ATENOS: Un Programa para Mejorar la Seguridad en WSDL

Agustín Ferrari, Edgardo Bernardis, Mario Berón, Hernán Bernardis,
Maria Joao Tinoco Varanda Pereira, Miguel Bustos, Daniel Riesco
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950, San Luis, Argentina
Departamento de Informática e Comunicações
Instituto Politécnico de Bragança
Bragança, Portugal
{ebernardis, mberon, hbernardis, driesco}@unsl.edu.ar,
{ferrariagustin93, miguelbsts}@gmail.com, mjoao@ipb.pt

Abstract

Con el crecimiento de internet y las distintas dinámicas de la sociedad actual, ha cambiado en gran medida la forma de interactuar e intercambiar información entre las personas y las empresas. Este intercambio se vuelve blanco de ataques por parte de todos aquellos actores que quieren obtener información útil y valiosa a sus propios intereses o de terceros. Ante este panorama se vuelve imperioso implementar todo tipo de medidas y acciones tendientes a evitar estos ataques, por tal motivo nace lo que se denomina Seguridad Informática. Toda acción, herramienta o metodología enfocada a evitar, contrarrestar o retrasar ataques contra activos sensibles juega un rol sumamente importante para los diversos actores.

Por lo antes explicado, se describe en este artículo una herramienta cuyo principal objetivo es desarrollar e incrementar el nivel de seguridad de Servicios Web.

1. Introducción

Actualmente se están popularizando los Web Services como artefactos de software a partir de los cuales se pueden construir sistemas más complejos. Según la W3C, un Web Service es: "*Una aplicación de software identificada por una URI, cuya interface y enlaces son capaces de ser definidos, descritos y descubiertos como artefactos XML. Un web service soporta interacción directa con otros agentes de software usando mensajes basados en XML intercambiados a través de protocolos basados en internet*". Muchas organizaciones construyen sus sistemas basándose en una arquitectura orientada a servicios web, algunos de ellos se publican al resto del mundo de manera libre, mientras que otros son utilizados de manera interna por sus equipos de desarrollo. Esto permite que cada equipo de desarrollo elija la arquitectura que desee para construir sus proyectos sin afectar la vinculación con el resto del sistema y/o proyectos. La interacción entre proyectos se convierte en un intercambio de mensajes con la información necesaria dentro, sin la necesidad de vinculación a nivel de arquitectura subyacente más que la necesaria a la invocación de los servicios web.

Construir un Web Service y que pueda ser utilizado por cualquier otra persona u organización en el mundo ha sido posible debido a la creación de estándares y lenguajes formales para la definición de los mismos.

Todo Web Service posee una especificación que provee la información necesaria para invocarlo. Uno de los estándares de descripción más conocido es WSDL (Web Service Definition Language) [15]. Las especificaciones WSDL son un dialecto XML, con reglas bien definidas para especificar cada componente del WS. Cuántos parámetros recibe y de qué tipo son, qué datos retorna y de qué tipo, qué protocolo de internet usa para su comunicación, qué operaciones posee, son entre otras tantas, características del WS que se encuentran especificadas en su WSDL asociado.

Así como el archivo WSDL sirve para que un agente de software o persona pueda interpretarlo para usar el servicio web que describe, también puede dar información a personas no deseadas o incluso exponer vulnerabilidades. Más aún si se considera que existen herramientas que generan los WSDLs de manera automática para un servicio web, con lo cual el nivel de

atención a la información que se publica no siempre se encuentra bajo un estricto control. Esto se vuelve más importante para aquellos casos en donde los servicios web pertenecen a bancos, tarjetas de créditos, servicios de compra/venta online, entre otros. Incluso también para los servicios web que no se publican, son privados y necesitan mayor control y seguridad como los que pertenecen a empresas privadas y redes militares.

Empresas competidoras pueden aprender el know-how y conseguir copiar el diseño para ofrecer servicios similares y competitivos. Pero no solo se trata de competencia, los ataques de seguridad como espionaje de información, suplantación de clientes, inyección de comandos y denegación de servicio también son posibles ya que los atacantes pueden aprender sobre los datos intercambiados y los patrones de invocación de los documentos WSDL. Si bien la legibilidad de las descripciones de los servicios hace que los servicios web sean reconocibles, también contribuye a la vulnerabilidad del servicio [16]. Todos contienen información formal (código fuente) e informal (identificadores, comentarios, documentación, etc.) y es en este tipo de información en donde los atacantes hacen foco para obtener información beneficiosa a sus propósitos. Suena lógico entonces incrementar la seguridad que posee un determinado WSDL para evitar e impedir los ataques.

2. Seguridad

Con los avances de la tecnología, sobre todo en el ámbito de internet, se vuelve sumamente importante y necesario la protección de todo tipo de información. En la actualidad, es realmente alta la cantidad de delitos que se llevan adelante en contra de información personal o de empresas. Todo tipo de información es valiosa, ya sea desde simples datos personales hasta sistemas y bases de datos empresariales.

Con el auge de internet, el intercambio de archivos se ha vuelto un punto esencial en la sociedad actual. Los consumidores intercambian información no sólo entre ellos sino también con los vendedores. Para el intercambio de información se utilizan diferentes medios entre los que se pueden mencionar redes sociales, correo electrónico, sistemas punto a punto, pagos online, juegos. Todo esto se fundamenta en la confianza y el correcto funcionamiento del software y del hardware subyacente a dicho proceso.

