de los involucrados maneja su información en tecnología diferente
- Se necesita autonomía en la decisión de compartir datos
- Necesitan seguridad en sus comunicaciones a través de la red
- Se necesita implementar autenticación tanto del usuario como de la entidad de software

Precondiciones:

- Todos los involucrados deberán de solicitar un WSID al autenticador e incluirlo en sus servicios web
- Usar el modelo desarrollado en la implementación de los servicios web
- A cada uno de los usuarios se le proporcionaran credenciales, con diferentes políticas de acceso a los recursos.
- Cada uno de los integrantes deberá de notificar los cambios realizados respecto a la autorización de acceso, al autenticador, mediante el modelo desarrollado

Para la verificación de este escenario se realizaran los siguientes test case [24]:

Tipo de prueba	Prueba de caja negra
Numero de caso de prueba	1
Nombre de caso prueba	Falsificación de token en servicio web
Descripción del caso de prueba	En esta prueba se verificara, el comportamiento del servicio web autenticador, ante la petición de autenticación de otro servicio web, el cual se presume es falso. Al servicio web que solicita ser autenticado, se le proveerá la llave publica utilizada en el modelo, de tal forma que los mensajes que emita sean validos, para esto el servicio web proveerá un token creado de forma aleatoria

Elementos a ser probados	
1	Servicio web autenticador
2	Proceso de autenticación en servicio web autenticador

Especificaciones	
Entradas	**Esperadas** **Salida/Resultado**
Mensaje : <<peticion+token+URIOrigen+Uridestino>llave simetrica>llave publica	Petición rechazada al no proveer el hash de un IDWS valido

Pasos	
1	Se creara un servicio web, el cual tomara el rol de servicio web solicitante respecto al modelo desarrollado
2	Este servicio web, conoce la ubicación del autenticador y procederá a enviar una solicitud de información, para la cual creara un token de forma aleatoria
3	La petición estará estructurada de la siguiente forma de acuerdo a el modelo <envelope> <header> <key> 774ujdlslfj30349955jfg453403943434343434iifr00opxj44n4434mn</key> </header> <body> <peticion> Autenticar </peticion> <token>**ksldne45mlsldj7dosdnjs88llsdm303485OSDJSDN1233449**</token> <URI>"http://www.aswinanand.com/sendsms.php" </URI> <URIDestino>http://batakmontong/solicitud.asp< URIDestino > </body> </envelope>
4	Para encriptar la petición, se generara una llave simétrica de forma aleatoria, con esta se encriptara el contenido **envelope** del mensaje, a su vez, se adjuntara la llave publica en el encabezado en el apartado key.
5	El apartado key se encriptara con la llave publica provista por el autenticador a todos los involucrados
6	El mensaje se envía al autenticador
7	El autenticador recibirá el mensaje o con su llave privada procederá a desencriptar el encabezado key, para obtener la llave privada con la que fue encriptada el resto del mensaje
8	Si el autenticador no puede obtener la llave privada se considera seguro

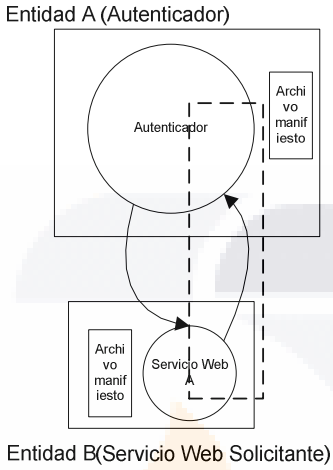
9	Al obtener el contenido del mensaje, en base a la correcta desencriptacion de los datos se verificara que token que se envía no haya caducado y además que corresponda con el URI asignado en base al URI de procedencia
10	Se verificara que mediante el token, se tenga autorizado el acceso al recurso que se solicita
Componentes del modelo a ser probados	
	
En la figura se muestran los componentes que intervienen en este caso prueba, los cuales son el autenticador y el servicio web solicitante del modelo desarrollado. Se verifica el comportamiento del autenticador ante la petición de autenticación con un token no valido, el cual es enviado desde Entidad B hacia entidad A	

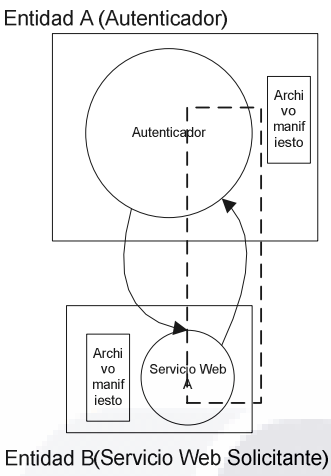
Tabla 5: Test Case: Token Tampering, Escenario1

Tipo de prueba	Prueba de caja negra
Numero de caso de prueba	2
Nombre de caso prueba	Falsificación de Id en servicio web
Descripción del caso de prueba	En esta prueba se verificara, el comportamiento del servicio web autenticador, ante la petición de autenticación de otro servicio web, el cual se presume es falso. Al servicio web que solicita ser autenticado, se le proveerá la llave publica utilizada en el modelo, de tal forma que los mensajes que emita sean validos, para esto el servicio web proveerá un token creado de forma aleatoria

Elementos a ser Probados	
1	Servicio web autenticador
2	Proceso de autenticación en servicio web

Especificaciones	
Entradas	**Esperadas** **Salida/Resultado**
Mensaje : <peticion+IDWS+URIOrigen >llave publica	Petición rechazada al no proveer un IDWS valido

Procedimiento	
1	Se creara un servicio web, el cual tomara el rol de servicio web solicitante respecto al modelo desarrollado
2	Este servicio web, conoce la ubicación del autenticador y procederá a enviar una solicitud de información, para la cual creara un IDWS de forma aleatoria y generara un resumen hash del Id generado
3	La petición estará estructurada de la siguiente forma de acuerdo a el modelo <envelope> <header> <key> 774ujdlslfj30349955jfg453403943434343434iifr00opxj44n4434mn</key> </header> <body> <peticion> Autenticar </peticion> <IDWS>**ksldne45mlsldj7dosdnjs88llsdm303485OSDJSDN1233449**</IdWS> <URI>"http://www.aswinanand.com/sendsms.php" </URI> </body> </envelope>
4	Para encriptar la petición, se generara una llave privada de forma aleatoria, con esta se encriptara el contenido **envelope** del mensaje, a su vez, se adjuntara la llave publica en el encabezado en el apartado key.
5	El autenticador recibirá el mensaje o con su llave privada procederá a desencriptar el encabezado key, para obtener la llave privada con la que fue encriptada el resto del mensaje.
6	Si el autenticador no puede obtener la llave privada se considera seguro
7	Al obtener el contenido del mensaje, en base a la correcta desencriptacion de los datos se verificara que el hash del IDWS que se envía sea correcto, además que la URI de la cual procede corresponde con el IDWS
8	Si el hash del IDWS no está registrado, o no corresponde con la URI, se deberá de rechazar la petición



En la figura se muestran los componentes que intervienen en el caso de prueba, estos son el autenticador y el servicio web solicitante respecto del modelo desarrollado.
Se verifica el comportamiento del autenticador (Entidad A) ante la petición de autenticación de la servicio web solicitante (Entidad B), la cual envía una petición valida, pero con IDWS falsificado

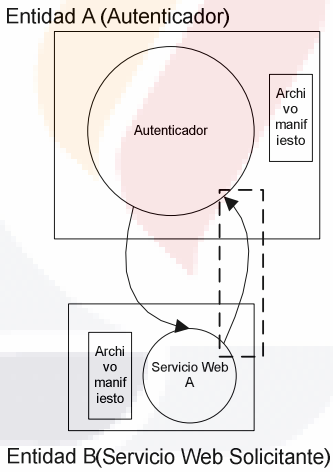
Tabla 6: Test Case 2, credentials tampering, Escenario 1

Tipo de prueba	Caja negra, repetición
Numero de caso de prueba	3
Nombre de caso prueba	Prueba de fortaleza de encriptación
Descripción del caso de prueba	En este caso de prueba se verificara la fortaleza de los mensajes enviados ante ataques de repetición, en la cual el objetivo es, en base a mensajes capturados, obtener el IDWS o el token de sesión

Elementos a ser Probados	
1	Mensajes entre autenticador y Servicio web

Especificaciones	
Entradas	**Esperadas Salidas/Resultados**
Mensaje<hash IdWs+URI+peticion>llave publica	Id de sesión o id de SW

Procedimiento	
1	Se emitirá un mensaje con la petición de autenticación entre el servicio web y al autenticador
2	Se capturara el mensaje mediante un sniffing como un proxy WebScaraB, o bien mediante **WireShark**
3	Se capturara una serie de estos mensajes y se buscaran patrones para intentar acceder al IDWS o bien el token de sesión enviado
4	Se verificara la estructura del mensaje SOAP buscando debilidades en la forma de su estructura y los datos que envía



En la figura se muestran los componentes involucrados en el caso de prueba, se capturaran los mensajes emitidos desde la entidad B, los cuales están representados por una flecha. Se verificar la fortaleza de las encriptación de los mensajes emitidos

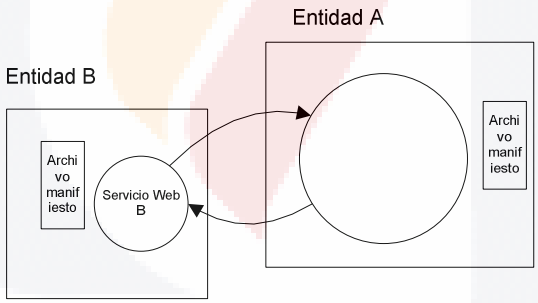
Tabla 7: Test Case 3: dictionary attack, brute force attack, escenario 1

Tipo de prueba	Prueba caja negra
Numero de caso de prueba	4
Nombre de caso prueba	Falsificación de autenticación
Descripción del caso de prueba	En esta prueba se verificara el comportamiento del servicio web, ante la petición de un autenticador no valido

Elementos a ser Probados	
1	Servicio web autenticador
2	Servicio web emisor/destinatario

Especificaciones	
Entradas	**Esperadas Salida/Resultado**
Mensaje : <petición+URIOrigen>llave privada	Petición rechazada al no proveer una encriptación valida, conforme a la llave publica y no poder acceder al contenido

Procedimiento	
1	Un servicio web, emitirá un mensaje hacia otro servicio web, el mensaje incluirá la petición de procesamiento y estará encriptado con una llave creada de forma aleatoria
2	El servicio web destinatario recibirá la petición
3	El servicio web destinatario deberá de desencriptar la información con su llave publica
4	Evaluar si el servicio web rechaza la petición en caso de que la llave privada no sea la correcta



En la figura los componentes involucrados son, el servicio web autenticador (Entidad A) y el servicio web proveedor (Entidad B), el cual recibe un mensaje firmado, pero no con llave privada del autenticador valido

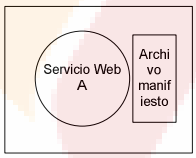
Tabla 8: Test Case 4: tampering credentials, escenario 1

Tipo de prueba	Prueba de almacenamiento
Numero de caso de prueba	5
Nombre de caso prueba	Almacenamiento de credenciales de autenticación
Descripción del caso de prueba	A fin de que un servicio web pueda comunicarse con el autenticador de forma continua, este debe de mantener un record de las URI´s a las cuales tiene acceso así como mantener el token de sesión que negocia al comenzar sus transacciones. Esta información debe de ser almacenadas en un lugar seguro y evitar que usuarios maliciosos busquen impersonar a otros al obtener acceso a estas. Se intentara acceder al token de sesión alojado dentro del servicio web

Elementos a ser Probados	
2	Servicio web emisor/destinatario

Especificaciones	
Entradas	**Esperadas Salida/Resultado**
Archivo tk.xml	Obtener el token de sesión y las URI´s alojandos en el archivo tk.xml

Procedimiento	
1	Se revisara el archivo tk.xml en cada servicio web de tal forma que se intente obtener el token residente dentro del archivo
2	Intentar acceder al contenido mediante pruebas de diccionario de datos
3	Intentar acceder al contenido mediante pruebas de fuerza bruta
4	Usar algoritmo RSA y TDES para intentar acceder al contenido del archivo



En este test case se probara el componte servicio web emisor/proveedor, concretamente el proceso de autenticación el cual contiene el archivo tk.xml que contiene el token negociado

Tabla 9: Test Case 5: storage test, escenario1

CAPITULO V:

5 RESULTADOS Y CONCLUSIONES

5.1 Resultados

El modelo se comprobó al compararse contra caso de estudio usando las pruebas planteadas, en la cuales se puede deducir la siguiente Tabla 10 respecto a cada uno de los casos de prueba.

Test Case	Objetivo	Resultado
1	Probar la falsificación de un token usado por el servicio web	Desde el punto de vista del modelo, la prueba resultara negativo, puesto si no se envía el hash del IDWS, la petición no se procesara. Además el token no estaría registrado en el autenticador por lo cual la autenticación fallaría en caso de poder acceder
2	Probar la falsificación de él ID del servicio web	Desde el punto de vista del modelo, la prueba resultara negativo, puesto que al generar un resumen hash de un elemento, este siempre es diferente, además no existe un patrón especifico de generación respecto a elementos parecidos, por lo cual el hash que envía resultara negativo
3	Probar de fortaleza de encriptación, en los mensajes intercambiados	Desde el punto de vista del modelo, la prueba resultara negativa, puesto que incluso al haber atrapado los paquetes intercambiados, es difícil que estos puedan ser decodificados, debido a que al momento de escribir estas líneas, algunos algoritmos para la generación de llaves simétricas han sido rotos pero solo por supercomputadoras, además, los algoritmos para el cálculo de llaves asimétricas permanecen sin romper, por lo cual es poco probable que la información enviada en cada mensaje pueda ser obtenida mediante este medio
4	Probar al respuesta ante una petición de autenticación falsa desde el lado del autenticador	Desde el punto de visa del modelo, el test resultara negativo, puesto que para poder acceder a la información de autenticación incluida en el mensaje recibido, esta deberá de forzosamente haber sido encriptado con la llave pública o viceversa, de otra forma es casi imposible acceder a la información contenida y la autenticación fallara
5	Probar la fortaleza de acceso al el archivo de almacenamiento de token y URI	Desde el punto de vista del modelo, el test resultara negativo, puesto que el archivo se encuentra encriptado en base al IDWS del servicio web, el cual esta embebido dentro de el, y es únicamente accesible mediante métodos específicos y para tareas especificas

Tabla 10: Resultados de Test

5.2 Conclusiones

Teniendo en cuenta el objetivo general y específicos de la investigación se puede concluir lo siguiente:

Los elementos necesarios para dentro de un modelo de autenticación comprenden:

- **Autenticador:** de servicios web, llega a tener la capacidad de verificar la identidad de la entidad que se lo solicite, para al realizar una solicitud de autenticación se deberá de verificar varios aspectos del servicio como lo es la privacidad, la integridad. Esto debido a que si llegase a fallar a privacidad alguna otra entidad podría tomar la identidad que está tratando de verificar y usarla en posteriores sesiones.
- **Solicitante**: Otro de los elementos a considerar debe de ser la entidad que solicita verificar (esto se mostro en la Figura 18), y debe de tener una característica única e intransferible, esta entidad se debe de adaptar al modelo implementado para poder realizar exitosamente la verificación.
- **Identidad Única**: La asignación de una entidad única a un servicio web, debido a su naturaleza de software, debe de estar basada en algo que sea intransferible, para lo cual se implemento la creación de un ID único el cual permanece embebido dentro del. Este ID además está registrado en el Autenticador.
- **Canal Seguro**: Un canal seguro es el medio por el cual las comunicaciones pueden ser transmitidas, este canal debe de estar protegido contra intrusos que intercepten o copien la información que está siendo transmitida.

En base lo anteriormente descrito se responde la pregunta "¿Que elementos deben de ser parte de un esquema de seguridad basado en la autenticación de servicios web?"

En el apartado Identidad de Servicios Web se describió la forma en la cual se asigno una identidad al un servicio web, esto mediante un factor de autenticación 2, el cual implica tanto "que se sabe", como el "que se tiene", con ello se cumple el objetivo "Especificar un modelo de seguridad para Servicios Web que autentifique a estos como entidades" y se responde a la pregunta "¿Cómo asignar una identidad única a un servicio web?"

En base a las pruebas planteadas, se probó la autenticación, la privacidad y la confianza entre los servicios web participantes:

El objetivo "Incorporar en el modelo una flexibilidad en el uso de credenciales", se cumple parcialmente puesto que solo se utilizo un tipo de credencial en el desarrollo.

En base los objetivos anteriormente mencionados, se cumple el objetivo general "Desarrollar un modelo para la autenticación de servicios web que implemente una correcta autenticación entre estos, permitiendo establecer una identidad a cada uno de ellos.", puesto que mediante el modelo desarrollado se define una identidad para cada uno de los servicios web, además mediante el modelo y los test case desarrollados se verifica que la autenticación se realiza en forma correcta.

5.3 Contribuciones

La presente investigación presenta las siguientes contribuciones:

- Presenta un modelo para permitir establecer una relación de confianza entre servicios web clientes y proveedores mediante el uso de un Autenticador central y PKI.
- Provee una guía para su implantación a un nivel conceptual sin estar atada a ninguna tecnología en específico.
- Menciona un método para asignar a un servicio web una identidad y la forma en la cual esta es verificada mediante el modelo desarrollado.
- Describe varios casos de prueba para verificar los elementos de autenticación usados en el modelo desarrollado.