Por lo antes mencionado es que surge lo que se conoce como Seguridad Informática (SI). Existen diversas definiciones de SI, en el caso de este trabajo, se adhiere a la siguiente definición de SI: *Preservación de la confidencialidad, integridad y disponibilidad de la información en el Ciberespacio. A su vez, el Ciberespacio se define como el entorno complejo que resulta de la interacción de las personas, software y servicios en Internet por medio de redes y dispositivos tecnológicos conectados a el, y que no existe en ninguna forma física* [1].

La información es un conjunto organizado de datos, que cambia su enfoque y su estado de conocimiento dependiendo del ámbito en la que se la utilice. Por ejemplo, si la información se conceptualiza bajo el punto de vista de la ingeniería es el estudio de las características y estadísticas del lenguaje que permite su análisis desde un enfoque matemático, científico y técnico. Desde el punto de vista de una empresa es el conjunto de datos propios que se gestionan y mensajes que se intercambian personas y/o máquinas dentro de una organización [2].

La información se ve afectada por muchos factores, motivo por el cual se vuelve importante su seguridad. De aquí que Seguridad de la Información es: *una disciplina, cuyo principal objetivo es mantener el conocimiento, datos y sus significados libres de eventos indeseables, tales como el robo, espionaje, daños, amenazas y otros peligros. La Seguridad de la Información incluye todas las acciones tomadas con anticipación, para evitar eventos no deseados* [3].

El objetivo de la SI es obtener un nivel aceptable de seguridad, entendiéndose por aceptable un nivel de protección suficiente para que la mayor parte de potenciales intrusos, interesados en los equipos con información de una organización o persona, fracasen en cualquier intento de ataque contra los mismos. Asimismo, se encarga de establecer los mecanismos para registrar cualquier evento fuera del comportamiento normal y tomar las medidas necesarias para re-establecer las operaciones críticas a la normalidad [4].

Los principios generales de la seguridad de la información [5, 6] son:

- Integridad: implica que debe salvaguardarse la totalidad y la exactitud de la información que se gestiona. A su vez, puede incluir:
  - Autenticidad: autoría indiscutible. Definir que la información requerida es válida y utilizable en tiempo, forma y distribución.
  - No Repudio: la seguridad de que una parte no puede negar posteriormente los datos de origen; suministrando la prueba de integridad y el origen de los datos, y que puede ser verificado por un tercero. Evitar que cualquier entidad que envió o recibió información alegue, ante terceros, que no la envió o recibió.
- Confidencialidad: implica que debe protegerse la información de forma tal que sólo sea conocida y accedida por las personas

autorizadas y se la resguarde del acceso de terceros.
- Disponibilidad: implica que debe protegerse la información de forma tal que se pueda disponer de ella para su gestión en el tiempo y la forma requerida por el usuario.

El punto o centro de ataque a la seguridad informática se da en una Vulnerabilidad: *debilidad de un activo o control que puede ser explotado por una o más amenazas* [7]. La presencia de una vulnerabilidad no puede causar daño en sí misma, ya que es necesario que exista una amenaza que la aproveche. Una vulnerabilidad que no tiene una amenaza, puede no requerir la aplicación de un control, pero debe ser reconocida, supervisada y, en lo posible, eliminada.

Las amenazas surgen a partir de la existencia de vulnerabilidades, es decir que una amenaza sólo puede existir si existe una vulnerabilidad que pueda ser aprovechada, independientemente de que se comprometa o no la seguridad de un sistema de información. Una amenaza se puede definir [8] como: *cualquier elemento o acción que es capaz de aprovechar una vulnerabilidad y comprometer la seguridad de un sistema de información*.

Las amenazas se pueden clasificar o dividir en dos tipos; las intencionales, en caso de que deliberadamente se intente producir un daño (por ejemplo el robo de información). Las no intencionales, en donde se producen acciones u omisiones de acciones que si bien no buscan explotar una vulnerabilidad, ponen en riesgo los activos de información y pueden producir un daño (por ejemplo las amenazas relacionadas con fenómenos naturales).

3. Ofuscación

Según el Diccionario de la Real Academia [9], ofuscar significa deslumbrar, turbar la vista, oscurecer, trastornar o confundir las ideas. Es decir, se refiere a encubrir deliberadamente el significado de alguna cosa haciéndola más confusa y complicada de interpretar, evitando la comprensión de la misma. La palabra ofuscación fué elegida para esta actividad porque connota oscuridad, ininteligibilidad y desconcierto, y porque ayuda a distinguir este enfoque de otros métodos. La ofuscación se puede comparar con el camuflaje, aunque este último a menudo se considera una herramienta para la desaparición total [10].

3.1. Ofuscación de Código

La ofuscación de código es un conjunto de transformaciones que convierte un programa en uno funcionalmente equivalente, pero ininteligible haciendo difícil su entendimiento y aplicarle ingeniería inversa. La ofuscación de código aplica una o más transformaciones de código que hacen que el código sea más resistente al análisis y la manipulación, pero preservan su funcionalidad [11].

La ofuscación de código es un área rica para la exploración de la ofuscación en general, la cual está progresando hacia sistemas que son relativamente fáciles de usar y enormemente difíciles de vencer. Esto es incluso aplicable al hardware en el cual se están utilizando componentes dentro de los circuitos para crear una “ofuscación lógica” con el fin de evitar la ingeniería inversa de la funcionalidad de un chip [12].