Ahora bien para el cumplimiento de los objetivos particulares se llevaron de la siguiente manera

#	Objetivos Particulares	Forma de resolverlo
1	Especificar un modelo de seguridad para servicios web que autentifique a estos como entidades	Para cumplimiento de este modelo, se tomo en cuenta varios trabajos que previamente habían sido desarrollados, además se incorporo el uso de diversos estándares utilizados para servicios web, y una aplicación de llaves simétricas y asimétricas para garantizar la autenticación y la identidad de los participantes.
2	Facultar al modelo desarrollado para proveer a los servicios web involucrados, una identidad independiente de la del usuario que haga uso de el	El modelo desarrollado implemento una asignación de Id único para cada uno de los involucrados el cual especifico que este debería de ir embebido en el código, además de especificar los procesos necesarios para que este fuera validado dentro del modelo
3	Probar la autenticación de entidades a través del modelo desarrollado	Se desarrollaron test cases dentro de un caso de estudio, los cuales están enfocados a la verificación, de diversos elementos que integran el modelo, en los cuales se verifica tanto la integridad de la información, como la verificación de la autenticación de la identidad asignada a los servicios web participantes
4	Incorporar en el modelo una flexibilidad en el uso de credenciales	Se agrego un Id único a cada uno de los servicios web involucrados en el modelo

Tabla 11: Resolución de Objetivos

5.4 Aportaciones de la Estancia de Investigación

Dentro de la estancia realizada en la Florida Atlantic University (FAU), ubicada en Boca Raton, Florida, Estados Unidos, se realizo un articulo basado en patrones, más concretamente en Misuse Patterns, aplicados a la autenticación de servicios web, el desarrollo de este articulo así como los diferentes puntos de vista que se hicieron, permitió fortalecer el trabajo de tesis que se venía haciendo.

El desarrollo del articulo "Misuse Patterns for Authentication in Web Services" especifica la forma en la cual 3 diferentes ataques son llevados a cabo contra servicios web, específicamente en el área de autenticación, en el artículo se especifica cómo es que se realiza desde el punto de vista del atacante, cuales son los daños que podría generar, las posibles soluciones para el problema y cuales son pruebas que se pueden obtener en un examen forense después de realizar el ataque, todo lo anteriormente descrito en forma de patrón.

5.5 Experiencia Individual

Dentro de la experiencia obtenida durante el trascurso del posgrado, se obtuvo varias cosas, tanto a nivel académico como personal, en el nivel académico, destaca el uso de la tecnología y las practicas propias que comprende el área de ingeniería de software, todo esto a través de las diferentes materias con las que se conto así como la metodología de la investigación.

En el aspecto personal se obtuvo una gran experiencia, en la cual destaca el fomento por la investigación, me mostraron fronteras mas lejanas de las que tenia contempladas, un mundo mayor en el cual dentro de un tema existe una gran cantidad de temas a tratar.

5.6 Trabajo Futuros

Como trabajo futuro se pretende extender el modelo, hacia entornos federados, en el cual se pueda implementar el SSO (Simple Sign On, Firma única).

Probar el modelo desarrollado mediante un desarrollo de software.

Ampliar el uso de la flexibilidad de credenciales al modelo desarrollado.

Trabajar con Misuse patters para servicios web en las diversas áreas existentes, como lo es:

- La integridad en el intercambio de la información
- Autorización y acceso a la información
- Ingeniería social

Además de especificar patrones de diseño de servicios web para acceso seguro a aplicaciones en "cloud computing" (computo en nueb)

CAPITULO VI:

REFERENCIAS

[1] T. Erl, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall, 2004.

[2] A. C. Machado and C. A. G. Ferraz, "Guidelines for Performance Evaluation of Web Services," 2005.

[3] T. Grob, "Security Analysis of the SAML Single Sign-on Browser/Artifact Profile," *IEE*, 2003.

[4] M. Mora T, "Descripcion del metodo de investigacion conceptual," UAA, 2003.

[5] R. Mendoza Gonzalez, M. Vargas Martin, J. Muñoz Arteaga, F. Alvarez Rodriguez, and C. A. Ochoa Ortiz Zezzatti, "Web Service Security Specification based on Usability Criteria and Pattern Approach," *JOURNAL OF COMPUTERS*, vol. IV, no. 8, 2009.

[6] R. Oppliger and S. Gajek, "Effective Protection Against Phishing and Web Spoofing," *SpringerLink*, Sep. 2005.

[7] Web Services Interoperabily Organization. (2005, Jul.) Web Services Interoperabily . [Online]. http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0-20050507.doc

[8] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*. Wiley, 2006.

[9] H. Caudel Garcia, J. Muñoz Lopez, J. Munoz Arteaga, and F. Alvarez Lopez, "Especificación de Esquema de Seguridad Autentificación de Servicios Web Heterogéneos," in *Simposio en Ingenieria de Software*, Mexico, D.F, Jul. 2009.

[10] N. Arshad, "Identity Protection Factor (IPF)," *ACM*, 2008.

[11] B. Karthikeyan, R. Corin, C. Fournet, and A. D. Gordon, "Secure Sessions for Web Services," *ACM*, 2006.

[12] R. Hernandez Sampieri, C. Fernandez Collado, and P. Baptista Lucio, *Metodologia de la Investigacion*, Cuarta Edicion ed. Mexico, Mexico: Mc GrawHill, 2008.

[13] L. Hohmann, *Beyond Software Architecture*, A. Wesley, Ed..

[14] F. Tartanoglu, V. Issarny, A. Romanovsky, and N. Levy, "Dependability in the Web Services Architecture," *ICSE Workshop on Architecting Dependable Systems SpringerLynk*, 2002.

[15] F. Curbera, W. A. Nagy, and S. Weerawarana, "Web Services: Why and How," *ACM*, 2001.

[16] E. M. Maximilien, A. Raanabahu, and R. Engehausen, "IBM Altocumulos: A Cross-Cloud Middleware and Platform," *24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, 2009.

[17] E. M. Maximilien, A. Ranabahu, R. Engehausen, and L. C. Anderson, "Toward Cloud-Agnostic MiddleWares," *24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, 2009.

[18] B. Hartman, D. J. Flinn, K. Beznosov, and S. Kawamoto, *Mastering Web Services Security*. Whiley Publishing, 2003.

[19] E. Al-Masri and Q. H. Mahmoud, "Investigating Web Services On The World Wide Web," *ACM*, 2008.

[20] K. Bierhoff, M. Grechanik, and E. S. Liongosaro, "Architectural Mismatch in Service-Oriented Architectures," *ACM*, 2007.

[21] J. Newmarch, "A critique of Web Services," 2004.

[22] OASIS. (2009, Apr.) UDDI Version 3.02 API Specification. [Online]. http://www.oasis-open.org/specs/#uddiv3.0.2

[23] N. N. Cova Suazo, "Orquestación de servicios Web orientada a aspectos," Centro de investigación y de Estudios Avanzados del Instituto Politécnico Nacional Tesis, 2005.

[24] M. Jensen, N. Grushchaka, R. Herkenhoner, and N. Luttenberger, "SOA and Web Services: New Technologies, New Standars -New Attacks," *IEEE*, Sep. 2007.

[25] K. Ramarao and C. Prasad, *SOA Security*. Manning, 2008.

[26] A. Nash, W. Duane, C. Joseph, and D. Brink, *PKI Infraestructura de Claves Publicas*. McGraw-Hill, 2002.

[27] OWASP. OWASP. [Online]. http://www.owasp.org/

[28] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1995.

[29] D. M. Kienzle and M. C. Elder, "Final Technical Report: Security Patterns for web Application Development".

[30] H. Park and S. Redford, "Client Certificate and IP Address Based Multi-factor Authentication for J2EE Web Applications," *ACM*, pp. 167-174, 2007.

[31] T. Oda, G. Wurster, P. D. V. Oorschot, and A. Somanaji, "SOMA: Mutual Approval for Included Content in Web Pages," *ACM*, 2008.

[32] J. White, "Security in a Web-Services World: A Proposed Architecture and RoadMap," 2002Abril.

[33] Oasis. OASIS. [Online]. http://www.oasis-open.org/specs/index.php#samlv2.0

[34] OASIS. (2007, Mar.) WS Trust 1.3 Standar. [Online]. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

[35] B. Galbraith, et al., *Profesional Web Services Security*. United States: Wrox, 2002.

[36] C. Bryce, P. Courdec, J.-M. Seigneur, and V. Cahill, "Implementation of the SECURE Trust Engine," *Lecture Notes in Computer Science*, vol. 3477, 2005.

[37] W3C. (2007) Web Services Policy 1.5 - Framework. [Online]. http://www.w3.org/TR/ws-policy/

[38] OASIS. (2006, Feb.) Web Services Security. [Online]. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss#overview

[39] O. Ajaj and E. B. Fernandez, "A pattern for the WS-Policy standard," in *8th Latin American Conference on Pattern Languages of Programs*, Bahia, Brazil, 2010.

[40] S. Balasumramaniam, L. Grace A, E. Morris, S. Soumya, and D. B. Smith, "Challenges for Assuring Quality of Service in a Service-Oriented Environment," *ACM*, 2009.

[41] IBM. (2009, Apr.) Web Services architecture overview. [Online]. http://www.ibm.com/developerworks/webservices/library/w-ovr

[42] C. Peltz, "Web Services Orchestration and choreography," *IEEE*, 2003.

[43] G. Chafle, S. Chandra, V. Mann, and M. G. Nanda, "Decentralized Orchestration of Composite Web Services," *ACM*, 2004.

[44] S. Fugkeaw, P. Manpanpanich, and S. Juntapremjitt, "Adding SAML to Two-Factor Authentication and Single Sign-On Model for Dynamic Access Control," *IEEE*, 2007.

[45] D. J. Lutz, "Federation Pyments using SAML Tokens with Trusted Plataform Modules," *IEEE*, 2007.

[46] M. Ates, C. Gravier, J. Lardon, J. Fayolle, and B. Sauviac, "Interoperability between heterogeneous federation architectures:Illustration with SAML and WS-Federation".

[47] IBM. (09, Jun.) IBM Especificacion WS-Federation Specification 1.1. [Online]. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-FederationSpec05282007.pdf

[48] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *Mis Quartertly*, vol. 28, no. 1, 2004.

[49] W3C. (2002,) XML Encryption. [Online]. http://www.w3.org/TR/xmlenc-core/

[50] W3C. (2008, Jun.) XML Signature. [Online]. http://www.w3.org/TR/xmldsig-core/

[51] OASIS. (2007, Mar.) WS Secure Conversation. [Online]. http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html

[52] SIFT, "A Web Services Security Testing Framework," Nov. 2006.

[53] K. Bhargavan, C. Fournet, and A. D. Gordon, "Verifing Policy-Based Web Services Security," *ACM*, 2008.

[54] OASIS. Web Service Policy. [Online]. http://www.w3.org/2002/ws/policy/

[55] L. J. Camp, "Identity, Authentication, and Identifiers in Digital Government," 2003.

[56] OWASP. (2010) OWASP Pruebas de Autenticación. [Online]. http://www.owasp.org/

[57] C. Wong and D. Grzleak. (2009, Nov.) SIFT Information Security Services. [Online]. http://www.sift.com.au/assets/downloads/SIFT-Web-Services-Security-Testing-Framework-v1-00.pdf

[58] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley, 1996.

[59] E. B. Fernandez, J. C. Pelaez, and M. Larrondo- Petrie, "Attack Patterns, A New Forensic And Design Tool," in *IFIP International Federation for Information Processing*, P. Craiger and S. Shenoi, Eds. SpringerLynk Boston, 2007, vol. 247, ch. 24, pp. 345-357.

[60] E. B. Fernandez, Y. Nobukazu, and H. Washizaki, "Modeling Misuse Patterns," *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pp. 566-571, Mar. 2009.

[61] J. C. Pelaez, E. B. Fernandez, L.-P. Marie M, and C. Wieser, "Misuse patterns in VoIP," in *14th Conference on Pattern Languages of Programs*, Illinois, 2007.

[62] E. B. Fernandez, N. Yoshioka, and H. Washizaki, "A Worm misuse pattern," in *AsianPLoP 2010: 1st Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, 2010.

[63] O. Ajaj and E. B. Fernandez, "A pattern for the WS-Trust standard for web services," in *AsianPLoP 2010: 1st Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, 2010.

[64] A. Barbir, C. Hobbs, E. Bertino, F. Hirsch, and L. Martino, "Challenges of testing Web Services and security in SOA Implementations," in *Test Analysis of web Services*. SpringerLink, Sep. 2007, pp. 395-440.

[65] O. Ajaj and E. B. Fernandez, "A pattern for the WS-Policy standard," in *1st Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, 2010.

[66] P. Morrison and E. B. Fernandez, "The Credential Pattern," in *2006 conference on Pattern languages of programs*, Portland Oregon, 2006.

[67] M. Jensen, N. Gruschka, and R. Herkenhoner, "A Survey of Attacks on Web Services: Classification and countermeasures," *Computer Science - Research and Development*, vol. 24, no. 4, pp. 185-197, May 2009.

[68] K. Hashizume and E. B. Fernandez, "PATTERN LANGUAGES OF PROGRAMS CONFERENCE 2009," in *PATTERN LANGUAGES OF PROGRAMS CONFERENCE 2009*, Chicago Illinois, 2009.

[69] E. Bertino and L. D. Martino, "A Service-Oriented Approach to Security--Concepts and Issues," in *11th IEEE International Workshop on Future Trends of Distributed Computing Systems*, Sedona, AZ, USA, Mar. 2007.

[70] A. Singhal, T. Winograd, and K. Scarfone. (2007,) http://www.nist.gov/. [Online]. http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf

[71] B. Hartman, D. J. Flinn, K. Beznosov, and S. Kawamoto, *Mastering Web Services Security*, K. A. Malm, Ed. Indianapolis, Indiana, EUA: Whiley Publishing, 2003.

[72] H. Washizaki, E. B. Fernandez, K. Marayuma, and A. Kubo, "Improving the Classification of Security Patterns," in *20th International Workshop on Database and Expert Systems Application*, Linz, Austria, 2009, pp. 165-170.

ANEXOS

A. Glosario

Ataque: Cualquier acción que intente violar la integridad, confidencialidad o disponibilidad de un sistema

Certificado: También conocido como certificado digital. Un certificado es un documento electrónico unido con algunas piezas de información, tales como la identidad y la clave publica de un usuario. Una autoridad de certificación (CA) suele expedir certificados, pero una empresa, un gobierno o alguna otra entidad pueden firmar su propio certificado. Este certificado autofirmado es el certificado raíz para la entidad y se usa para firmar certificados subordinados.

Certificado X.509: información digital firmada por una autoridad de certificado; un certificado X.509 contiene información relacionada con el sujeto que enlaza a un usuario especifico con su clave publica. El certificado X.509 contiene, por ejemplo, el nombre distinguido del sujeto, la clave publica RSA, el nombre del expedidor y la firma digital.

Cookie: Es un fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Estas pueden ser borradas, aceptadas o bloqueadas según desee.

Crawling: Técnica que permite inspeccionar las carpetas, documentos dentro de un servidor.

Credencial: Una credencial es un registro que contiene la información de autenticación (credenciales) necesaria para conectarse a un recurso o sistema.

DNS: Domain Name Server (Sistema de Nombre de Dominio). Es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a internet o a una red privada. Este sistema asocia información variada con nombres de dominios asignado a cada uno de los participantes. Su función más importante, es traducir (resolver) nombres inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente.

HTTPS: (HyperText Transfer Protocol Secure) Variante segura del protocolo HTTP. Bajo HTTPS, la conexión entre cliente y servidor se cifra usando un nivel de socket seguro (SSL).

IDL: Interface Definition Languaje (Lenguaje de definición de Interface). Es un lenguaje de especificación de interfaces que se utiliza en software de computación distribuida. Ofrece la sintaxis necesaria para definir los procedimientos o métodos que queremos invocar remotamente. Una vez

tengamos esta interfaz creada deberemos pasarla por un compilador de interfaces que generará el proxy o stub cliente y el skeleton o stub servidor.

IP: es una etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz (elemento de comunicación/conexión) de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red del protocolo TCP/IP. A través de Internet, los ordenadores se conectan entre sí mediante sus respectivas direcciones IP.

Kerberos: Sistema de autenticación y autorización cliente-servidor desarrollado por el MIT a finales de la década de los 70's. Usa cifrado simétrico.

Cifrado Asimétrico: un método criptográfico que usa una clave para cifrar un mensaje, y una clave diferente para descifrarlo. Es el fundamento de la Infraestrutura de Claves Publicas PKI

Llave Pública: El cifrado asimétrico o PKI, la clave de cifrado que se presenta públicamente para comunicarse con seguridad con el poseedor de la llave privada. La clave publica se puede usar para descifrar un mensaje creado por a clave privada del usuario, la cual brinda la prueba de la creación de un mensaje autentico; la clave publica del usuario se puede usar para cifrar un mensaje privado solo para ese receptor.

Llave Privada: El cifrado asimétrico o PKI, la clave de cifrado confidencial que mantiene el privado el usuario. La llave privada se puede utilizar para cifrar un mensaje, lo cual suministra una prueba de la creación autentica de un mensaje cuando se descifra con la llave publica correspondiente; debido a que la clave privada también se puede usar para descifrar un mensaje, protege la privacidad de la comunicación que envían otros, quienes usan la clave publica para descifrar mensajes.

Middle Man: (Hombre en medio) El ataque de un pirata informático que se posiciona entre dos partes sin sospecha alguna, durante una sesión de comunicación.

Pishing: Es un tipo de delito encuadrado dentro del ámbito de las estafas cibernéticas, y que se comete mediante el uso de un tipo de ingeniería social caracterizado por intentar adquirir información confidencial de forma fraudulenta (como puede ser una contraseña o información detallada sobre tarjetas de crédito u otra información bancaria). El estafador, conocido como phisher, se hace pasar por una persona o empresa de confianza en una aparente comunicación oficial electrónica, por lo común un correo electrónico, o algún sistema de mensajería instantánea1 o incluso utilizando también llamadas telefónicas, con la intención de robar su identidad o sus datos.

RSA: Uno de los primeros criptosistemas de clave publica, patentado en 1983. RSA es un criptosistema de clave pública, en base en el problema de factorización. RSA corresponde a las iniciales de Rivest, Shamir y Adleman, los creadores del criptosistema de clave publica de RSA y los fundadores de RSA Data Security.

Servlet: Un servlet es un objeto que se ejecuta en un servidor o contenedor JEE, especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente HTML. Otras

opciones que permiten generar contenido dinámico son los lenguajes ASP, PHP, JSP (un caso especial de servlet), Ruby y Python. Forman parte de JEE (Java Enterprise Edition), que es una ampliación de JSE (Java Standard Edition).

Sniffing: Se trata de dispositivos y técnicas que permiten al atacante "escuchar" las diversas comunicaciones que se establecen entre ordenadores a través de una red (física o inalámbrica) sin necesidad de acceder física ni virtualmente a su ordenador.