3.1.1. Transformaciones de Código. La investigación sobre protección de software ha aumentado de forma constante en la última década. La compilación de código se ha convertido en mucho más que solo traducir un programa de computadora en uno ejecutable. Por lo general los programas se escriben en un lenguaje de alto nivel dadas las ventajas que estos ofrecen. Sin embargo tiene ciertas desventajas que hacen que la compilación de código incluya implícitamente numerosas técnicas de optimización. que van desde eliminar código muerto, asignación de registros óptima y asignaciones eficientes al objetivo conjunto de instrucciones de la arquitectura [11]. Y es en estas etapas de compilación y/o desarrollo de software en donde se pueden realizar y aplicar distintas transformaciones de código.

Se han desarrollado muchas técnicas para maximizar la ofuscación de código, de las distintas partes que componen un programa, de tal manera que su análisis sea sumamente difícil. Algunas tan simples como codificar el nombre de los identificadores, pero la ofuscación de código ofrece muchas más posibilidades y variedades. Una buena ofuscación se compone de una o más transformaciones de código que transforman un programa de tal forma que resulta más difícil aplicarle ingeniería inversa. La única restricción para estas transformaciones, sean manuales o automatizadas, es preservar la funcionalidad original del programa.

Las transformaciones de código para ofuscar un programa se pueden dividir en cuatro clases principales [13], estas son:

- Transformaciones Léxicas o de Diseño: afectan la información en el código que es innecesaria para su ejecución y que reduce la información disponible para un lector humano. Ejemplo de esto es la codificación de nombres, la eliminación de comentarios, etc.
- Transformaciones de Flujo de Control: actúan modificando el flujo de control del programa.
- Transformaciones de Flujo de Datos: operan sobre las estructuras de datos usadas en el programa.

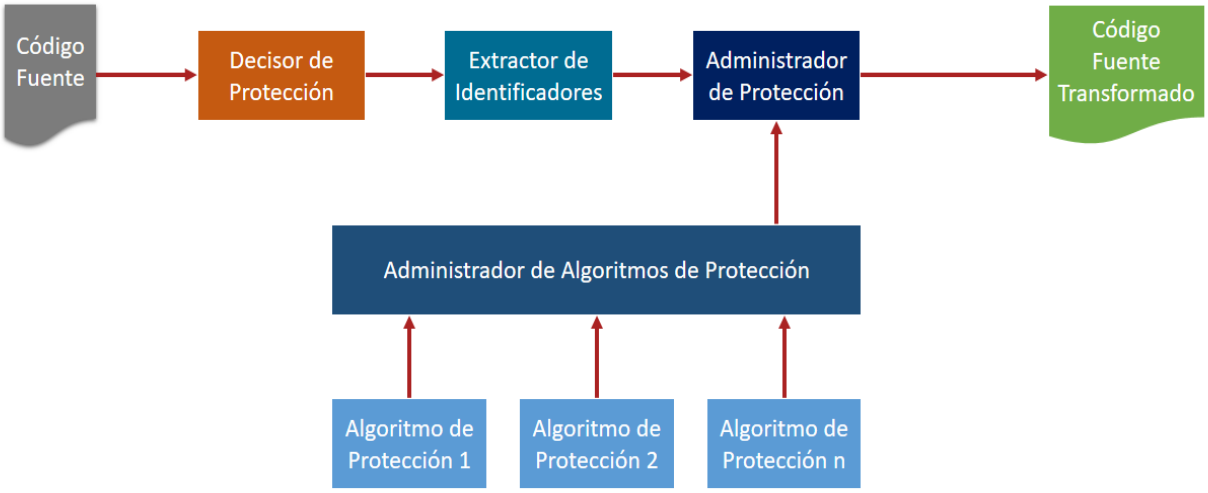

Figura 1

- Transformaciones Preventivas: intentan detener el funcionamiento correcto de los decompiladores o desofuscadores de código.

4. Trabajos Relacionados

Antes de describir la herramienta que hace a esta investigación, es importante detallar que otras soluciones se encuentran en la bibliografía respecto de la temática planteada.

Muchas investigaciones se concentran en el nivel de seguridad de los mensajes donde los mensajes SOAP son transferidos entre un Servicio Web y sus clientes [18].

El framework SOA básico no posee una seguridad adecuada y las implementaciones de seguridad disponibles dependen del propietario del framework.

En el caso de los WSDL solo unos pocos investigadores han propuesto soluciones de seguridad para contrarrestar los ataques. Algunos proponen un modelo para encriptar documentos WSDL para manejar los problemas de seguridad [19], afirmando que es adecuado para servicios web que tienen reglas críticas para definir políticas de seguridad organizacional. Otro propone un framework de seguridad para proteger WSDL usando estándares de seguridad de servicios web incluyendo cifrado XML y firmas digitales XML [20].

En mayor o menor medida las soluciones propuestas giran en torno a frameworks y/o utilizan métodos de encriptación para los archivos WSDL, pero lo más marcado de la situación es que no hay tanta investigación o propuestas para mejorar la seguridad en WSDLs.

5. Herramienta

En esta sección se describe ATENOS (**A**u**T**omatic **EN**criptation **O**fuscation **S**ystem) una herramienta cuyo principal objetivo es mejorar la seguridad de Servicios Web basados en WSDL.

5.1. Arquitectura

Como se observa en la Figura 1, ATENOS se compone o divide arquitecturalmente en las siguientes componentes:

- Decisor de Protección.
- Extractor de Identificadores.
- Administrador de Protección.
- Administrador de Algoritmos de Protección.

Cada una de estas partes tienen un rol y funcionalidad marcada y diferenciada del resto. Cada una de estas componentes se explican a continuación.

5.1.1. Decisor de Protección. En este punto la herramienta puede tomar dos caminos diferentes en su ejecución. El software realiza un profundo análisis sobre el WSDL a fin de detectar si contiene la clave de seguridad que indica su estado de protegido o desprotegido. En el caso de estar desprotegido, se da comienzo al proceso de protección mediante el módulo de extracción de identificadores. De otro modo, si el WSDL contiene la clave, ATENOS toma otro rumbo que desemboca en el proceso de desprotección del documento, volviendo a su estado original.

5.1.2. Extractor de Identificadores. Para poder aplicar técnicas que incrementen la seguridad de un servicio web, es necesario primero extraer información del mismo. Las modificaciones a realizar deben ser muy precisas y específicas para no introducir errores que cambien la lógica o funcionamiento del servicio web. Las técnicas de extracción de información juegan un papel fundamental debido a que su nivel de precisión en la información a extraer determina en gran medida el éxito o no del proceso. Si se desea modificar el nombre de un identificador pero la modificación se aplica a una etiqueta reservada del WSDL, el proceso no solamente es erróneo sino que generará un WSDL incompleto e incluso un archivo XML mal formado por poseer etiquetas que no cierran.

En el caso particular de este trabajo, se utilizan los reconocidos parsers DOM (Document Object Model) cuya efectividad se encuentra más que comprobada y, al ser específicos para archivos XML, su aplicación es prácticamente directa. Estos parsers construyen el árbol de sintaxis abstracta AST (abstract syntax tree) de un documento XML con un nivel de información elevado evitando la pérdida de tiempo en la creación de un parser para XML. Cada nodo del árbol posee información de las etiquetas del XML y aplicando diferentes recorridos al AST se logra extraer la información que se desea. Luego, esta información se analiza, se modifica (o no) y se guarda nuevamente en el AST.

Un recorrido completo final del árbol genera el WSDL modificado.

5.1.3. Administrador de Protección. Toda aplicación web está conformada por distintos tipos de información, tanto formal como informal. En base al análisis detallado de la misma es posible definir estrategias que permitan subsanar las vulnerabilidades y proteger las partes que se consideren susceptibles de ataques [14].

Utilizando la información extraída del WSDL en la parte de extracción de información (Extractor de Identificadores) se pueden manipular diferentes partes del mismo para mejorar su seguridad. Esto se puede lograr mediante la utilización de funciones de ofuscación al realizar las modificaciones y/o transformaciones necesarias que aumentaran el nivel de seguridad. Estas transformaciones pueden ser sobre partes específicas del WSDL (identificadores, operaciones, etc.) o en la totalidad del mismo. Dichas modificaciones dependen del nivel de seguridad deseado, partiendo de un nivel básico en donde se ofuscan partes específicas del WSDL, como por ejemplo el nombre de las operaciones, hasta llegar a un nivel máximo en donde se realiza una transformación completa del WSDL.

Este módulo se encarga de aplicar los algoritmos (propios como agregados) que mejoran la seguridad de los WSDLs según el criterio elegido por el usuario. Cada algoritmo disponible para aplicar tendrá su propia técnica o metodología, en el caso caso de los algoritmos nativos de ofuscación provistos junto con la herramienta utilizan como método para proteger la información Transformaciones Léxicas o de Diseño. Esto se debe a que dichas transformaciones se aplican sobre nombres; como se está trabajando con WSDLs cuyo lenguaje base es el XML, el cual es un lenguaje de transporte de datos y en el caso de los WSDL la mayoría de la información son Tags con información y nombres.

En el caso de los algoritmos personalizados por el usuario pueden utilizar las técnicas que consideren necesarias. Pero más allá que el sistema está enfocado en la ofuscación como técnica para mejorar la seguridad, el sistema está preparado para aplicar incluso algoritmos de encriptación. El sistema brinda la posibilidad de aplicar el paso inverso necesario en toda encriptación, es decir, si se aplicó un algoritmo de encriptación a un WSDL, el sistema permite aplicar la desencriptación del mismo archivo.

La herramienta permite ofuscar Tags WSDL predefinidos, la ofuscación puede realizarse sobre alguno en particular o en combinación de todos ellos. Los Tags WSDL que la herramienta da como opción para asegurar son:

- Operaciones.
- Puertos.
- Servicios.
- Direcciones.

5.1.4. Administrador de Algoritmos de Protección. La herramienta por sí misma cuenta con algoritmos de ofuscación propios para incrementar la seguridad de un WSDL, los cuales no se pueden modificar o eliminar de la herramienta. Sin embargo ATENOS cuenta con un componente particular que permite administrar los algoritmos de seguridad que se pueden utilizar a la hora de incrementar la seguridad de un WSDL. Agregar, modificar y eliminar nuevos algoritmos en la herramienta son las funcionalidades que brinda este módulo particular. Esta característica la distingue entre otras herramientas similares, ya que dentro de la bibliografía consultada, los algoritmos o metodologías son acotados a los provistos por los desarrolladores de la misma sin proveer un mecanismo para agregar algoritmos de terceros.