Spoofing: Es el uso de técnicas de suplantación de identidad en la red generalmente con usos maliciosos.

SSL: (Nivel se socket seguro) Secure Socket Layer, un estándar abierto propuesto por Netscape Communications para ofrecer servicios seguros (Cifrados y Autenticados) en la WWW, a través de Internet. SSL usa cifrado de clave publica RSA.

Token: Es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación o esquema de uso.

URI: Uniform Resource Identifier, Un URI es una cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema.

Vulnerabilidad: Es la posibilidad de que la integridad, confidencialidad o disponibilidad de un sistema sea vulnerada.

B. Reporte de estancia de investigación

Durante la estancia de investigación que comprendió del 28 de Mayo al 31 de julio del 2010, en Florida Atlantic University (FAU), ubicada en Boca Raton, Florida, Estados Unidos, bajo la tutela del Dr. Eduardo B. Fernandez y la Dra. Marie Petrie, respecto a la parte americana, y del Dr. Jaime Muñoz Arteaga y Dr. Ricardo Mendoza Gonzales en la parte mexicana, se desarrollo un paper basado en "Misuse Patterns", estos patrones estuvieron enfocados a los servicios web, concretamente en el área de autenticación. Por esta situación, el artículo fue escrito en el idioma ingles.

El objetivo de este paper es mostrar algunas vulnerabilidades presentes dentro de los servicios web, en la cual una de estas variantes, es atacada en este trabajo de investigación, concretamente el caso "principal spoofing".

Cabe destacar que para la realización de esta estancia se conto con el imprescindible apoyo de Conacyt, la Universidad Autónoma de Aguascalientes y Florida Atlantic University, tanto a nivel económico como académico.

El artículo desarrollado se muestra a continuación:

Misuse Patterns for Authentication in Web Services

Abstract: We present here three generic attacks against authentication in web services. A misuse pattern describes the actions performed by an attacker to accomplish his goals. We describe these attacks using misuse patterns. Specifically, we describe Principal Spoofing, Forged Claims, and Man-In-The-Middle attacks.

Keywords: **Authentication, Misuse Patterns, Security Patterns, Web Services Security**

1 Introduction

Authentication is a fundamental security mechanism to verify identity claims of users or other systems. It determines if someone or something is who its claims to be. An identity [1] is a set of permanent or long lived attributes associated with an entity.

Web services can communicate with other web services or applications and share information. However, there are some web services which contain sensitive information, not intended for public access. In that case the resources from some web services can only be accessed by authorized users, who must be first authenticated.

Authentication is a complex process that could generate problems, for example it is possible to infer some basic issues through the authentication, test proposed by OWASP (Open Web Application Security Project). Some related authentication attacks are shown in the next sections.

Many of the features that make web services attractive, including greater accessibility to the data, dynamic application-to-application communication, and relative autonomy (lack of human intervention) may brings serious security challenges. Some of the problems that web services could have in the authentication area are the following: (these were derived by analyzing the OWASP test set of authentication [2])

- Principal Spoofing
- Forged Claims
- Man in the Middle

In this work we propose three misuse patterns from authentication in web services. These patterns are useful for the designers and developers of web services, who can avoid these situations. If they have these problems, these patterns, can be a guide to know what happened and to correct the corresponding vulnerabilities that lead the attack.

The rest of the paper is organized as follows: Section 2 provides some background. Section 3 shows details of the structure of our patterns. Section 4 presents the patterns. Section 5 gives some conclusions and future work.

2 Background

OWASP is an open source project, which is focused on improving applications and web services security. One of this objectives is the OWASP Testing Guide [2], focused on application security testing procedures. This testing guide shows a summary, description, and test with use cases about authentication problems. There is also a framework [3] of tests specifically from web services.

A pattern is well understood solution to a recurrent problem and implies best practices about its topic [4] Misuse Patterns [5],[6] describe from the point of view of the attacker, how a type of attack is performed (what system units it uses and how), proposes ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and helps analyze the attack once it has happened by indicating where we can find forensics data as well as what type of data.

Additionally, misuse pattern provide an insight into how an attack is carried out, as well as the extent to which the attack affects a web service (access to resources, loss of trust, or others).

The existing collections of misuse patterns [7],[8] are oriented to help designers to know the point of view of the attacker. In this work we try to understand three problems of authentication that; principal spoofing, man in the middle, and forged claims.

3 Approach

The specification of each misuse pattern is written from the point of view of the attacker, this pattern format was proposed in [4], [5] and will includes the following:

Name: Name of the pattern.
Intent: A short description of the intended purpose of the pattern (which problem it solves for an attacker).
Context: This section describes the general environment, including the conditions under which the attack may occur.
Problem: Definition of the goal of the attack pattern. From a hacker's perspective, the problem is how to find a way to attack the system.
Solution: Description of the solution of the hacker´s problem. This will be represented with UML diagrams to show the system before and during the attack. Sequence diagrams show the exchange of messages needed to accomplish the attack.
Known Uses: Specific incidents where this attack occurred. Details of past attacks are useful to decide where to look for evidence and how stop the attack.
Consequences: Discussing the benefits and drawbacks of an attack pattern from the attacker´s viewpoint.
Countermeasures: Description of the security measures necessary in order to stop, mitigate, or trace this type of attack. This implies an enumeration of which security patterns are effective against the attack.
Related Patterns: Discussion of other attack patterns with different objectives but performed in a similar way or with similar objectives but performed in a different way.

4 Authentication Misuse Patterns

4.1 Principal Spoofing

Name
Principal Spoofing
Intent
A Principal Spoofing attack tries to impersonate the identity of a user, using the user´s credentials to make requests on their name, with the intention of accessing a specific system; in this case, this system is based on web services.
Context
Through use of web services, enterprises can exchange data. They publish a WSDL file with functionalities and security policies to define access to the web service.
The requester must create a web service which should adapt its methods and policies to the web service provider. Security can be implemented at the level of messages, each user is given credentials to access and the responsibility of protect them from others users. The web service can be protected or not using WS-Security and WS-Policies.
Problem
How we can effectively perform a Principal Spoofing attack against web services so we can access the information of another user?
Solution
There are situations in which the users need to access a system based on web services. This kind of system must be protected with the implementation of several web services security standards, e.g. WS-Security [9] to protect messages sent, or WS-Policy [10] to verify the security requirements of the web service. These security standards use the credentials of the user to protect communication between web services, such as the signing and encryption of messages. If the attacker manages to get valid credentials, he can communicate with other web services. This type of vulnerability is mentioned in [11] and is named "Principal Spoofing"; in this attack, the user does not know that his credentials were stolen, until the damage is done (such as alteration of information, access to his resources, alteration of privilege, even attaching malicious code that could be harmful to the server).
When the attacker has the credentials, he can use the WSDL (Web Service Definition Language) file in order to discover the security policies of the web services and create a valid message.

This situation is conditional on the attacker obtaining a user´s credentials, the valid user not knowing of the theft and not having reported the theft.

This attack can be carried out, taking into account the following:

- If the communications between web services are not protected, sniffer software could intercept packages while travelling in the network, and obtain the credentials carried in the messages.
- Using social engineering to get the credentials of the user.
- The system could be using inadequate security policies; in this way is possible to cheat the system.
- Another vulnerability may be that the system does not verify the origin of the request.

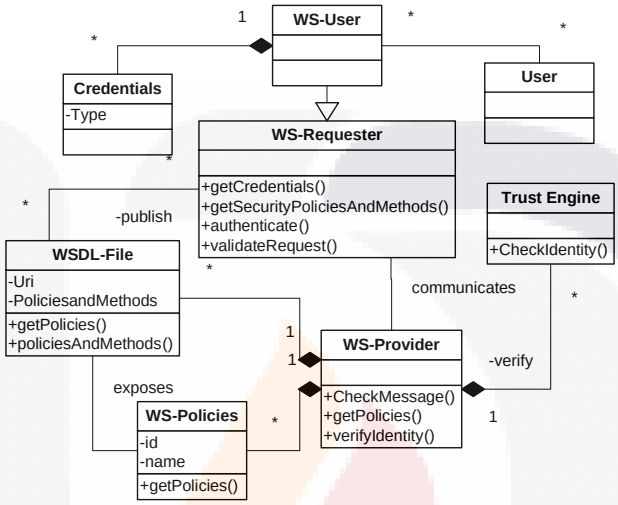


Figure 1: Web Services Diagram

The common relationship between web services is showed in the Figure 1, its shows the class diagram. The **WS-User** has **Credentials** to communicate with another web service.