Es importante notar que a la hora de realizar la protección del documento, el usuario debe seleccionar el algoritmo que desea usar para proteger el WSDL. Estos algoritmos pueden cambiar en ejecución debido a que el módulo de Administrador de Algoritmos de Protección permite incorporar nuevos algoritmos propuestos por el usuario, por lo tanto, a la hora de seleccionar un algoritmo y utilizarlo, no es posible crear directamente

un objeto de la clase algoritmo seleccionado, ya que si el usuario selecciona un algoritmo que él creó, no se puede conocer el nombre exacto para crear un objeto de esa clase. Por lo tanto, como es necesario un objeto de el algoritmo (Clase) que el usuario selecciona se hace uso de la Reflexión.

La Reflexión es: *la capacidad integral de un programa para observar o cambiar su propio código, así como todos los aspectos de su lenguaje de programación (sintaxis, semántica o implementación), incluso en tiempo de ejecución. Se dice que un lenguaje de programación es reflexivo cuando proporciona a sus programas la capacidad de reflexión* [17]. La palabra integral es muy importante, la verdadera reflexión no impone límites a lo que un programa puede observar o modificar. Sin embargo la reflexión que se lleva adelante en Atenos solo se circunscribe a este módulo. Esto es no sólo por motivos de seguridad, sino también por el hecho de que es necesaria solo en este ámbito del sistema; por lo tanto es innecesaria incluirla en todo el programa.

En el caso de la eliminación, la tarea es directa y trivial, sólo basta con seleccionar y eliminar el algoritmo deseado. Una explicación más detallada merece el agregado o modificación de algoritmos.

Al momento de agregar un nuevo algoritmo es necesario cumplir con unos pasos específicos de la herramienta para que luego su funcionamiento y utilización sea correcta. Al agregar algoritmos, el sistema necesita de forma obligatoria el nombre del algoritmo y de forma opcional la descripción del mismo. Acto seguido el sistema automáticamente crea un "algoritmo" con la estructura necesaria para poder ser incorporado. Este es agregado y visualizado en la tabla de algoritmos y marcado como disponible para su utilización ya que no posee errores de compilación.

Una vez hecho esto, el usuario debe modificar o programar su funcionalidad (o de cualquier otro algoritmo agregado según corresponda). De esta forma el usuario puede desarrollar todo aquello que sea necesario para el algoritmo personalizado siempre y cuando se apegue a ciertas reglas específicas.

El sistema crea una clase con el nombre, dado por el usuario al momento de crear un nuevo algoritmo, en la cual existen dos métodos obligatorios a programar, estos son:

1. protect(): realiza todas las acciones necesarias para proteger la información según el método desarrollado por el usuario.
2. unprotect(): realiza todas las acciones necesarias (si corresponde) para desproteger (o volver atrás) la información según el método desarrollado por el usuario en el método protect().

Para manejar la información se utilizan dos atributos importantes y que sin estos, los métodos no se van a ejecutar correctamente. Los atributos son:

1. String informationUnprotect: contiene la información que se desea proteger y se utiliza en el método protect(), o la información que se desprotegió luego de ejecutar el método unprotect().
2. String informationProtect: almacena la información protegida luego de ejecutar el método protect().

Como último paso para que el algoritmo editado pueda utilizarse normalmente, el sistema se reinicia y re-compila de manera automática (previa directiva del usuario).

En caso de que no haya errores de compilación, se incorpora el algoritmo a la lista de algoritmos disponibles y se marca como disponible, caso contrario no se agrega a la lista y se marca como no disponible.

6. Caso de Estudio

Es fundamental presentar un caso de estudio que muestre y valide la funcionalidad y utilidad de ATENOS y de los conceptos presentados. La finalidad de esta sección consiste en mostrar cómo se comporta ATENOS respecto de la información ingresada, siendo los resultados obtenidos acordes a lo esperado.

Como caso de estudio se toma un WSDL real y público orientado al servicio de Facturación Electrónica perteneciente a la AFIP [21]. No es motivo de este trabajo poner en duda la seguridad de tal servicio ni la necesidad de asegurar el mismo, simplemente se utiliza dicho servicio web para mostrar la funcionalidad de la herramienta ante un ejemplo real.

Inicialmente, se selecciona el archivo WSDL del servicio web por medio del botón para *Agregar* (ver figura 2 a). En caso de ocurrir algún error, se puede quitar de la lista de selección el archivo usando el botón *Quitar* (ver figura 2 b).

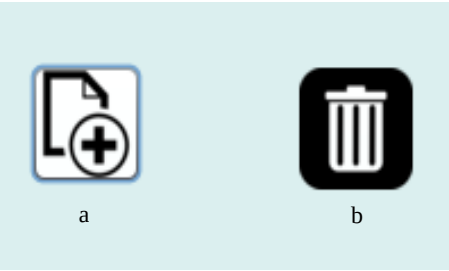

Figura 2

Primero que todo, como se observa en la Figura 3, se procede a cargar un archivo WSDL al cual se le quiere mejorar su seguridad.

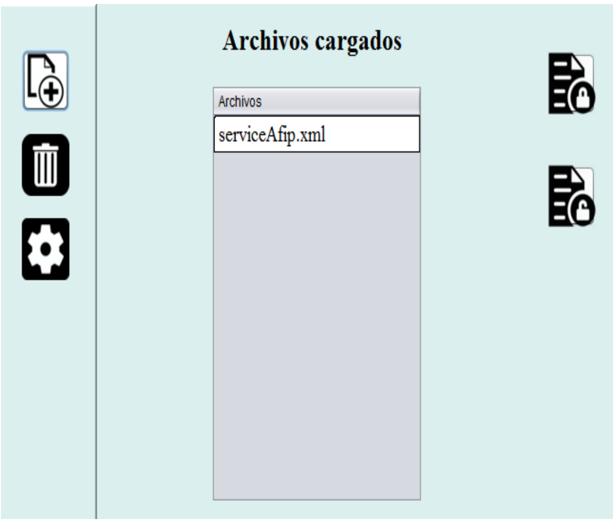

Figura 3

Luego se presiona el botón *Asegurar* (ver Figura 3, botón superior en esquina superior derecha) el cual despliega una ventana con la vista previa del WSDL al cual se le va a aplicar la transformación que mejorará su seguridad. Tal como se ve en la Figura 4.

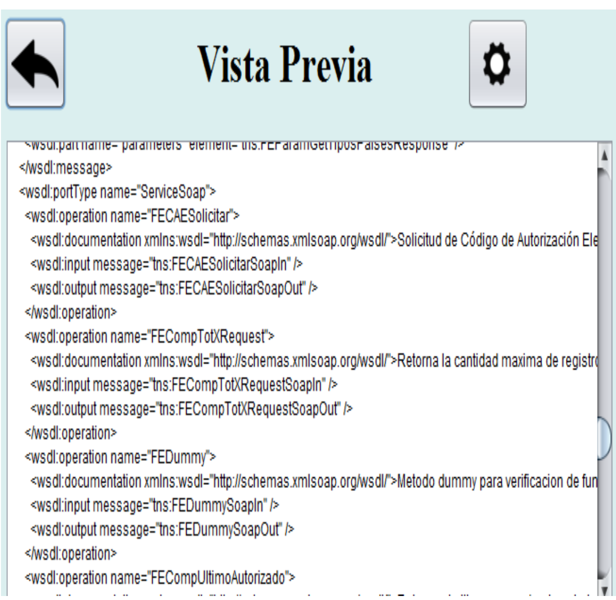

Figura 4

Esta ventana permite previsualizar el WSDL, volver a la ventana anterior (ver figura 4, botón esquina superior izquierda) y realizar configuraciones (ver figura 4, botón esquina superior derecha). Éste último despliega un cuadro de diálogo como el de la Figura 5 en el que se elige el algoritmo para asegurar la información que se va a aplicar junto con los tags del WSDL a los cuales se les va a aplicar el algoritmo.

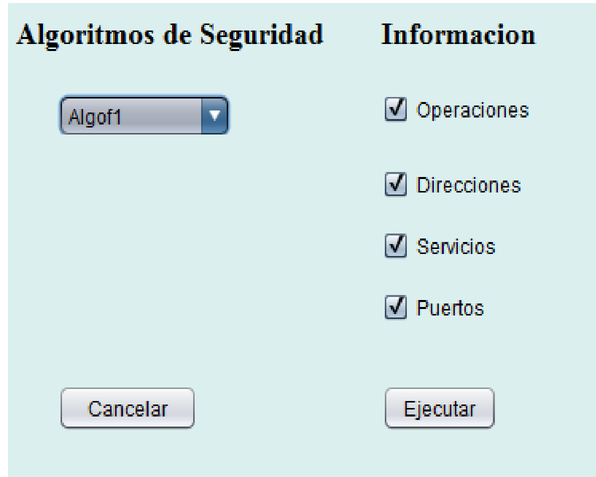

Figura 5

Para asegurar este WSDL se utiliza un algoritmo de ofuscación propio de este equipo de investigación, el cual viene con la misma. Dicho algoritmo realiza transformaciones léxicas aplicadas a los Tags WSDL que se elijan asegurar. En el caso particular de este caso de estudio, se aplican las transformaciones a todos los Tags WSDL disponibles a transformar por la herramienta. Es decir, se aplican las transformaciones para mejorar la seguridad de las Operaciones, Puertos, Servicios y Direcciones (Ver Figura 5).

Presionando el botón *Ejecutar* la herramienta muestra una vista previa del archivo asegurado junto a la opción de *Descargar* (Ver Figura 6, esquina superior derecha), la cual permite guardar el archivo en el disco.

Figura 6

Por una cuestión de espacio, se procede a mostrar extractos del archivo original en donde se muestra cómo Atenos ofuscó los tags de dicho WSDL.