The **WS-User** has one or more **User**s, and every one can have **Credentials** that allow it communicates. The **WS-Requester** can be a **WS-User** o another kind of web service. The **WS-Provider** has several **WS-Policies**, and this are exposed in the **WSDL-File**, on this way, the **WS-Requester** can be created, based on the **WS-Policies** showed in the **WSDL-File**. When the **WS-Requester** tries to communicate with the **WS-Provider**, the **WS-Provider** has to check the request with the **Trust Engine.**

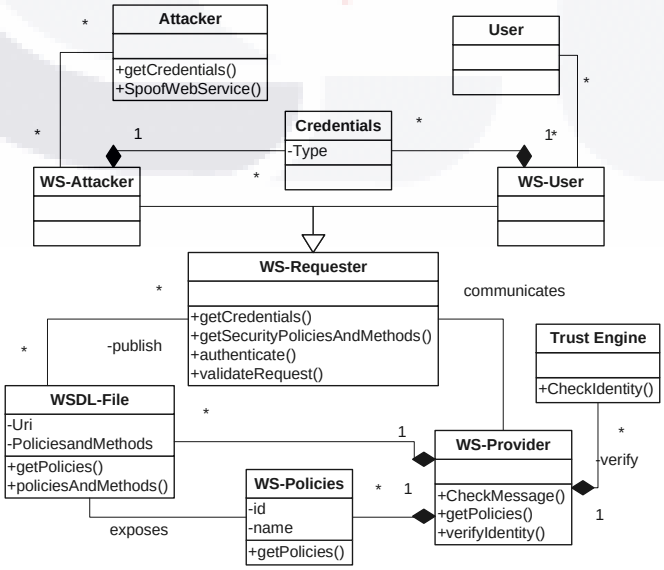


Figure 2: Misuse Pattern: Spoofing Principal

To make possible the Principal Spoofing attack, the user must have the user´s credentials as showed in the Figure 2. This diagram shows the structure of a Principal Spoofing Attack, the class **WS-Attacker** and **Attacker** are the elements introduced by the attacker.

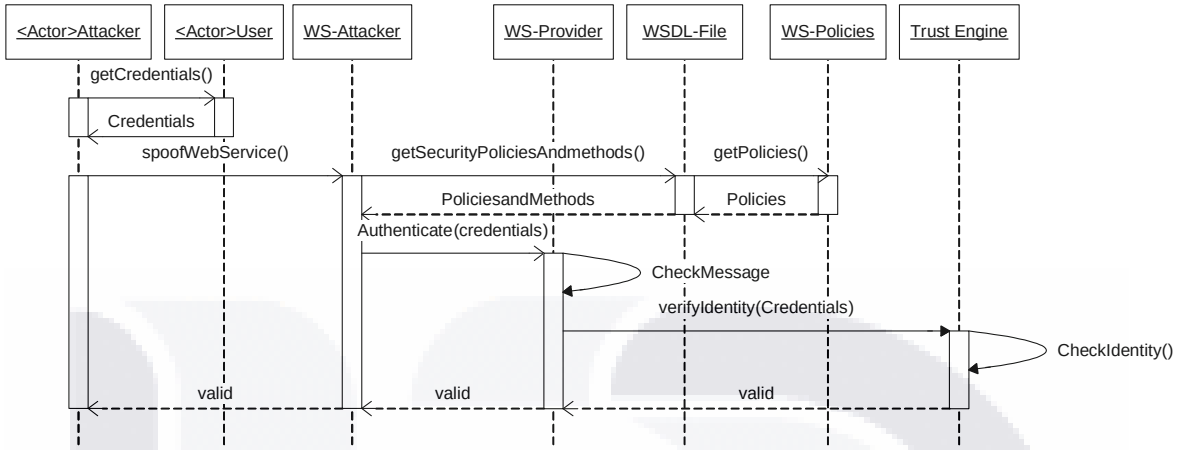


Figure 3: Principal Spoofing Sequence Diagram

The sequence diagram of the in the Figure 3 shows the sequence of steps to perform a Principal Spoofing Attack.

An **Attacker** can obtain credentials of a valid **User**; this can happen in several ways like social engineering, pissing, even if the communications not are protected, the attacker can capture the packages of communication with sniffing software as WireShark or WebScarab and analyse them to obtain the credentials. When the attacker has the credentials, he can access to the **WSDL-File** of the **WS-Provider**, in this file the **WS-Policies** and methods to access to the web service are published. The **WSDL-File** can be published in the UDDI service or can be access through techniques such "crawling". With the information obtained from the **WSDL-File** and the credentials of the **User**, the **Attacker** can create a Web Service to make valid request to the **WS-Provider**.

When the attacker has the web service, he can make a request using the credentials of the user. The **WS-Provider** receives the request and checks the message against the policies of the web service, if the message is correct, the WS-Provider, sends the credentials to the **Trust Engine** to check the identity of the user. Because the credentials are valid, the authentication will be correct.

When security policies are known, will be able to develop another web service with other behaviour, according to the user needs. User credentials will be used as identification to access the system´s functionality.

Consequences

The success of this attack implies:

- The attacker can access to information sensitive from the valid user
- Malicious code can be attached to the messages
- The attacker can read copy or modify information or make transactions in the name of the valid user

Countermeasures and Forensics

Human behaviour cannot be controlled, so cannot be avoided that the human being deceived into getting their credentials. Also can be victim of a theft of his credentials through an oversight.

The attack can be mitigated through the following measures:

1. Encrypt communications between web services; in order to be impossible the access to the information when this is intercepted while travels in the web, regardless the use of sniffing software, this can be done using security standards as XML Encryption and WS Security.
2. For each of the web services that are developed:
   a. Generate an intern credential, can be an Id or some kind of token. This credential will be embedded inside of the code.

b. Web service provider must publish a policy in which this credential requests for each transaction.
c. Also if the communication is done using a PKI Infrastructure, the credential must be sent encrypted using the public key provided by web services provider. With this measure even if the message is intercepted by a sniffer software, it not be able to access content of message, because it is encrypted with a asymmetric key must have the private key to decrypt.

Likewise the next forensic evidences are possible:
With logs on the Web Service provider can identify which user was affected, and what were the actions performed, before he received the theft report.
Check the precedence of the IP from the attack, and check with the list of the valid point access, but the attacker also can spoof the IP address.

Related Patterns

WS-Trust: This pattern helps to establish the relationships that should arise between Web services, to share their credentials [9].

4.2 Forged Claims

Name
Web Services Forged Request.
Intent
Forged claims attack [10], try access to system functionalities through the use of forged credentials, or tokens made in a random way. The objective is:
Access to information and resources protected. Realize some action in the name of another person, these actions can damage economically to the valid user, also can modify or delete information; this actions will be in favor of the attacker.
Context
In an environment of web services, several kinds of credentials are used by the users, like certificates, tokens or an Id with its password, when begins communication between two parts, this can be protected o not for someone of different standards like WS-Security or WS-Trust.
This kind of attack happen between the communications of web services could be because those do not have protected his communications or the authentication system is not properly implemented, whereby allow the access to resources, only because the request presents his credentials, and the web service of authentication does not verify the authority that issued.
Problem
How effectively perform a forged claims attack and access to resources of the system?
How access to delicate information and resources from a specific web service? How can access to a specific web service if do not have valid credentials?
Solution

Self-generated certificates can be created and presented to the authentication service, or create random tokens and presents those to system, and use tokens obtained from previously messages through techniques like sniffing and try to access to the system with those.

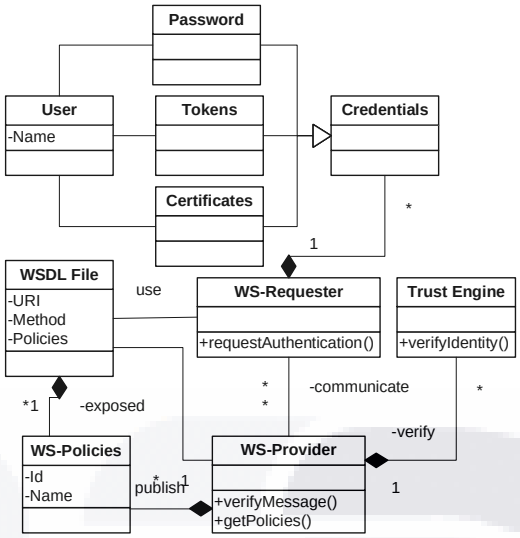


Figure 4: Web services diagram 2

In the figure Figure 4 shows the class diagram from a communication between **WS-Provider** and **WS-Requester** using Credentials. An **User** can have several types of Credentials to communicate with a Web Service, this **Credentials** can be **Passwords**, **Tokens** or **Certificates**. The **WS-Requester** is created based on the **WSDL-File**, this file expose the **WS-Policies** from the **WS-Provider**, to can communicate with each. When the **User** tries to communicate with another web service, he need to use the **WS-Requester**, this web service need the Credentials of the **User**, to make a requestAuthentication. When the request is made, the **WS-Provider** check the received message against the **WS-Policies**, if the message is correct, check the **Credentials** with the **Trust Engine** to verify the identity of the **User**.

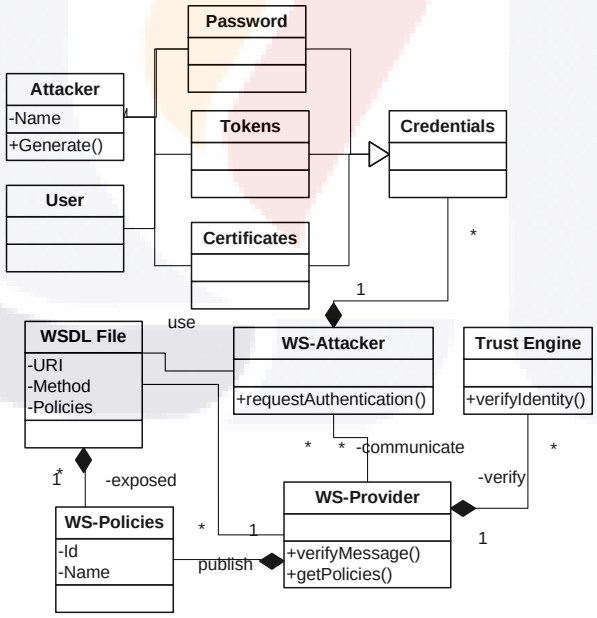


Figure 5: Class Diagram Forged Claims

The class diagram of Figure 5 shows the structure of the forged claims attack. The class **Attacker** and **WS-Attacker** are the elements of the attacker. The **Attacker** is the person responsible of making the attacks, and the **WS-Attacker**, can be a new web service generated by the Attacker using the **WSDL-File** or can be the same web service like **WS-Requester** showed in the above diagram.

The Attacker can find the location of the web service to wich want access; this is possible with the **WSDL-File** of the web service. The WSDL-File exposes the methods and policies necessary to communicate with the web service. To obtain the credentials of the **User** the Attacker can use tools kind of sniffing like WireShark or Scarab.
This attack can be a consequence of the Middle Man attack. Also Credentials like tokens can be in a cookie file in the terminal of the user. In the case of the certificates, this can be auto-generated, with the same information of the provider, but without his signature.
With this credentials, Is possible create a web service ans assemble a message to request authentication to the web service provider and try to access to several resources.

The sequence diagram of Figure 6 shows the sequence of steps to perform an instance of Forged claims. An Attacker, with knowledge of the place in where the web service provider resides, can obtain o generate credentials, and with this credentials, he can make a web service to generate requests to the web service provider, can even use the same web service used by a valid user. When the attacker had the credentials, he make a request to the WS-Provider using the WS-Attacker, in the request, the attacker send the credentials, auto generated to the WS-Provider. When the WS-Provider receives credentials, the message is checked to validate the correct structure of the message. If the message is valid, then the content of the message is checked against the WS-Policies, to verify the correct use of these in the message. If the policies are correct, the message is sent to the Trust Engine.

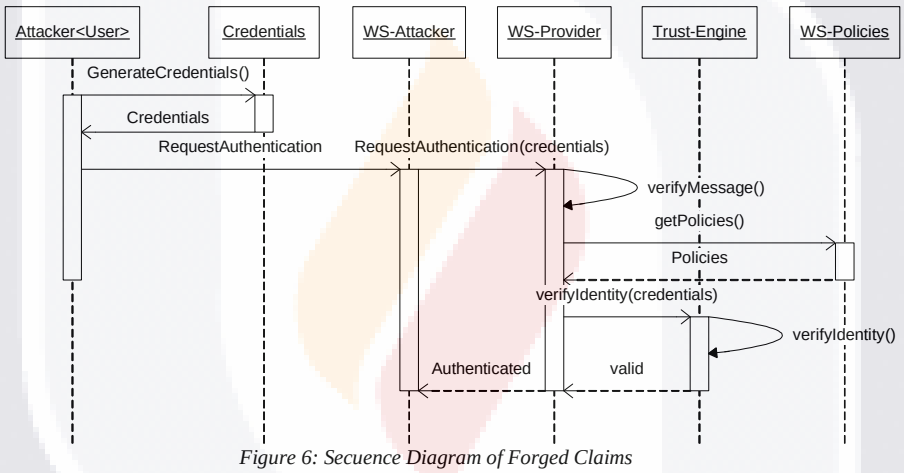


Figure 6: Secuence Diagram of Forged Claims

The trust Engine check the identity of the user throught their credentials. If the credentials are correct a message with the positive validation of the credentials is sent to the WS-Provider. The WS-Provider sent this message to the User with the positive authentication of theirs credentials.

In this Attack, the vulnerability is foundly mainly in the Trust Engine, because, it must be able to verify in the correct way the credentials.

Consequences
If the attack is successful, the attacker can access to the functionalities of the system, also can elevate privileges of the user that is using or another user, also can add, delete or modify delicate information in the system records. This kind of attack can be a consequence of the middle man attack.
Countermeasures and Forensics
The first thing to do is ensure that the authentication process is correct, because it is the key point at which the problem occurs. Regarding to the self-generated tokens, the tokens issued should be reviewed and verify the policies of use of these tokens. And the issued tokens will be omitted.

Related Patterns

WS-Trust: This pattern helps to establish the relationships that should arise between Web services, to share their credentials [9].

WS-Policy Pattern: This pattern helps to define security policies necessary for communication between web services [11].

Credential Pattern: this pattern shows how a set of credentials and the authentication process with the authorities that issued them [11].

4.3 Middle Man

The Man in the Middle attack is possible in [13] using "SOAPAction Spoofing", even when security standards like WS-Security have been implemented. To achieve this kind of attack the attacker is able to downgrade the level of cryptography used to secure the message.
In this way, when the attack takes place, both the provider and the requester, do not know the changes that are made by the person in the middle. This kind of attack is in the area of authentication, because the parties interact with another entity, which is not a party whom they trust.
The damage that this can cause is in the decision making process, which occurs in the affected parts, as these could be changed, altering information leading to elevation of privileges.

Name
Man in the Middle
Intent
In the Man in the Middle attack [12] an attacker insert himself in the middle of communications between two web services without being detected., without any of them is aware of what happens.
The attacker makes this attack because it needs manipulate information between both parties in his favor. If is an economic transaction, the manipulated in order to apply the products they want or make deposits to a bank account owned. If the information is being exchanged is very delicate, the attacker wants to access it since it could be on credentials exchanged between both parties, which would help to develop another type of attack.
Context
Using web services can share information among various entities. To achieve this, we must know where it is resident and what methods expose to be consumed.WSDL file exposes the methods, security policies and the URI where the web service is a resident. Communication between the two sides may or may not be protected; to protect it, can use WS-Security and WS-Trust. When performing communication relies on the other party is who they claim to be. To access the functionality of a Web service, you must create another service that address to the URI obtained from the WSDL file. The WSDL file can even be signed by the web service provider in such a way that guarantees their legitimacy.
Problem
How perform a Middle Man successfully?
How to get the information shared between two parties?
How alter the information shared between two parties?
Solution
In the Figure 7 shows the class diagram of a common interaction between two web services, in this diagram an **User** can access to a web service (**WS Provider**) throught of a **WS Requester,** to make this web service, the user need to know the **WSDL-File** to know the methods and policies from the **WS-Provider**, also the **User**, may need to have some **Credentials** to access to the web service

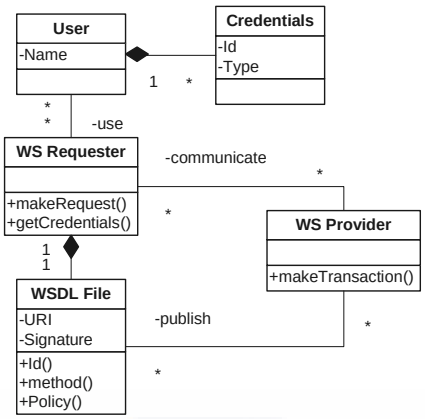


Figure 7: Web Service Interaction

The class diagram of Figure 8 shows the structure of the Mand in the Middle attack. The class **WS Attacker** and **WSDL File Attacker** are elements introduced by the attacker. **WS Requester** is another web service developed by the attacker

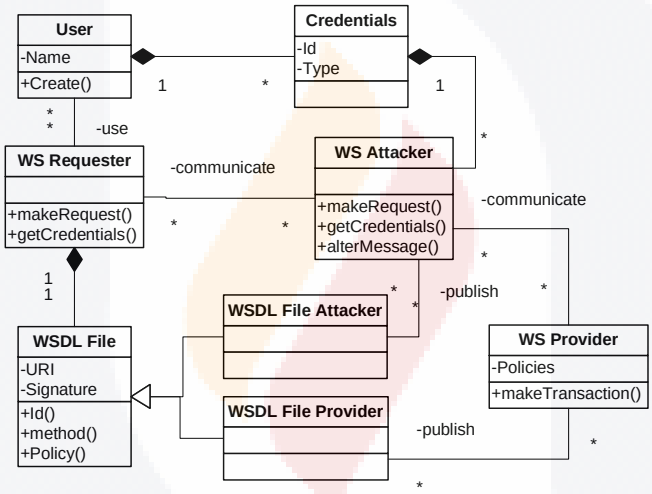


Figure 8: *Misuse Pattern:Man in the Middle*

In the Figure 9 shows the sequence diagram with the steps to perform a man in the middle attack.
An **Attacker** needs to obtain information of the original WSDL file from the application that others want to access. And may publish a **WSDL-File Attacker**, with the same methods of the original WSDL-File of the web service, but in this file the URI from the web service is another from the original. When a **User** access to this file, the request are addressed to another place. In this way, the **WS-Attacker** stay in the middle of the communication of the **WS Requester** and the **WS Provider,** and can access to all the information between them, also it can copy or modify the information .