En la Figura 7 se puede observar un extracto del WSDL del archivo original con algunas operaciones sin modificar.

```
970   </wsdl:message>
971   <wsdl:portType name="ServiceSoap">
972     <wsdl:operation name="FECAESolicitar">
973       <wsdl:documentation>Solicitud de Código de Autorización Elec
974       <wsdl:input message="tns:FECAESolicitarSoapIn"/>
975       <wsdl:output message="tns:FECAESolicitarSoapOut"/>
976     </wsdl:operation>
977     <wsdl:operation name="FECompTotXRequest">
978       <wsdl:documentation>Retorna la cantidad maxima de registros
979       <wsdl:input message="tns:FECompTotXRequestSoapIn"/>
980       <wsdl:output message="tns:FECompTotXRequestSoapOut"/>
981     </wsdl:operation>
982     <wsdl:operation name="FEDummy">
983       <wsdl:documentation>Metodo dummy para verificacion de funcic
984       <wsdl:input message="tns:FEDummySoapIn"/>
985       <wsdl:output message="tns:FEDummySoapOut"/>
986     </wsdl:operation>
987     <wsdl:operation name="FECompUltimoAutorizado">
988       <wsdl:documentation>Retorna el ultimo comprobante autorizadc
989       <wsdl:input message="tns:FECompUltimoAutorizadoSoapIn"/>
990       <wsdl:output message="tns:FECompUltimoAutorizadoSoapOut"/>
991     </wsdl:operation>
992     <wsdl:operation name="FECompConsultar">
993       <wsdl:documentation>Consulta Comprobante emitido y su códigc
994       <wsdl:input message="tns:FECompConsultarSoapIn"/>
995       <wsdl:output message="tns:FECompConsultarSoapOut"/>
996     </wsdl:operation>
997     <wsdl:operation name="FECAEARegInformativo">
998       <wsdl:documentation>Rendición de comprobantes asociados a ur
999       <wsdl:input message="tns:FECAEARegInformativoSoapIn"/>
000       <wsdl:output message="tns:FECAEARegInformativoSoapOut"/>
001     </wsdl:operation>
002     <wsdl:operation name="FECAEASolicitar">
003       <wsdl:documentation>Solicitud de Código de Autorización Elec
004       <wsdl:input message="tns:FECAEASolicitarSoapIn"/>
```

Figura 7

Luego de las modificaciones realizadas por Atenos, los nombres de las operaciones han sido modificadas de tal manera que ya no son legibles y/o entendibles como antes.

En la Figura 8 se pueden observar los cambios realizados sobre las Operaciones.

```
</wsdl:message>
<wsdl:portType name="ServiceSoap">
  <wsdl:operation name="IHFDHVrolflwdu">
    <wsdl:documentation>Solicitud de Código de Autorización Ele
    <wsdl:input message="tns:FECAESolicitarSoapIn"/>
    <wsdl:output message="tns:FECAESolicitarSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="IHFrpsWrw[Uhtxhvw">
    <wsdl:documentation>Retorna la cantidad maxima de registros
    <wsdl:input message="tns:FECompTotXRequestSoapIn"/>
    <wsdl:output message="tns:FECompTotXRequestSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="IHGxpp|">
    <wsdl:documentation>Metodo dummy para verificacion de funci
    <wsdl:input message="tns:FEDummySoapIn"/>
    <wsdl:output message="tns:FEDummySoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="IHFrpsXowlprDxwrul}dgr">
    <wsdl:documentation>Retorna el ultimo comprobante autorizad
    <wsdl:input message="tns:FECompUltimoAutorizadoSoapIn"/>
    <wsdl:output message="tns:FECompUltimoAutorizadoSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="IHFrpsFrqvxowdu">
    <wsdl:documentation>Consulta Comprobante emitido y su códig
    <wsdl:input message="tns:FECompConsultarSoapIn"/>
    <wsdl:output message="tns:FECompConsultarSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="IHFDHDUhjLqirupdwlyr">
    <wsdl:documentation>Rendición de comprobantes asociados a u
    <wsdl:input message="tns:FECAEARegInformativoSoapIn"/>
    <wsdl:output message="tns:FECAEARegInformativoSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="IHFDHDVrolflwdu">
    <wsdl:documentation>Solicitud de Código de Autorización Ele
```

Figura 8

El resto de los tags que la herramienta permite cambiar y que están sin ofuscar se pueden observar en la Figura 9.

```
<wsdl:service name="Service">
  <wsdl:documentation>Web Service orientado al servicio de Facturacion electrc
  <wsdl:port binding="tns:ServiceSoap" name="ServiceSoap">
    <soap:address location="https://wswhomo.afip.gov.ar/wsfev1/service.asmx"/>
  </wsdl:port>
  <wsdl:port binding="tns:ServiceSoap12" name="ServiceSoap12">
    <soap12:address location="https://wswhomo.afip.gov.ar/wsfev1/service.asmx"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Figura 9

Luego de aplicar Atenos, las modificaciones que se obtienen al aplicar el algoritmo de ofuscación sobre los tags Puertos, Servicios y Direcciones son los que se observan en la Figura 10.

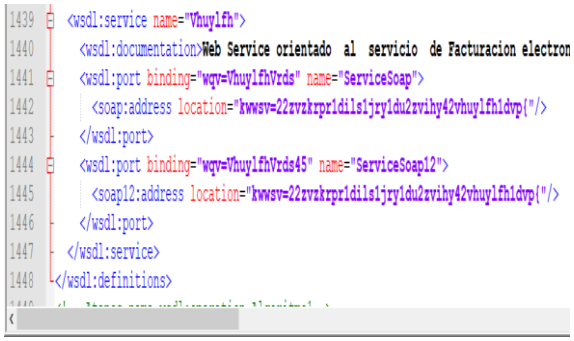

Figura 10

Como se puede observar en las distintas imágenes, aplicar Atenos mejora considerablemente su seguridad al evitar que un atacante pueda entender la información que contienen los distintos tags del WSDL y, en base a esto, realizar los ataques que este tipo de información puede permitirle realizar.

7. Conclusiones

Las conclusiones obtenidas respecto de este trabajo se enumeran a continuación:

- Se desarrolló un prototipo con diferentes recorridos en el árbol de sintaxis abstracta que permite extraer la información de los identificadores.
- Se aplicó el proceso, mediante la utilización de un parser, a distintos WDSLs. Para el caso de estudio de este trabajo se aplico a un servicio web de Facturación Electrónica perteneciente a la AFIP.
- Se logró mejorar significativamente la seguridad del WSDL al ofuscar los tags del Servicio Web dificultando su entendimiento y por consiguiente disminuyendo considerablemente el tipo de ataques que se pueden realizar sobre el mismo.
- Se desarrollaron e implementaron tres algoritmos para ofuscar, los tags de un wsdl, mediante la aplicación de diferentes técnicas de transformaciones léxicas sobre la información tomada como objetivo para incrementar su seguridad.
- Se logró crear una plataforma que permite agregar modificar y eliminar algoritmos propios de los usuarios; siendo la misma una distinguible del resto de las herramientas de similares características.
- Se permite utilizar también la encriptación de los archivos al brindar la opción de desencriptar el archivo.

8. Futuras Extensiones

Como futuras extensiones para esta herramienta, como así también en el ámbito de la investigación se destacan:

- Mejorar las técnicas de ofuscación utilizadas en los algoritmos provistos por la herramienta.
- Ampliar el número de tags que se pueden ofuscar con la herramienta.
- Crear una interface que permita aplicar la ofuscación a diversos lenguajes de programación independientemente de su semántica.
- Parametrizar y abstraer los tags e identificadores de manera que el usuario pueda aplicar la ofuscación a los que considere importantes sin necesidad de que estén predefinidos.

9. Referencias

[1] ISO/IEC. Iso/iec 27032:2012 information technology - security techniques - guidelines for cybersecurity.

[2] Jorge Ramió Aguirre. Libro Electrónico de Seguridad Informática y Criptografía. Universidad Politécnica de Madrid, 2006.

[3] Jeremy Hilton Yulia Cherdantseva. Understanding information assurance and security. 2013A.

[4] Alejandra Stolk. Técnicas de seguridad informática con software libre, 2013. Parque Tecnológico de Mérida. ESLARED.

[5] Salton, International Telecommunication Union. SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY. Telecommunication security. Overview of cybersecurity. 2008.

[6] Salton, Borghello, Cristian F. Seguridad Informática, sus implicancias e implementación. Tesis Licenciatura en Sistemas. Universidad Tecnológica Nacional. 2001.

[7] ISO/IEC. Iso/iec 27000:2016 information technology - security techniques - information security management systems - overview and vocabulary, 2016.

[8] http://www.seguridadinformatica.unlu.edu.ar/. UNLU. 2018.

[9] http://www.rae.es/. RAE. 2018.

[10] Obfuscation: A user's guide for privacy and protest. Brunton, Finn and Nissenbaum, Helen. 2015.

[11] Code obfuscation techniques for software protection. Cappaert, Jan. Katholieke Universiteit Leuven. 2012.

[12] Rajendran, Jeyavijayan and Sam, Michael and Sinanoglu, Ozgur and Karri, Ramesh. Security Analysis of Integrated Circuit Camouflaging. Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. 2013.

[13] A taxonomy of obfuscating transformations. Collberg, Christian and Thomborson, Clark and Low, Douglas. 1997.

[14] Edgardo Bernardis, Hernán Bernardis, Mario Berón, Germán Montejano. “Seguridad en Servicios Web”. XIX Workshop de Informática y Ciencias de la Computación (WICC). Buenos Aires, Argentina. Abril de 2017.

[15] WSDL Specification for W3C https://www.w3.org/TR/wsdl..

[16] Pananya Sripairojthikoon, Twittie Senivongse. “Concept-Based Readability Measurement and Adjustment for Web Services Descriptions”. ICACT Transactions on Advanced Communications Technology (TACT) Vol. 3, Issue 1, January 2014.

[17] Malenfant, J., Jacques, M., & Demers, F. N. (1996, April). A tutorial on behavioral reflection and its implementation. In Proceedings of the Reflection (Vol. 96, pp. 1-20).

[18] Ibrahim, B. M., & Hassan, M. F. (2015, May). A new customizable security framework for preventing WSDL attacks. In Mathematical Sciences and Computing Research (iSMSC), International Symposium on (pp. 24-29). IEEE.

[19] Mirtalebi, Arezoo, and Mohammad Reza Khayyambashi, “Enhancing Security of Web Service against WSDL Threats,” 2nd IEEE International Conference on Emergency Management and Management Sciences (ICEMMS), pp. 920-923, IEEE, 2011.

[20] Shahgholi, Narges, Mehran Mohsenzadeh, Mir Ali Seyyedi, and Saleh Hafez Qorani, “A New SOA Security Framework Defending Web Services Against WSDL Attacks,” IEEE 3rd Inernational Conference on Privacy, Security, Risk and Trust (PASSAT), pp. 1259-1262, IEEE, 2011.

[21] AFIP. https://wswhomo.afip.gov.ar/wsfev1/service.asmx. 2018.