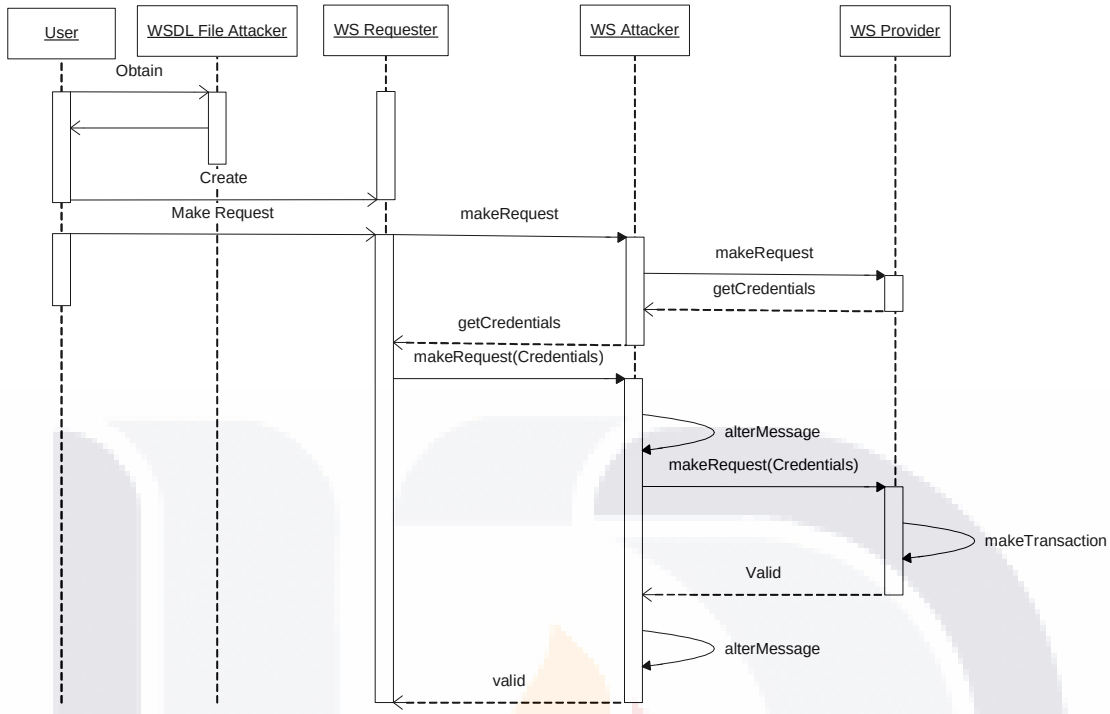


Figure 9: man in the Middle Secuence Diagram

Consequences

As a result of the attack, the communications between the two parties will be compromised, the "person in the middle" now decides that is what the parties mutually sent and receive. In some situations when communications security is implemented but not in a strong way, and a system of timestamp is used, the attacker cannot modify the information, but the information can be copied without modification done by the attacker, for example only copying the information and processes. If the information is sensitive now it will be compromised, or if in this situation was an exchange of credentials, the attacker could have access to these, and make another kind of attack, like principal spoofing.

Countermeasures and Forensics

As a countermeasure, the communications will be protected in the two parties implicated, when transmissions are continuous and with a large amount of data, only the most delicate data will be protected, that for the system performance does not decrease.

As a measure forensic, the data source will be verified; the data source must be another legitimate part. If this is not correct, it must try to find the place of origin by the method of "IP Traceback". And also check the timestamp in the messages and determinate if they have too much latency in their reception respect of the time it was sent.

In the communications infrastructure try to verify the points where the attacker could have infiltrated and protect the communications in special the wireless.

Related Patterns

Symmetric Encryption and XML Encryption Patterns: This pattern help to the encryption of the data while this are traveling between two points [14].

WS-Trust: This pattern helps to establish the relationships that should arise between Web services, to share their credentials [9].

WS-Policy Pattern: This pattern helps to define security policies necessary for communication between web services [15].

5 Conclusions

Through the creation of the patterns listed above, in the moment of develop web services, designers may have a point of reference, about how avoid potential attacks and vulnerabilities, to which they could face. And also have a guide about how design a web service that problem.

6 Future Work

How future work, we will define more misuse patterns about authentication of web services and try to test the patterns developed in this work in a empirical way.

7 References

[1] L. J. Camp, "Identity, Authentication, and Identifiers in Digital Government," 2003.

[2] OWASP. (2010) OWASP Pruebas de Autenticación. [Online]. http://www.owasp.org/

[3] C. Wong and D. Grzleak. (2009, Nov.) SIFT Information Security Services. [Online]. http://www.sift.com.au/assets/downloads/SIFT-Web-Services-Security-Testing-Framework-v1-00.pdf

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley, 1996.

[5] E. B. Fernandez, J. C. Pelaez, and M. Larrondo- Petrie, "Attack Patterns, A New Forensic And Design Tool," in *IFIP International Federation for Information Processing*, P. Craiger and S. Shenoi, Eds. SpringerLynk Boston, 2007, vol. 247, ch. 24, pp. 345-357.

[6] E. B. Fernandez, Y. Nobukazu, and H. Washizaki, "Modeling Misuse Patterns," *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pp. 566-571, Mar. 2009.

[7] J. C. Pelaez, E. B. Fernandez, L.-P. Marie M, and C. Wieser, "Misuse patterns in VoIP," in *14th Conference on Pattern Languages of Programs*, Illinois, 2007.

[8] E. B. Fernandez, N. Yoshioka, and H. Washizaki, "A Worm misuse pattern," in *AsianPLoP 2010: 1st Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, 2010.

[9] O. Ajaj and E. B. Fernandez, "A pattern for the WS-Trust standard for web services," in *AsianPLoP 2010: 1st Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, 2010.

[10] A. Barbir, C. Hobbs, E. Bertino, F. Hirsch, and L. Martino, "Challenges of testing Web Services and security in SOA Implementations," in *Test Analysis of web Services*. SpringerLink, Sep. 2007, pp. 395-440.

[11] P. Morrison and E. B. Fernandez, "The Credential Pattern," in *2006 conference on Pattern languages of programs*, Portland Oregon, 2006.

[12] Web Services Interoperabily Organization. (2005, Jul.) Web Services Interoperabily . [Online]. http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0-20050507.doc

[13] M. Jensen, N. Gruschka, and R. Herkenhoner, "A Survey of Attacks on Web Services: Classification and countermeasures," *Computer Science - Research and Development*, vol. 24, no. 4, pp. 185-197, May 2009.

[14] K. Hashizume and E. B. Fernandez, "PATTERN LANGUAGES OF PROGRAMS CONFERENCE 2009," in *PATTERN LANGUAGES OF PROGRAMS CONFERENCE 2009*, Chicago Illinois, 2009.

[15] O. Ajaj and E. B. Fernandez, "A pattern for the WS-Policy standard," in *1st Asian Conference on Pattern Languages of Programs*, Tokyo, Japan, 2010.

[16] A. Nash, W. Duane, C. Joseph, and D. Brink, *PKI Infraestructura de Claves Publicas*. McGraw-Hill, 2002.

[17] E. Bertino and L. D. Martino, "A Service-Oriented Approach to Security--Concepts and Issues," in *11th IEEE International Workshop on Future Trends of Distributed Computing Systems*, Sedona, AZ, USA, Mar. 2007.

[18] A. Singhal, T. Winograd, and K. Scarfone. (2007,) http://www.nist.gov/. [Online]. http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf

[19] B. Hartman, D. J. Flinn, K. Beznosov, and S. Kawamoto, *Mastering Web Services Security*, K. A. Malm, Ed. Indianapolis, Indiana, EUA: Whiley Publishing, 2003.

[20] H. Washizaki, E. B. Fernandez, K. Marayuma, and A. Kubo, "Improving the Classification of Security Patterns," in *20th International Workshop on Database and Expert Systems Application*, Linz, Austria, 2009, pp. 165-170.

[21] OASIS. (2006, Feb.) Web Services Security. [Online]. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss#overview

[22] OASIS. Web Service Policy. [Online]. http://www.w3.org/2002/ws/policy/

C. Constancia de Estancia de Investigación en la FAU (Florida Atlantic University)

30 julio de 2010

Alumno: Héctor Caudel García

Asesor Nacional: Jaime Muñoz Arteaga

Asesor Internacional: Eduardo B. Fernández

Durante la estancia en Florida Atlantic University, realizada del 17 de mayo al 31 de Julio de 2010, se realizaron las siguientes actividades:

Se expuso ante profesores de Florida Atlantic University, el tema de tesis sobre el cual se está trabajando, y se recibió retroalimentación por parte de ellos.

Se replanteo el proyecto a trabajar con FAU sobre patrones de autenticación, para hacerlo más especifico hacia servicios web, debido a que la definición del problema a atacar, estaba orientada hacia el área de autenticación en general

Se escribió el articulo llamado "Misuse Patterns for Web Services Authentication", en el cual se describen Misuse Patterns para tres problemas concernientes a la autenticación de los servicios web, estos patrones están enfocados a ver el problema desde el punto de vista del atacante, todo ellos para poder obtener un panorama general de cómo se realiza este ataque mediante un esquema de patrón, como contrarrestarlo y las pruebas forenses necesarias para poder identificarlo.

En base al trabajo realizado para la elaboración del artículo anteriormente mencionado se fortaleció el sustento para la definición de la problemática de tesis

Eduardo B. Fernández
Florida Atlantic University

Jaime Muñoz Arteaga
Universidad Autónoma de Aguascalientes

FAU
FLORIDA ATLANTIC UNIVERSITY

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES