Universidad de Buenos Aires
Facultad de Ciencias Económicas

Universidad de Buenos Aires

Facultad de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final

Titulo

Análisis de Efectividad de Herramientas de Penetration Testing en Web Services

Autor

Ing. Johan Javier Pizarro Martínez

Tutor

Dr. Juan Pedro Hecht.

Año de Presentación

2018

Cohorte

2015

Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

Johan Javier Pizarro Martínez

_____

Firma

95431349

_____

DNI

Resumen

Este trabajo de especialización está enfocado en un estudio detallado de las herramientas comerciales y no comerciales que actualmente permiten llevar a cabo el penetration testing sobre servicios web basados en el protocolo SOAP.

Las herramientas y principales vulnerabilidades seleccionadas en este trabajo de especialización están basadas en las recomendaciones dadas por la comunidad de OWASP y dentro de ella personal experto en el tema de seguridad de web services.

Por último este trabajo da a conocer resultados obtenidos de pruebas prácticas realizadas a varias de las herramientas con el fin de determinar el grado de efectividad en el momento de encontrar vulnerabilidades y las contramedidas que recomiendan para mitigarlas. Además se da a conocer tablas comparativas de cada una de las herramientas como resultado de varias pruebas realizadas por expertos de la comunidad OWASP.

Palabra Claves: SOAP, OWASP, Protocolo, Penetration Testing, Web Services.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

Con la llegada de la tecnología móvil cada vez son más las empresas que exponen sus servicios web a la red, por consiguiente muchas compañías hoy por hoy están apostando cada vez más a la construcción de aplicaciones móviles que ayuda a todo tipo de usuario a acceder a la información en tiempo real, esto ha implicado el desarrollo y exposición de muchos Web Services en las redes sin tener en cuenta la inseguridad que estos pueden llegar a tener en el momento de comunicarse con otros servicios de aplicaciones internas de las compañías o de aplicaciones externas a estas. En consecuencia, los Web Services se han convertido en el principal vector de ataques debido a que la información que se transmite en la comunicación es sensible y escasa de mecanismos de seguridad que impidan el acceso y manipulación de intrusos que quieran disponer de estos datos para fines delictivos.

Por todo ello cada vez más son las empresas que pierden dinero debido a ataques, de manera que se ve la necesidad de realizar pruebas de seguridad que permitan garantizar la disponibilidad, autenticidad e integridad de los Servicios Web mediante la selección de efectivos programas especializados que ayuden a identificar los principales defectos de seguridad, para luego poder priorizarlos y con base a ello generar un informe de análisis de vulnerabilidades y así mismo dar conocer recomendaciones sobre como mitigarlas.

2. ¿Qué se entiende por SOA?

SOA (Arquitectura orientada a servicios) es un marco de trabajo conceptual que establece una estructura de diseño para la integración de aplicaciones, que permite a las organizaciones unir los objetivos de negocio, en cuanto a flexibilidad de integración con sistemas legados y alineación directa a los procesos de la empresa, con la infraestructura de TI. Esto permite la reducción de costos de implementación, innovación de servicios a clientes, adaptación ágil ante cambios y reacción temprana ante la competitividad, ya que, combinan fácilmente las nuevas tecnologías con aplicaciones independientes, permitiendo que los componentes del proceso se integren y coordinen de manera efectiva y rápida. (Vera, 2013) [30].

3. ¿Qué es un Web Services?

Los servicios Web son tecnologías que utilizan un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Permiten también la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración, tanto dentro de las organizaciones como con socios comerciales. (W3C, s.f.) [27]

Los web services permiten una comunicación más sencilla entre diferentes sistemas esto es posible con ayuda del protocolo de mensajería estandarizada denominada XML que permite el intercambio de información con otros servicios web.

Para el intercambio de datos los web services usan el protocolo SOAP (Simple Object Access) y para la descripción de las funcionalidades el lenguaje WSDL (Web Services Description Language).

La gran ventaja de la comunicación de los web services es que no dependen de ningún sistema operativo o lenguaje de programación en cual este instalada o construida una aplicación, esto explica que un sistema de información de órdenes de compra pueda transferir información a un sistema contable.

4. Diferencias de los Web Services y Aplicaciones Web

"Una aplicación web es una aplicación que accede a través de un navegador web que se ejecuta en la máquina de un cliente, por el contrario un web services es un sistema de software que permite que diferentes máquinas interactúen entre sí a través de una red" (Panda, Web Services Penetration Testing Part 1, 2013) [22].

La mayoría de las veces, los web services no necesariamente tienen una interfaz de usuario ya que se utiliza como un componente en un sistema, mientras que una aplicación web es completa con una GUI[1].

5. Principales Estándares utilizados en Web Services [33].

Los web services usan diferentes estándares y basados en las definiciones de la W3C se detallan los siguientes:

5.1 XML (Extensible Markup Language)

En su traducción al castellano "Lenguaje de Etiquetado Extensible", es un lenguaje similar a HTML, pero con la diferencia que es un lenguaje para describir datos y no mostrarlos. XML utiliza un formato basado en marcas y

[1] **GUI:** Interfaz Gráfica de Usuario.

en etiquetas que permite la lectura de datos a través de diferentes aplicaciones.

5.2 SOAP (Simple Object Access Protocol)

En su traducción al castellano “Protocolo Simple de Acceso a Objetos”, Es un protocolo para intercambiar mensajes en aplicaciones que se encuentran en entornos descentralizados y distribuidos. SOAP es utilizado para el intercambio de información a través del protocolo HTTP por medio de mensajes codificados en XML, además es independiente de cualquier sistema permitiéndole poder comunicar aplicaciones que se encuentren codificadas en cualquier lenguaje [32].

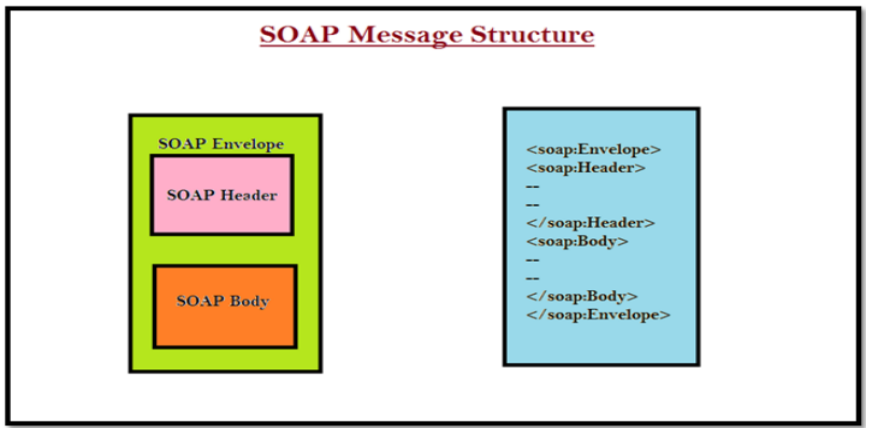


Figura 1. Estructura de Mensaje SOAP

Fuente: (Panda, Web Services Penetration Testing Part 1, 2013)[22]

5.3 WSDL (Web Service Description Language)

En su traducción al castellano “Lenguaje de Descripción de Servicios”, es un lenguaje basado en XML que sirve para describir los servicios Web y cómo acceder a ellos. El WSDL es parte integral de UDDI y parte del registro global de XML, en otras palabras es un estándar de uso público.

5.4 UDDI (Universal Description Discovery and Integration)

"Es un directorio público que es diseñado para que empresas puedan almacenar y publicar de forma estructurada sus web services, esto ayuda a que otras empresas puedan conocerlos y puedan utilizar estos web services que publican". (Gonzáles, 2004) [5].

5.5 UBR (Universal Business Registry)

Es un tercer intermediario dentro de la estructura de comunicación de los servicios web (UDDI, WSDL y SOAP) que se encarga de facilitar la comunicación entre el consumidor y proveedor. UBR proporciona una lista de servicios disponibles, el consumidor de servicios selecciona uno o más servicios disponibles.

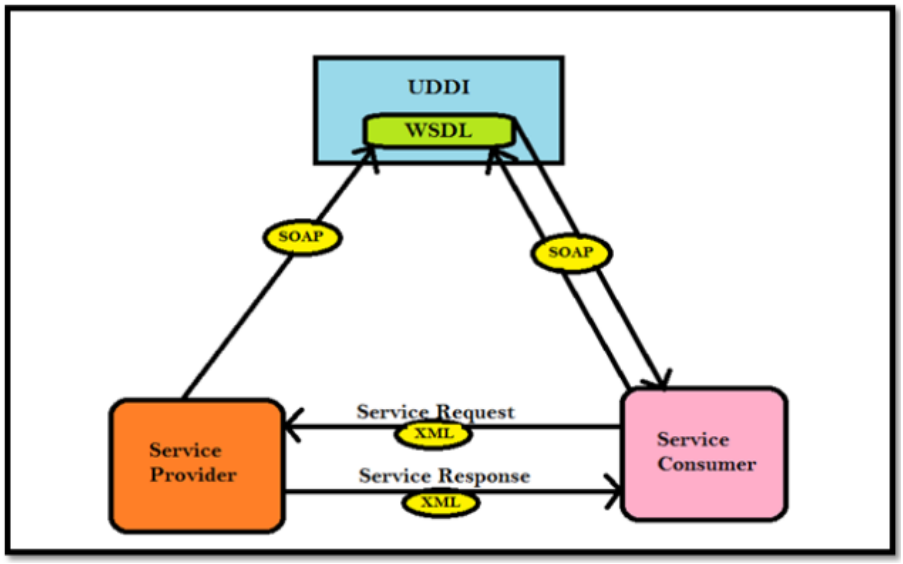


Figura 2. Principales Estándares de SOAP

Fuente: (Panda, Web Services Penetration Testing Part 1, 2013) [22]

5.6 ¿Por qué la Importancia de realizar un efectivo análisis de Vulnerabilidades En Web Services? [30]

Las tecnologías como XML y SOAP permiten conectar y comunicar los sistemas haciendo uso del protocolo HTTP un protocolo ampliamente soportado y aceptado por los servidores web de las compañías esto ha

ocasionado que los proxys tradicionales no inspeccionen el tráfico XML y SOAP en busca de ataques.

La tecnología XML tiene muchas ventajas debido a que posee un esquema basado en texto que ayuda a describir información de forma estructurada y por otro lado ayuda intercambiar datos entre sistemas empresariales distintos, de donde se infiere que es la base para un gran número de especificaciones SOA.

El protocolo XML está basado en texto plano de modo que puede ser legible por humanos de manera que la tecnología tiende a ser vulnerable ante amenazas y exposiciones de seguridad, además no es protegida por ningún dispositivo de seguridad que evite nuevos ataques basados en XML y muchas compañías deshabilitan las validación de XML por costos de performance.

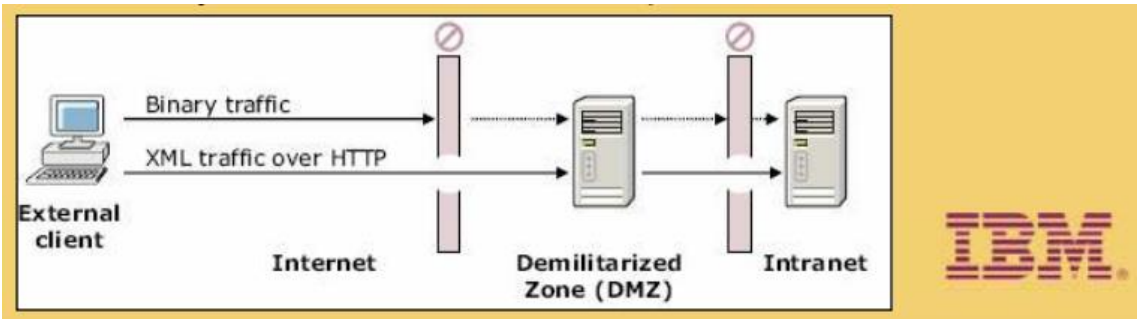


Figura 3. Tráfico de XML basado de HTTP

Fuente: (Szuster) [30]

El realizar un análisis de vulnerabilidades bien sea manual o con aplicaciones es el principal mecanismo para poder disminuir el riesgo a posibles ataques en la arquitectura de servicios web.

6.0 ¿Qué es Owasp?

El proyecto abierto de seguridad en aplicaciones web OWASP tiene una guía de pruebas de seguridad que es considerado como un marco de desarrollo de pruebas que le permite a la comunidad crear sus propias programas de pruebas de seguridad y con base a ello evaluar la seguridad de entidades organizacionales. (About The Open Web Application Security Project, 2017)[1]

6.1 ¿Qué es OWASP Cheat Sheets? [19]

De acuerdo a la definición dada por OWASP Cheat Sheet Series (2017) es una serie que fue creada para proporcionar una colección concisa de información de alto valor sobre temas específicos de seguridad de aplicaciones web. Estas hojas de trucos fueron creadas y aplicadas por varios profesionales de seguridad que tienen experiencia en temas específicos de igual modo proporciona una guía de seguridad excelente en un formato fácil de leer.

7.0 Penetration Testing en Web Services

Las pruebas de intrusión en servicios web están categorizadas en 2 tipos: Pruebas de caja negra y pruebas de caja gris.

7.1 Pruebas de Caja Negra

Este tipo de pruebas son realizadas cuando no existe información sobre el servicio web, para ello estas pruebas se enfocan en el punto de vista de un atacante e intenta recopilar la mayor parte información para así poder realizar un análisis de seguridad simulando los diferentes ataques bien sea por medios manuales o automáticos con ayuda de herramientas automatizadas. (Moral, 2014) [14]

Unos de los primeros pasos que recomienda OWASP para probar servicios web son las pruebas de caja negra y posteriormente un análisis de pruebas de caja gris.

7.2 Pruebas de Caja Gris

Las pruebas de caja gris requieren documentación detallada y nivel avanzado de una aplicación, con la finalidad de definir casos de pruebas. Este tipo de pruebas son beneficiosas porque hace uso de técnicas de pruebas de caja negra y las combinan con las de caja blanca debido a que se basan en casos de pruebas realizados por personas que conocen el código fuente de una aplicación. (Moral, 2014) [16]

7.3 Obtención de Información en Servicios Web

Cuando se requiere probar la seguridad de un web services por lo general los clientes no tienen un completo conocimiento del servicio por lo que solo conocen la ruta WSDL, por este motivo se utilizan la técnica de obtención de información o también llamada Fingerprinting en servicios Web el cual consiste en determinar la ruta WSDL del servicio al cual se le va a realizar las pruebas, y en ocasiones esta ruta no se conoce por lo que hay que recurrir al servicio UDDI con el fin de encontrar el servicio en concreto, esta fase es reconocida por OWASP como “Conocimiento cero”. (OWASP Testing Guide v3 Table of Contents, 2013) [20]

7.3.1 WSDL Google Hacking Attack [19]

De acuerdo a las recomendaciones que sugiere la guía de Web Service Security Testing Cheat Sheet, (2016) la forma más sencilla de encontrar web services públicos es haciendo uso del Hacking en buscadores ingresando cadenas en los buscadores tales como: Inurl:jws?wsdl, Inurl:asmx?wsdl, Inurl:aspx?wsdl, ,Inurl:ascx?wsdl,Inurl:ashx?wsdl, Inurl:dll?wsdl, Inurl:exe?wsdl, Inurl:php?wsdl, Inurl:pl?wsdl, Inurl:?wsdl, Filetype:jws,

Filetype:asmx, Filetype:ascx, Filetype:aspx, Filetype:ashx, Filetype:dll, Filetype:exe, Filetype:php y Filetype:pl.

7.3.2 Ejemplo Práctico WSDL Google Hacking

Ingresando a www.google.com, es posible hacer uso del comando filetype:wsdl, este comando arroja una lista de resultados de archivos wsdl, para este caso se selecciona el siguiente servicio “eds:wsdl”.

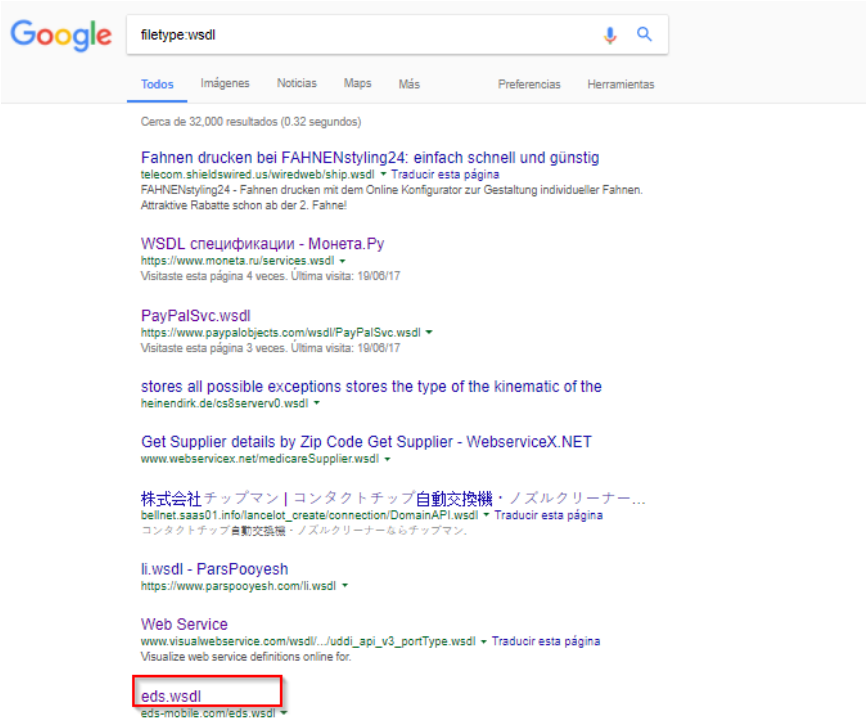


Figura 4. Google Hacking

Una vez dentro del archivo podemos obtener la ruta WSDL que para este caso es http://eds-mobile.com/eds.wsdl , el cual describe el servicio en formato XML:

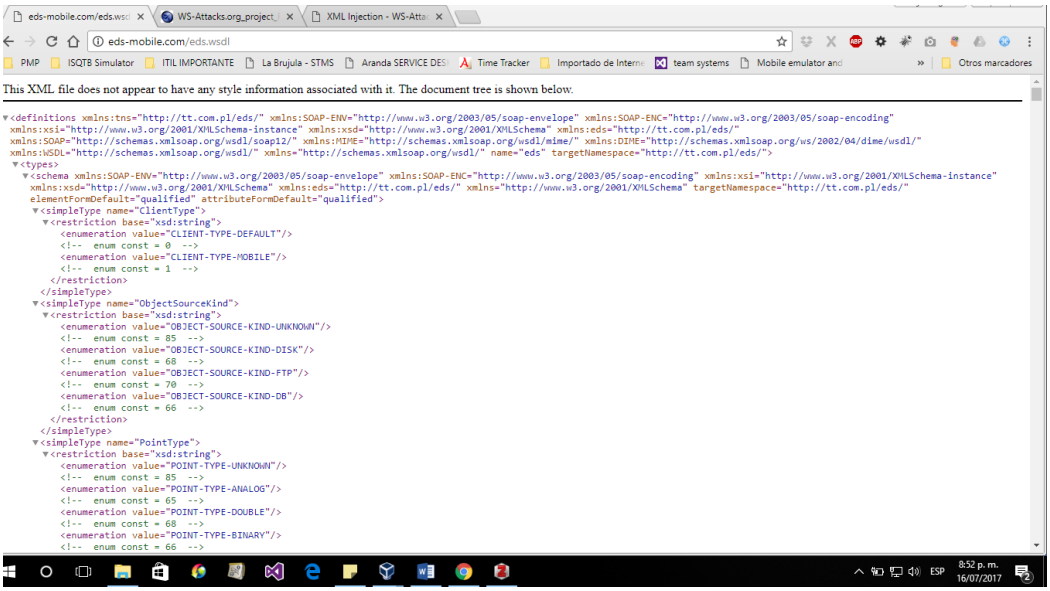


Figura 5. Archivo WSDL

Para este análisis de obtención de información existen herramientas (toolkits), que permite realizar un efectivo análisis de obtención de información, estas herramientas actúan como un consumidor de servicios web y realizan peticiones al UBR que ayuda a brindar una lista de servicios disponibles.

7.3.3 WSDL Scanning

En esta clase de pruebas se tiene conocimiento de la ruta del archivo WSDL y en base a esta ruta se trata de conocer nuevos métodos e interfaces del servicio web.

7.3.4 Ejemplo Práctico WSDL Scanning

Volviendo hacer uso del WSDL http://eds-mobile.com/eds.wsdl previamente encontrado durante la ejecución del comando filetype:wsdl, y con el fin de conocer los métodos podemos hacer uso de aplicaciones que permiten hacer uso del protocolo SOAP.

Estos son algunos de los métodos u operaciones que contiene el archivo WSDL http://eds-mobile.com/eds.wsdl

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:eds="http://t
   <soap:Header/>
   <soap:Body>
      <eds:addEvents>
         <eds:authString>?</eds:authString>
         <!--1 or more repetitions:-->
         <eds:newEvents>
            <eds:category>?</eds:category>
            <eds:type>?</eds:type>
            <eds:priority>?</eds:priority>
            <eds:message>?</eds:message>
```

Figura 6. Petición SOAP

7.3.5 Principales Vulnerabilidades [19]

De acuerdo a OWASP Cheat Sheet Series, (2017) estas son las principales vulnerabilidades que se descubren haciendo uso de herramientas efectivas de penetration testing en web services:

7.3.6 Fuzzing Scan

"El fuzzing es una técnica dentro de las pruebas de seguridad automatizadas que consiste en enviar datos inválidos de forma aleatoria a un sistema de destino crítico en seguridad que puede llegar a ser explotado con intenciones maliciosas" (Fuzzing Scan, 2017)[14]. Este tipo de pruebas se pueden llevar a cabo durante un periodo de tiempo prologado y lo que se busca es encontrar fallos o inestabilidades del sistema destino, más adelante se llevaran ejemplos prácticos de este ataque.

7.3.7 XSS

Cross-Site Scripting, comúnmente abreviado XSS es uno de los ataques más comunes sobre aplicaciones web, este ataque consiste en la

inyección de código script (javascript, VBScript, o Flash) dentro de las páginas web por otros usuarios. Este ataque tiene similitudes con el SQL Injection y el Xpath Injection, pero en este caso el objetivo del ataque son los usuarios del sitio web en lugar del propio sitio web. (Cross-site Scripting (XSS), 2016)[9]

7.3.8 SQL INJECTION

Un ataque por inyección SQL consiste en la inserción o "inyección" de una consulta SQL por medio de los datos de entrada desde el cliente hacia la aplicación. Un ataque por inyección SQL exitoso puede leer información sensible desde la base de datos, modificar la información (Insert/ Update/ Delete), ejecutar operaciones de administración sobre la base de datos (tal como parar la base de datos), recuperar el contenido de un determinado archivo presente sobre el sistema de archivos del DBMS[2] y en algunos casos emitir comandos al sistema operativo (Owasp, 2012) [17]. El resultado de esto es la obtención de información y por último el acceso a la aplicación.

7.3.9 XPath Injection

XPath es un lenguaje que permite construir expresiones que ayudan a consultar determinadas partes de un documento XML. Posiblemente podríamos compararlo con el lenguaje SQL que realiza consultas sobre una base de datos.

Los parámetros dentro del cuerpo SOAP sirven como datos de entrada en las consultas xpath de acuerdo a esto si una entrada del usuario no está siendo validada un atacante puede aprovechar esta vulnerabilidad y modificar una consulta xpath de ahí que pueda tener el control de obtener información de cómo se encuentra estructurada la información XML de modo que pueda obtener acceso a datos privados y pueda también elevar privilegios en el sitio

[2] **DBMS (Data Base Management System),** son las siglas en inglés para los Sistema de gestión de bases de datos.

web si los datos XML se utiliza para la autenticación. (Falkenberg, Xpath Injection, 2015)[12].

Este tipo de ataques suelen ser más peligrosos que las inyecciones SQL debido que los documentos XML no tienen ningún mecanismo de control de acceso.

7.3.10 Requisitos Previos del Ataque [13].

Estos son los requisitos previos que se deben tener en cuenta para realizar este tipo de ataques:

"El atacante deberá conocer el endpoint del servicio web, porque de lo contrario no podrá acceder al servicio web" (Falkenberg, Xpath Injection, 2015).

"El atacante deberá conocer los metadatos como el archivo WSDL del servicio web" (Falkenberg, Xpath Injection, 2015) .

"El atacante deberá poder acceder al servicio web dentro su dominio de red" (Falkenberg, Xpath Injection, 2015).

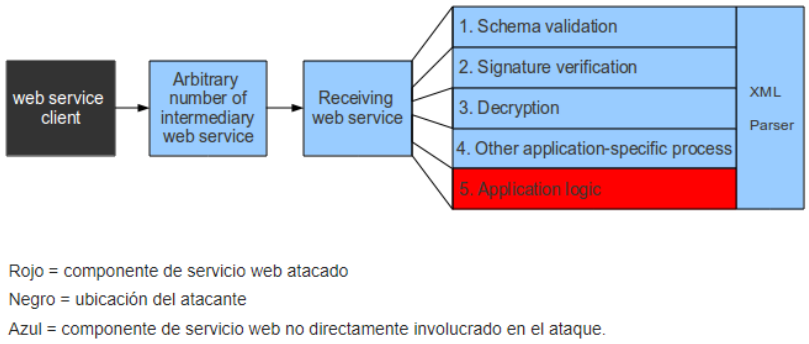


Figura 7. Representación del Ataque Xpath Injection

Fuente: (Falkenberg, Xpath Injection, 2015)

Este es un ataque directo a la lógica de la aplicación, el cual permite realizar consultas no previstas por el desarrollador.

7.3.11 XML Bomb (xDOS) [30].

La principal función de este ataque es deshabilitar o disminuir la performance de un Web Services a través de un peligroso mensaje en formato XML creado por un usuario malicioso. Un XML Bomb explota el software que parsea o analiza un mensaje XML conocido como Document Type Definitiion DTD[3], causando que el software tenga que procesar una cantidad de datos que crece exponencialmente desenlazando en un ataque de denegación de servicios.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SOAP-ENV:Envelope [
  <!ELEMENT SOAP-ENV:Envelope ANY>
  <!ATTLIST SOAP-ENV:Envelope entityReference CDATA #IMPLIED>
  <!ENTITY x0 "foo">
  <!ENTITY x1 "&x0;&x0;" >
  <!ENTITY x2 "&x1;&x1;" >
  ...
  <!ENTITY x98 "&x97;&x97;" >
  <!ENTITY x99 "&x98;&x98;" >
] >
```

[3] **DTD** es un documento que define la estructura de un documento XML. El procesador XML utiliza la DTD para verificar si un documento es válido, es decir, si el documento cumple las reglas del DTD.

<SOAP:Envelope entityReference="&x99;" xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">

<SOAP:Body>

<keyword xmlns="urn:parasoft:ws:store">foo</keyword>

</SOAP:Body>

</SOAP:Envelope>

Cuando un servicio web recibe dicho mensaje, y si el analizador XML tiene habilitado el procesamiento DTD, hará que el servidor expanda la referencia de entidad x99; Esta referencia se define en la sección DTD que viene con el mensaje para que sea igual a & x98; repetido dos veces, que a su vez se define como & x97; repetido dos veces y así en el camino a x0. En otras palabras:

x0 = "foo"

x1 = "foofoo"

x2 = "foofoofoofoo"

…

x99 = "foofoofoofoofoo…foo" or "foo" repeated

299 = 633825300114114700748351602688

A medida que el servicio expande el valor de x, el tamaño de la cadena que contiene las palabras concatenadas "foo" alcanza niveles exponenciales, lo que hace que se consuma rápidamente. (XML security: Preventing XML bombs, 2006)

7.3.12 Medidas Preventivas.

Asegurarse de que los servidores comerciales u open source que se adquieran tengan deshabilitado el DTD o manualmente deshabilitarla. (XML security: Preventing XML bombs, 2006)

7.3.13 XML Injection [12].

Este ataque consiste en intentar inyectar varias etiquetas XML en un mensaje SOAP con el objetivo de modificar la estructura XML. Los ataques de XML Injection exitosos son lo que se aplican a través de operaciones restringidas de aplicaciones dependiendo del cual sea la operación ejecutada, se pueden violar varios objetivos de seguridad. Ejemplos típicos están:

> "Modificación de datos de pago - objetivo de seguridad violado: Integridad" (Falkenberg, XML Injection, 2015)

> "Acceso de administrador no autorizado - objetivo de seguridad violado: Control de acceso" (Falkenberg, XML Injection, 2015)

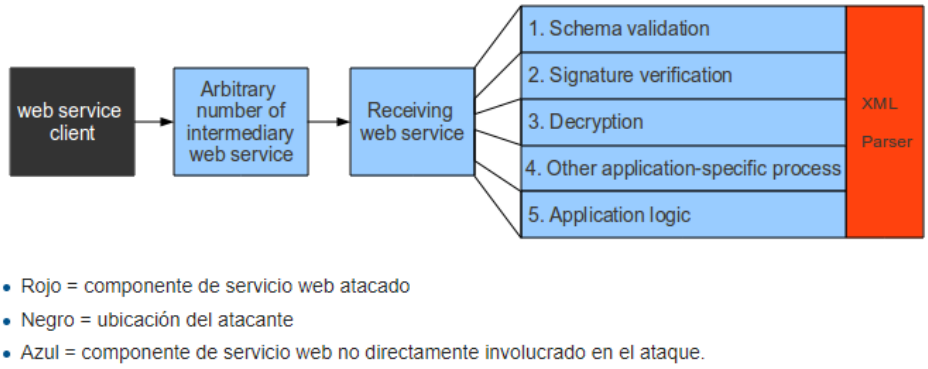
7.3.14 Requisitos Previos del Ataque.

Estos son los requisitos previos que se deben tener en cuenta para realizar este tipo de ataques:

> "El atacante deberá conocer el endpoint del servicio web, porque de lo contrario no podrá acceder al servicio web" (Falkenberg, XML Injection, 2015).

> "El atacante deberá conocer los metadatos como el archivo WSDL del servicio web" (Falkenberg, XML Injection, 2015).

> "El atacante deberá poder acceder al servicio web dentro su dominio de red" (Falkenberg, XML Injection, 2015).



- Rojo = componente de servicio web atacado
- Negro = ubicación del atacante
- Azul = componente de servicio web no directamente involucrado en el ataque.

Figura 8. Representación Gráfica XML Injection

Fuente: (Falkenberg, XML Injection, 2015) [12]

7.3.15 Ejemplo Práctico - XML Injection.

El archivo WSDL http://www.testfire.net/bank/ws.asmx?WSDL se usara para realizar una demostración de cómo se podría realizar un caso de prueba, para realizar este ataque.

Suponiendo que se desea realizar el pago de una transacción haciendo uso de la operación “TransferBalance” por medio de un mensaje SOAP:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envel
   <soapenv:Header/>
   <soapenv:Body>
      <ws:TransferBalance>
         <!--Optional:-->
         <ws:transDetails>
            <ws:transferDate>2017-07-22</ws:transferDate>
            <!--Optional:-->
            <ws:debitAccount>2</ws:debitAccount>
            <!--Optional:-->
            <ws:creditAccount>4958302233688005</ws:creditAccount>
            <ws:transferAmount>50000</ws:transferAmount>
         </ws:transDetails>
      </ws:TransferBalance>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 9. Petición de la Operación Transfer Balance

En la siguiente imagen se muestra la transacción de pago modificada por un atacante. Si el mensaje se lleva a cabo el atacante solo tiene que pagar 20 pesos en lugar de los 50000.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.altoromutual.c
   <soapenv:Header/>
   <soapenv:Body>
      <ws:TransferBalance>
         <!--Servicio Modificado Por Un Atacante-->
         <ws:transDetails>
            <ws:transferDate>2017-07-22</ws:transferDate>
            <!--Optional:-->
            <ws:debitAccount>2</ws:debitAccount>
        <!--<! Si el ataque se ejecuta exitosamente,
         la etiqueta <transferAmount> con su valor se sobrescribe por 20 -> -->
            <ws:creditAccount>123456<ws:transferAmount>20</ws:transferAmount><ws:creditAccount>123456
            </ws:creditAccount>
            <ws:transferAmount>50000</ws:transferAmount>
         </ws:transDetails>
      </ws:TransferBalance>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 10. Response de la Operación Transfer Balance

7.3.16 Medidas Preventivas – XML Injection.

Los archivos WSDL deben contener una descripción detallada de los elementos, atributos y tipos de datos utilizados.

```
▼<s:complexType name="MoneyTransfer">
  ▼<s:sequence>
     <s:element minOccurs="1" maxOccurs="1" name="transferDate" type="s:dateTime"/>
     <s:element minOccurs="0" maxOccurs="1" name="debitAccount" type="s:string"/>
     <s:element minOccurs="0" maxOccurs="1" name="creditAccount" type="s:string"/>
     <s:element minOccurs="1" maxOccurs="1" name="transferAmount" type="s:double"/>
   </s:sequence>
 </s:complexType>
▼<s:element name="TransferBalanceResponse">
  ▼<s:complexType>
    ▼<s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="TransferBalanceResult" type="tns:Transaction"/>
     </s:sequence>
   </s:complexType>
 </s:element>
▼<s:complexType name="Transaction">
  ▼<s:sequence>
     <s:element minOccurs="1" maxOccurs="1" name="Success" type="s:boolean"/>
     <s:element minOccurs="0" maxOccurs="1" name="Message" type="s:string"/>
   </s:sequence>
 </s:complexType>
```

Figura 11. Medidas Preventivas Ataque XML Injection

7.3.17 Validación Estricta del Esquema – XML Injection [13].

El usar la etiqueta <minOccurs> el número de ocurrencias del elemento “Cantidad de Transferencias” está limitado a 1, ya que por defecto el número máximo y mínimo de ocurrencias es 1. Entonces cualquier mensaje SOAP que intente violar este esquema será rechazado.

No es recomendable el uso de SAX[4] parser como contramedida. Esta contramedida funcionaria, pero al pasar a un nuevo framework que utilice un analizador basado en el DOM[5], el servicio web será vulnerable al ataque, es por este motivo que la recomendación es realizar una estricta validación del esquema (Falkenberg, Xpath Injection, 2015).

7.3.18 Soap Array Attack

Este ataque consiste en ingresar una matriz (array) de elementos dentro de un mensaje SOAP esto es debido a que el protocolo SOAP no limita el número de elementos dentro de una matriz de hecho esto se convierte en una principal vulnerabilidad que puede aprovechar un atacante para realizar un ataque de Denegación de Servicios DoS afectando la disponibilidad del servicio.

Para mencionar un breve ejemplo tenemos que un atacante envía más de mil elementos dentro de un mensaje SOAP y como resultado el servicio cargara esta información en memoria RAM antes de que el mensaje sea procesado por el analizador lo que implicará que la memoria del sistema se sature y de esta manera la disponibilidad del servicio se vea críticamente afectada.

[4] **SAX** son las siglas de "Simple API for XML", originalmente una API únicamente para el lenguaje de programación Java que después se convirtió en la API estándar de facto para usar XML en JAVA.

[5] **DOM**, define la manera en que objetos y elementos se relacionan entre sí en el navegador y en el documento.

7.3.19 Requisitos Previos del Ataque - Soap Array Attack [10]

"El atacante el endpoint del servicio web, el WSDL no es requerido esto se debe a que el ataque solo se concentrara en el analizador de XML (XML Parser)" (Falkenberg, Soap Array Attack, 2015).

"El servicio web víctima se debe encontrarse dentro del mismo dominio de red del ataque en caso contrario el ataque estaría limitado" (Falkenberg, Soap Array Attack, 2015).

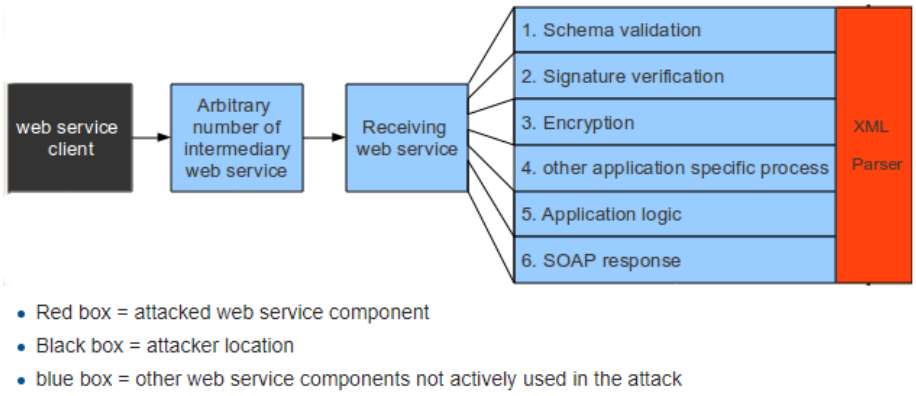


Figura 12. Representación Gráfica Soap Array Attack

Fuente: (Falkenberg, Soap Array Attack, 2015)[10]

7.3.20 Prueba de Caja Negra - Soap Array Attack

El archivo WSDL http://www.swi-prolog.org/pack/file_details/wsdl/examples/country.wsdl se usara para realizar una demostración de cómo se podría realizar un caso de prueba, para realizar este ataque.

Suponiendo que se desea consultar países por códigos haciendo uso de la operación “getCountryByCountryCode” tenemos que si una atacante quiere realizar este tipo de ataques puede declarar un array[6] en el mensaje SOAP con un millón de códigos.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:web="http://www.web
   <soapenv:Header/>
   <soapenv:Body>
      <web:GetCountryByCountryCode>
       <ns1:FunctionWithArrayInput xmlns:ns1="...">
       <DataSet xsi:type="soapenc:Array"
            soapenc:arrayType="xsd:string[100000000000]">
        <web:CountryCode xsi:type="xsd:string">20</web:CountryCode>
        </DataSet>
        </ns1:FunctionWithArrayInput>
      </web:GetCountryByCountryCode>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 13. Request SOAP Array Attack

7.3.21 Medidas Preventivas - Soap Array Attack

“Este ataque puede ser detenido aplicando estrictas validaciones de esquema[7], esto implica validar el número de elementos permitidos en horas de mostrar una respuestas en el response de un servicio”. (Falkenberg, Soap Array Attack, 2015) [10]

[6] **Array** es un conjunto ordenado en una estructura de filas y columnas.

```
<!-- start excerpt .. -->
<simpleType name="phoneNumber" base="string"/>

<element name="ArrayOfPhoneNumbers">
  <complexType base="SOAP-ENC:Array">
    <element name="phoneNumber" type="tns:phoneNumber" maxOccurs="10"/>
  </complexType>
  <anyAttribute/>
</element>
<!-- end excerpt... -->
```

Figura 14. Contramedidas para SOAP Array Attack

Fuente: (Falkenberg, Soap Array Attack, 2015)

7.3.22 Soap Action Spoofing

En los servicios web las solicitudes (Request) contienen algún tipo de operación que es ejecutada por la lógica de la aplicación, esta operación se encuentra en el primer elemento hijo del body del mensaje SOAP, cabe mencionar que mediante el transporte de un mensaje SOAP a través del protocolo HTTP el estándar SOAP permite el uso de un encabezado (Header) HTTP denominado SOAPAction el cual almacena el nombre de la operación ejecutada y de acuerdo ello informa al servicio web receptor que información está contenida en el cuerpo del mensaje SOAP del servicios web cliente todo esto ayuda a que el servicio web receptor no tenga que hacer un análisis XML (XML Parsing). Así que esta optimización en la comunicación de servicios web puede ser una vulnerabilidad el cual un atacante puede aprovechar de modo que pueda modificar el header SOAPAction durante la comunicación de dos servicios web y como resultado de esto aquellas operaciones entre un cliente de servicios web y el receptor pueden verse comprometidas [11].

7.3.23 Requisitos Previos del Ataque - Soap Action Spoofing [11].

El atacante conoce los endpoint de los servicios web, de lo contrario no podría acceder al servicio web.

"El atacante también tiene acceso al archivo WSDL" (Falkenberg, SOAPAction Spoofing, 2015)

"El acceso al servicio web se encuentra dentro del mismo dominio de red del atacante". (Falkenberg, SOAPAction Spoofing, 2015)

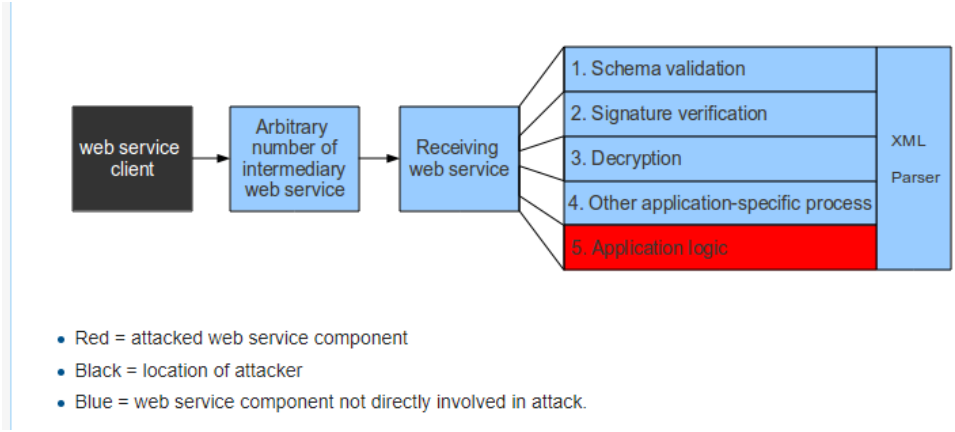


Figura 15. Representación Gráfica Ataque SOAP Action Spoofing

Fuente: (Falkenberg, SOAPAction Spoofing, 2015)

7.3.24 Ejemplo Práctico- Test Manual - Soap Action Spoofing

Suponemos que tenemos un servicio web vulnerable a SOAPAction Spoofing que consta de dos operaciones "createUser" y "deleteAllUsers". Además, el servicio web está protegido por una puerta de enlace (GateWay), deteniendo todas las llamadas "deleteAllUsers". La operación "createUser" puede ser ejecutada por cualquier persona. El "deleteAllUsers" sólo puede ser ejecutado por usuarios autorizados que estén directamente conectados al servicio web sin la pasarela intermedia.

```
POST /service HTTP/1.1
Host: myHost
SOAPAction: "createUser"

<Envelope>
  <Header />
  <Body>
    <createUser>
      <login>johndoe</login>
      <pwd>secret</pwd>
    </createUser>
  </Body>
</Envelope>
```

Figura 16. Request Soap Action Spoofing

Fuente: (Falkenberg, SOAPAction Spoofing, 2015)[11]

El atacante se encuentra delante de la puerta de enlace. Por lo tanto, no debería poder ejecutar el método "deleteAllUsers".

```
POST /service HTTP/1.1
Host: myHost
SOAPAction: "deleteAllUsers"

<Envelope>
  <Header />
  <Body>
    <createUser>
      <login>johndoe</login>
      <pwd>secret</pwd>
    </createUser>
  </Body>
</Envelope>
```

Figura 17. Response Soap Action Spoofing

Fuente: (Falkenberg, SOAPAction Spoofing, 2015)[11]

7.3.25 Medida Preventiva

"Si no se requiere, el atributo SOAPAction debe estar desactivado. Si es necesario, la operación dentro de SOAPAction y SOAP siempre debe compararse antes de ejecutar cualquier operación. Cualquier desajuste

debe considerarse como un ataque" (Falkenberg, SOAPAction Spoofing, 2015) [11].

7.4 Testing De Seguridad Automatizadas en Web Services

Las herramientas para realizar pruebas de seguridad sobre servicios web juegan un papel muy importante en horas de realizar un análisis efectivo para poder identificar el mayor número de vulnerabilidades. Desafortunadamente al día de hoy las herramientas de pentesting sobre aplicaciones web superan en cantidad a las herramientas de pruebas de intrusión para servicios web. La mayoría de aplicaciones que permiten realizar pruebas sobre servicios web están diseñadas para garantizar la calidad y no para realizar pruebas de seguridad.

El análisis de efectividad de herramientas de pruebas de intrusión sobre servicios web se llevara a cabo usando la guía y marco de desarrollo de pruebas de seguridad OWASP, apoyado en la guía de seguridad Web Service Security Testing Cheat Sheet de la serie OWASP Cheat Sheet y tomando como base el estándar para los medios de pagos con tarjeta de crédito el PCI DSS. (ASV Program Guide v3.0, 2017, pág. 25) [19][20][3].

Seleccionando las herramientas que OWASP Cheat Sheets recomienda para las pruebas de seguridad en Web Services se lleva a cabo algunos ejemplos prácticos con el fin de analizar el grado de efectividad. (OWASP Cheat Sheet Series, 2017) [19].

7.4.1 SoapUI [23]

Actualmente existen herramientas que permiten llevar a cabo el penetration testing automatizado en web services, entre ellas se puede

mencionar SoapUI by SmartBear siendo una de las primeras herramientas que recomienda la lista de OWASP Cheat Sheets.

SoapUI es una herramienta que se encuentra desarrollada en java y permite realizar pruebas sobre aplicaciones con arquitectura orientadas a servicios (SOA) y transferencia de estados representacional (REST). La aplicación soporta diferentes protocolos como SOAP, REST, HTTP, JMS, AMF y JDBC. La herramienta posee una versión de código abierto y una versión comercial SoapUI Pro. La principal ventaja de la versión comercial sobre la versión gratuita es el poder armar un plan de pruebas de seguridad automatizado, versión que se pondrá en prueba para analizar su efectividad en la detección de fallas de seguridad.

La aplicación SoapUI Pro en su última versión viene incluida como un componente dentro de Ready! Api 2.0.2 actualmente es la suite de pruebas desarrollada por la compañía SmartBear. Esta aplicación tiene varios componentes que anteriormente hacían parte de la aplicación SoapUI entre ellos están Secure Pro para las pruebas de seguridad sobre los diferentes web services basados en Rest y Soap, por otro lado esta SoapUI NG es el componente que permiten la generación de datos de pruebas, como componente también esta LoadUI NG encargado de realizar pruebas de performance sobre web services y por ultimo esta ServiceV que permite virtualizar las API JMS.

7.4.2 Pruebas de Black Box Con SoapUI

SoapUI permite realizar este tipo de pruebas con las dos versiones SoapUI Pro para pruebas automatizadas y SoapUI Free para pruebas manuales.

7.4.3 Ejemplos Prácticos

A continuación se realizaran ejemplos prácticos de la herramienta para las dos fases de pruebas automatizadas.

Estos casos de prueba prácticos son realizados mediante el web services http://www.testfire.net/bank/ws.asmx del el sitio web http://www.testfire.net/ , sitio web demo creado por IBM que permite demostrar la efectividad de las herramientas en la detección de vulnerabilidades de aplicaciones web. (IBM, 2017)[15].

7.4.4 Test Automatizado SoapUI Pro

Dentro Ready! API se encuentra el componente que permite realizar un penetration testing sobre web services, para el ejemplo práctico se hará uso de este componente y se usara el archivo WSDL http://www.testfire.net/bank/ws.asmx?WSDL el cual servirá para realizar un análisis de seguridad.

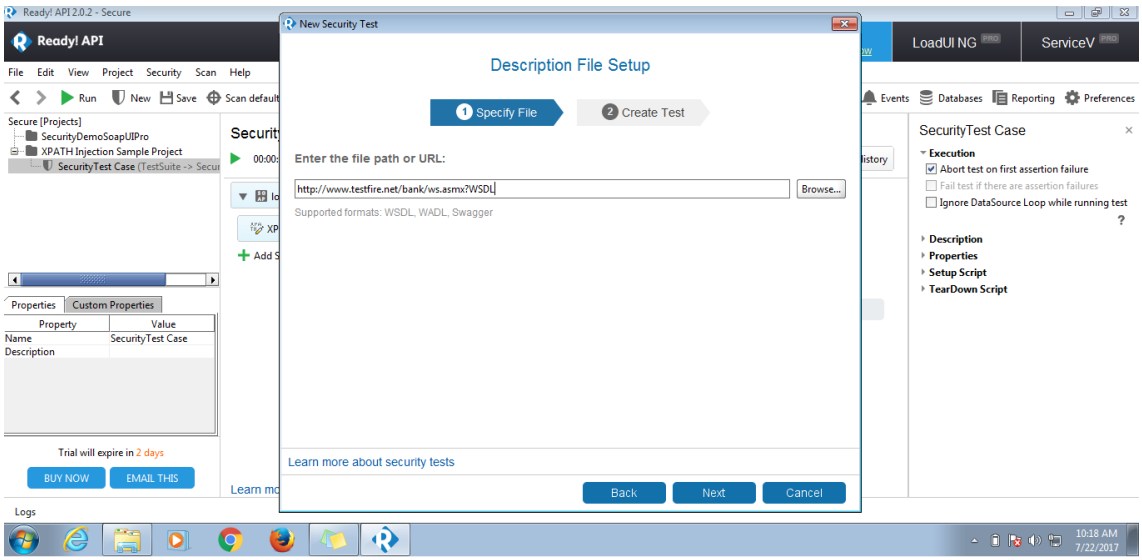


Figura 18. Interfaz SOAP UI Pro

El componente secure Pro permite realizar pruebas de seguridad sobre web services basados sobre protocolo SOAP, y permite crear un proyecto en el cual es posible seleccionar un grupo de pruebas de seguridad, lo que quiere decir que permite armar un plan de casos de pruebas basados en los siguientes casos: Boundary scan, Malicious Attachment, Cross site scripting, Fuzzing scan, Invalid Types, Malformed XML, Malicious Attachment, SQL Injection, XML Bomb, Xpath Injection, Weak Authentication.

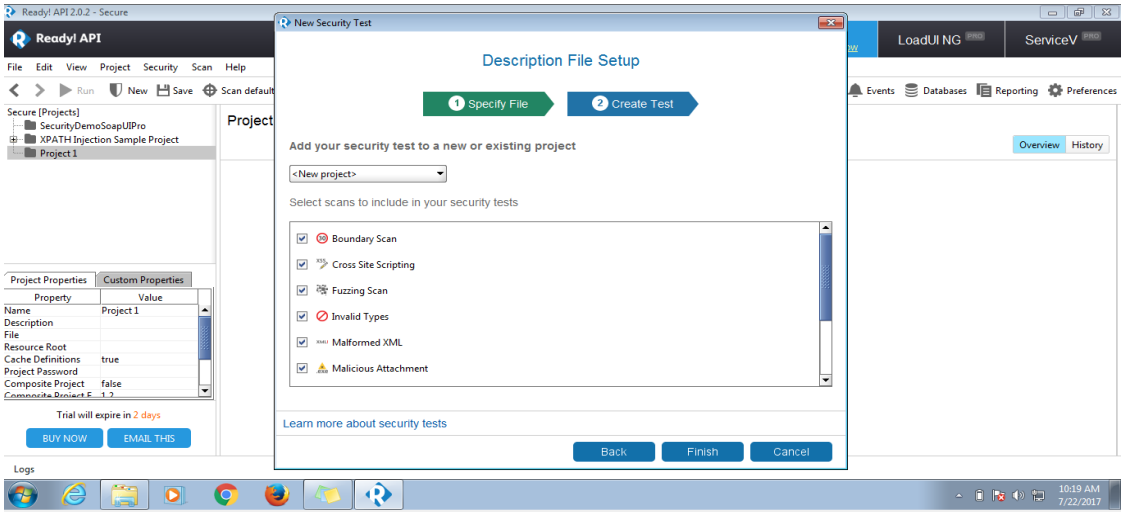


Figura 19. Pruebas de Seguridad Ready Api

La aplicación permite realizar un análisis de seguridad por cada operación del servicio y detallar un log de todas las transacciones que va realizando durante todas las pruebas de intrusión.

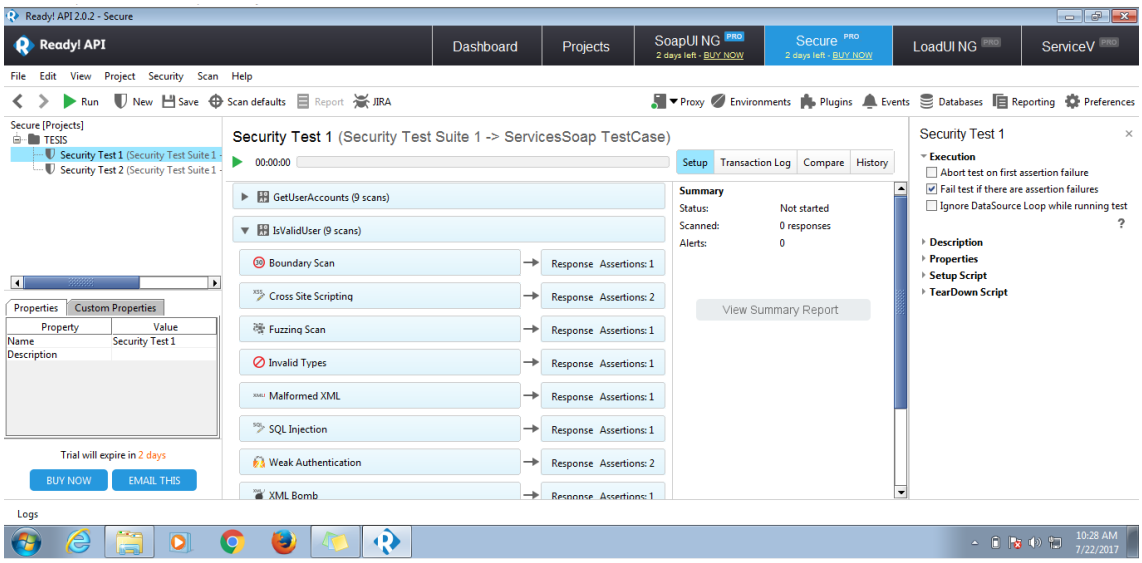


Figura 20. Ejecución de Pruebas Ready Api

Una vez que termina todo el plan de pruebas de seguridad previamente armado la aplicación genera un detallado resumen como resultado de las pruebas indicando el número de Issues (Fallas) o bugs de seguridad encontrados por cada uno de los ataques. Para este caso práctico arrojo un total de 62 Vulnerabilidades encontradas como producto de 1227 barridos que realizado la aplicación en las operaciones del servicio previamente seleccionados.

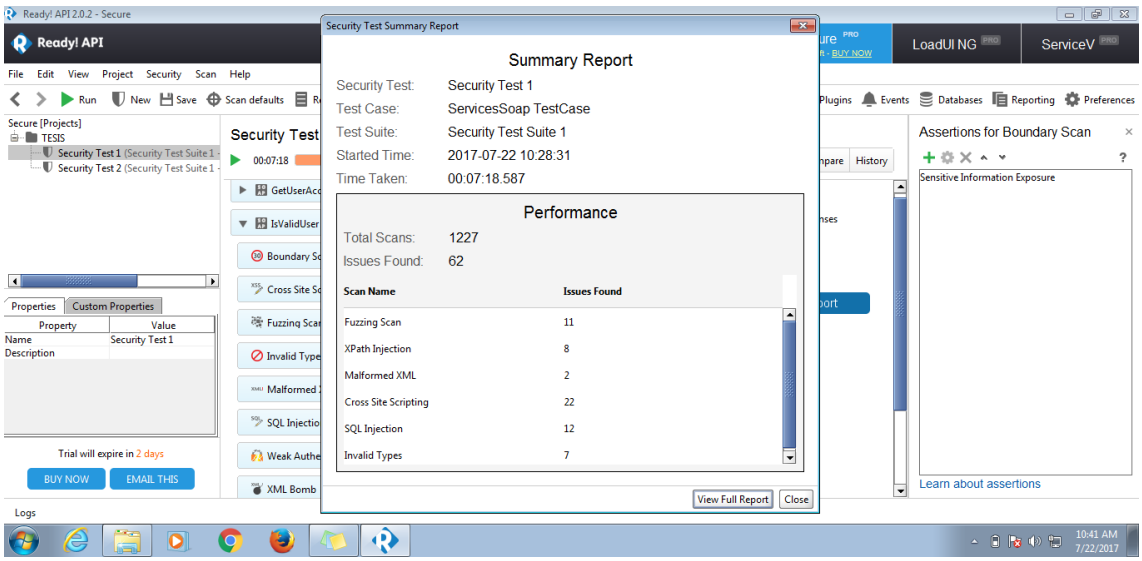
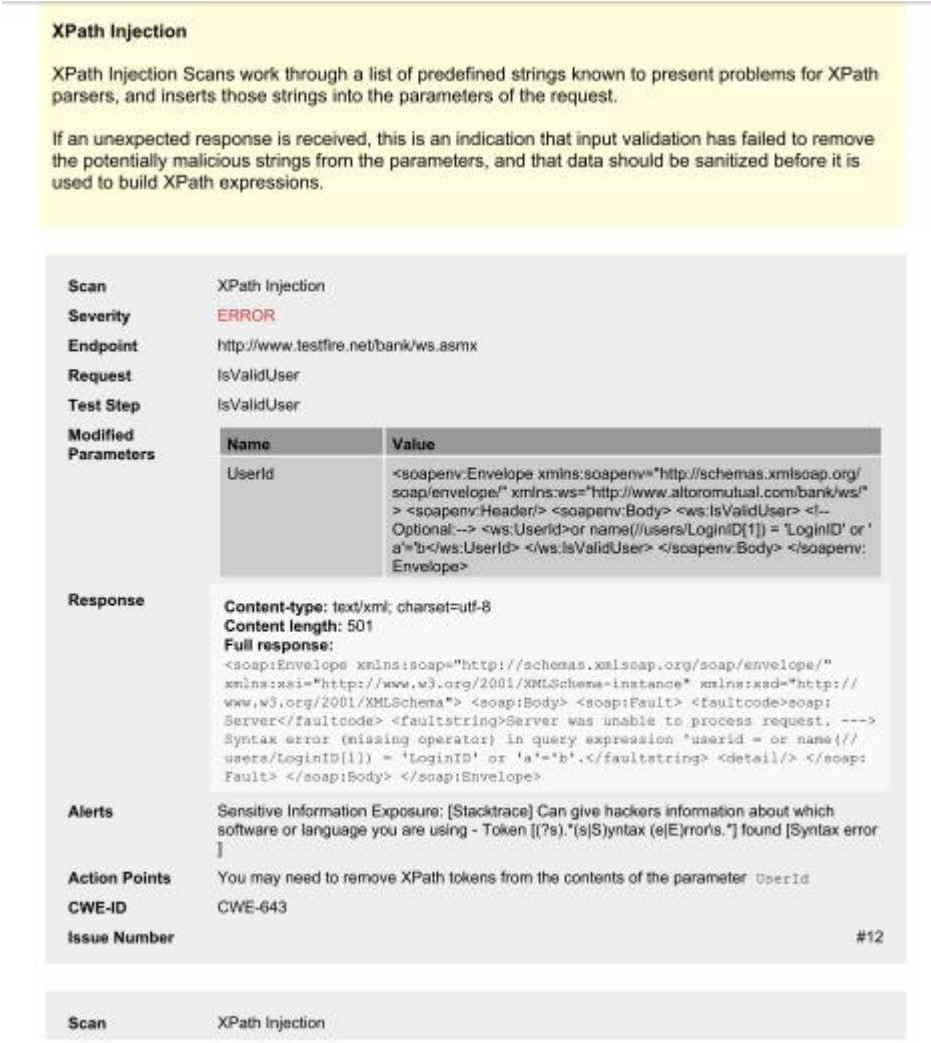


Figura 21. Reporte de Vulnerabilidades

Las vulnerabilidades que pudo detectar para este caso práctico fueron Fuzzing Scan, Xpath Injection, Malformed XML, Cross Site Scripting, SQL Injection e Invalid Types y sumando todas estas vulnerabilidades se obtuvo un total de 62 Bugs de seguridad Encontrados de las 1227 revisiones del sistema.

También la aplicación da a conocer un reporte de seguridad más detallado, el cual explica la razón de la falla de seguridad y las contramedidas a tomar para contrarrestar las vulnerabilidades encontradas.



XPath Injection

XPath Injection Scans work through a list of predefined strings known to present problems for XPath parsers, and inserts those strings into the parameters of the request.

If an unexpected response is received, this is an indication that input validation has failed to remove the potentially malicious strings from the parameters, and that data should be sanitized before it is used to build XPath expressions.

Scan	XPath Injection
Severity	ERROR
Endpoint	http://www.testfire.net/bank/ws.asmx
Request	IsValidUser
Test Step	IsValidUser
Modified Parameters	**Name**: UserId — **Value**: <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://www.altoromutual.com/bank/ws/" > <soapenv:Header/> <soapenv:Body> <ws:IsValidUser> <!--Optional:--> <ws:UserId>or name(//users/LoginID[1]) = 'LoginID' or 'a'='b</ws:UserId> </ws:IsValidUser> </soapenv:Body> </soapenv:Envelope>
Response	**Content-type:** text/xml; charset=utf-8 **Content length:** 501 **Full response:** <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <soap:Body> <soap:Fault> <faultcode>soap:Server</faultcode> <faultstring>Server was unable to process request. ---> Syntax error (missing operator) in query expression 'userid = or name(//users/LoginID[1]) = 'LoginID' or 'a'='b'.</faultstring> <detail/> </soap:Fault> </soap:Body> </soap:Envelope>
Alerts	Sensitive Information Exposure: [Stacktrace] Can give hackers information about which software or language you are using - Token [(?s).*(s|S)yntax (e|E)rror\s.*] found [Syntax error]
Action Points	You may need to remove XPath tokens from the contents of the parameter UserId
CWE-ID	CWE-643
Issue Number	#12

Scan	XPath Injection

Figura 22. Reporte Xpath Injection

7.4.5 WS-Attacker [4]

WS-Attacker es un framework modular open Source que se encuentra desarrollado en java y permite realizar pruebas de instrusión contra servicios web, este framework fue creado en la catedra de seguridad de redes y datos de la Universidad Ruhr de Bochumm (Alemania) y también con la colaboración de la empresa 3curity Gmbh. (Bochumm, s.f.) [4]

WS-Attaker permite cargar archivos WSDL y enviar mensajes SOAP al servicio web que requiera pruebas de instrusión muy parecido a la forma es que se carga los servicios web con la herramienta previamente mencionada SOAPUI. Por otra parte los diferentes tipos de ataques a los web services son realizados gracias a los diferentes plugins y librerías que el framework soporta.

7.4.6 Pruebas de Caja Negra

Ejemplo Práctico

El framework WS-Attacker maneja una interfaz amigable que permite un fácil uso el cual permite cargar el archivo WSDL y para este ejemplo práctico se trabajara con el siguiente archivo WSDL http://www.testfire.net/bank/ws.asmx?WSDL que servirá para realizar todos los posibles pruebas de intrusión que permite realizar la aplicación.

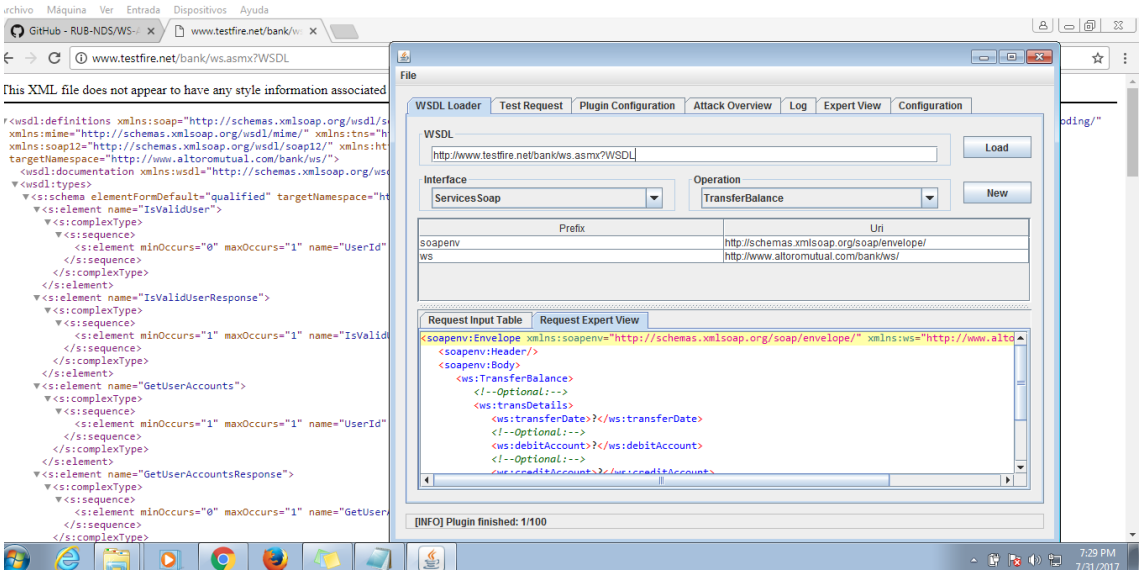


Figura 23. Interfaz WS-Attacker

La aplicación lista todas las operaciones o funciones que tiene disponible archivo WSDL por medio de su cliente SOAP, y al igual que la aplicación SOAP UI permite ingresar valores sobre esas funciones en los request y observar el resultado o response del servicio. Para este ejemplo se selecciona la operación o función “TransferBalance” dado que al analizar el conjunto de operaciones se concluye que es una de las operaciones más vulnerables de manera que si un atacante puede vulnerar esta operación tendría acceso realizar transferencias para beneficio propio.

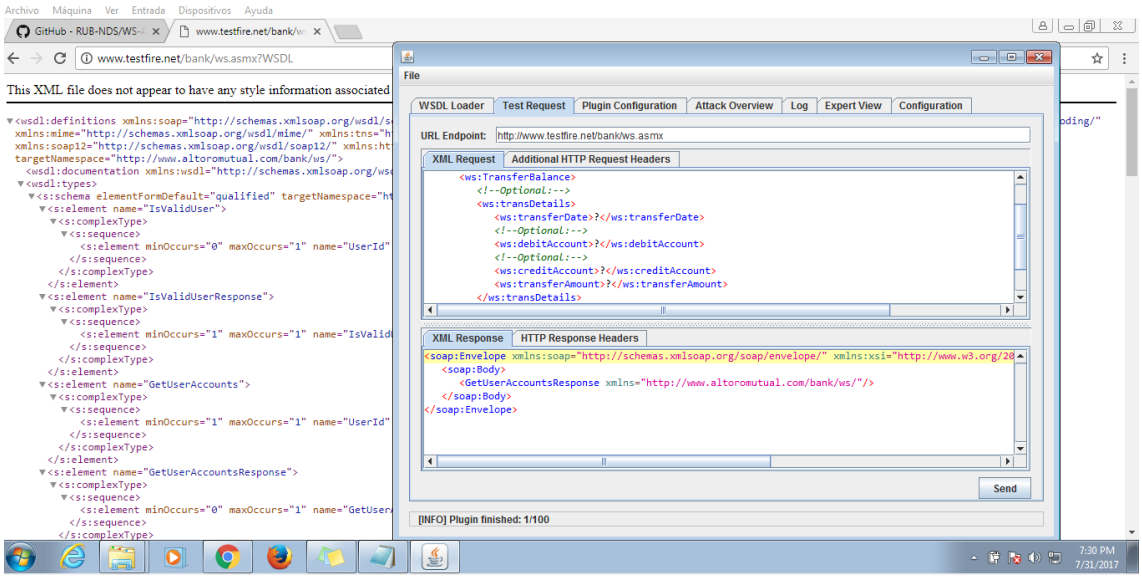


Figura 24. Pestaña Test Request

WS-Attacker permite listarnos una serie de ataques el cual llevaran a cabo todos los casos de pruebas de intrusión que necesitemos realizar al servicio, para este ejemplo práctico se ha seleccionado todos los ataques con el fin de determinar la efectividad de esta aplicación con respecto a la búsqueda de vulnerabilidades.

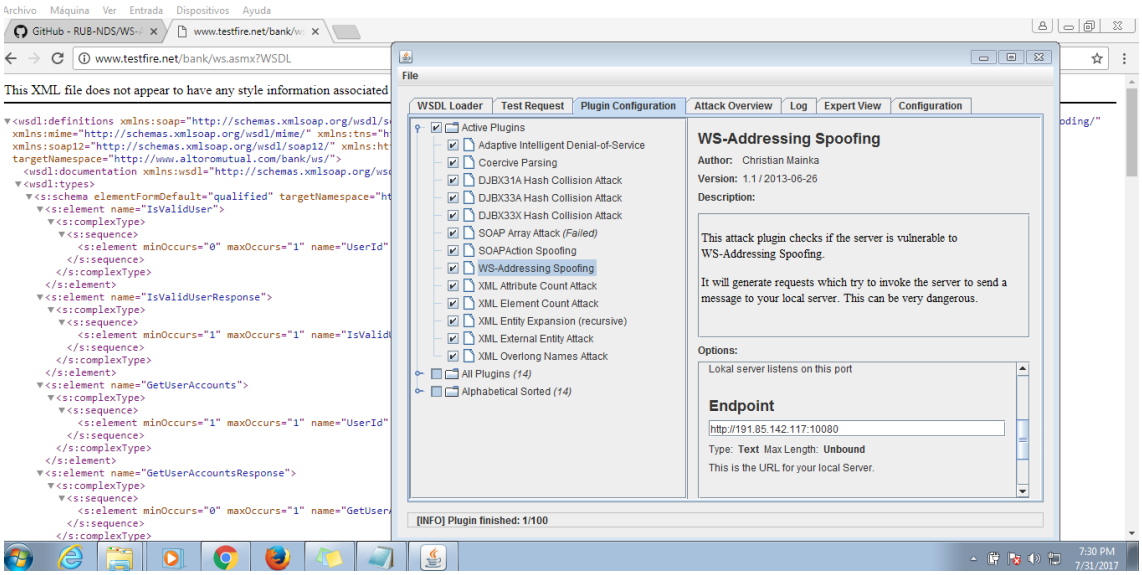


Figura 25. Configuración de Ataques

La herramienta permite la ejecución de cada uno de estos ataques de forma automática, de manera que se observa una lista de todas las pruebas

de intrusión que la aplicación va realizando informando el estado de cada prueba, el nivel de importancia de la vulnerabilidad y la descripción detallada del resultado de la prueba. Para este ejemplo práctico se puede observar el resultado de cada una de las pruebas programadas y la detección de una Crítica vulnerabilidad encontrada en el ataque SOAPAction Spoofing y dando a conocer la descripción detallada del resultado del ataque indicando que el servidor ejecuta la operación especificada por SOAPAction Header y como previamente se ha explicado esta es una vulnerabilidad que puede aprovechar un atacante de modo que pueda modificar el header SOAPAction durante la comunicación de este servicio con otro y de esta forma controlar la comunicación de estos servicios para beneficio propio.

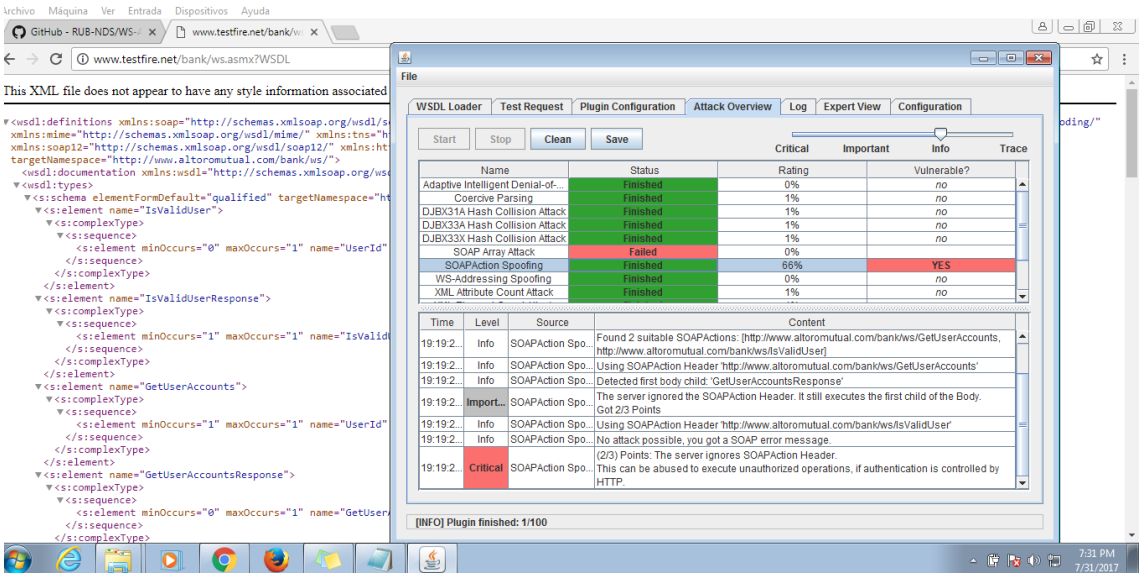


Figura 26. Resultado de los Ataques

7.4.7 IBM AppScan [23]

AppScan es una aplicación comercial desarrollada por IBM Security, actualmente se encuentra entre las herramientas de penetration testing más populares. AppScan permite automatizar las pruebas de intrusión en las aplicaciones web con el fin de encontrar vulnerabilidades. Además de evaluar aplicaciones web, AppScan permite realizar penetration testing a Web Services usando un cliente llamado Generic Services Client (GSC).

Generic Services Client (GSC), permite cargar archivos WSDL y sus respectivas operaciones. Es posible seleccionar por medio de la interfaz gráfica cada una de las operaciones y de acuerdo a la operación seleccionada ingresar los parámetros requeridos con el fin de poder enviar la solicitud al servidor y ver los resultados. Todos estos procesos son registrados por AppScan y posteriormente utilizados para crear casos de prueba basados en el número de solicitudes hechas por el GSC para el servicio.

7.4.8 Pruebas de Caja Negra

Ejemplo Práctico

AppScan tiene una interfaz más completa, además de poder realizar pruebas de intrusión a aplicaciones web, cuenta con un módulo independiente de pruebas para servicios SOAP llamado Generic Service Client (GSC) el cual sirve para cargar el archivo WSDL http://www.testfire.net/bank/ws.asmx?WSDL

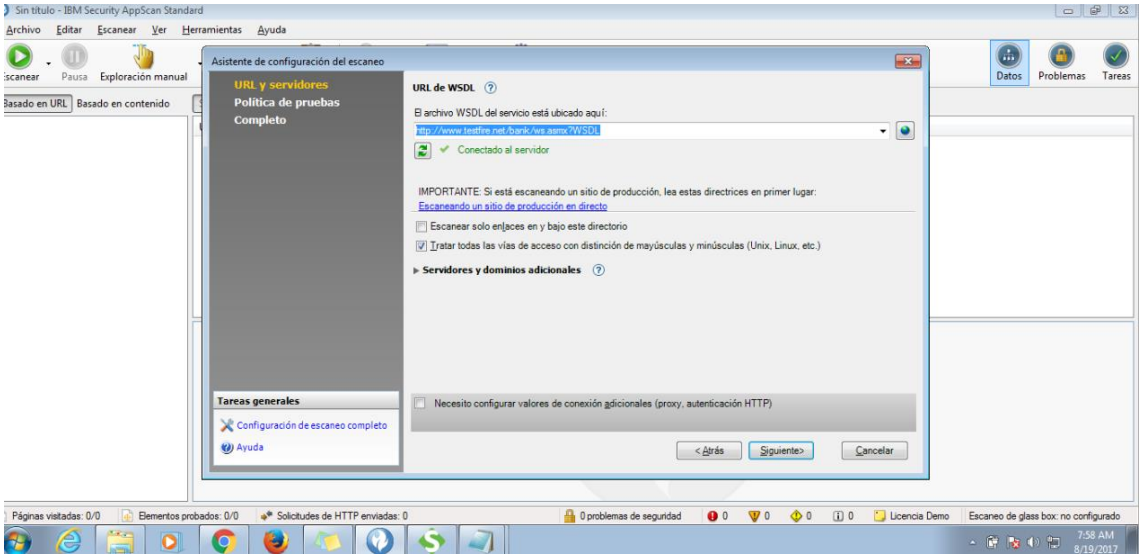


Figura 27. Generic Service Client

La aplicación cuenta con un amplio conjunto de validaciones de seguridad que permiten armar varios casos de pruebas, esto es realizado mediante el componente Políticas de pruebas.

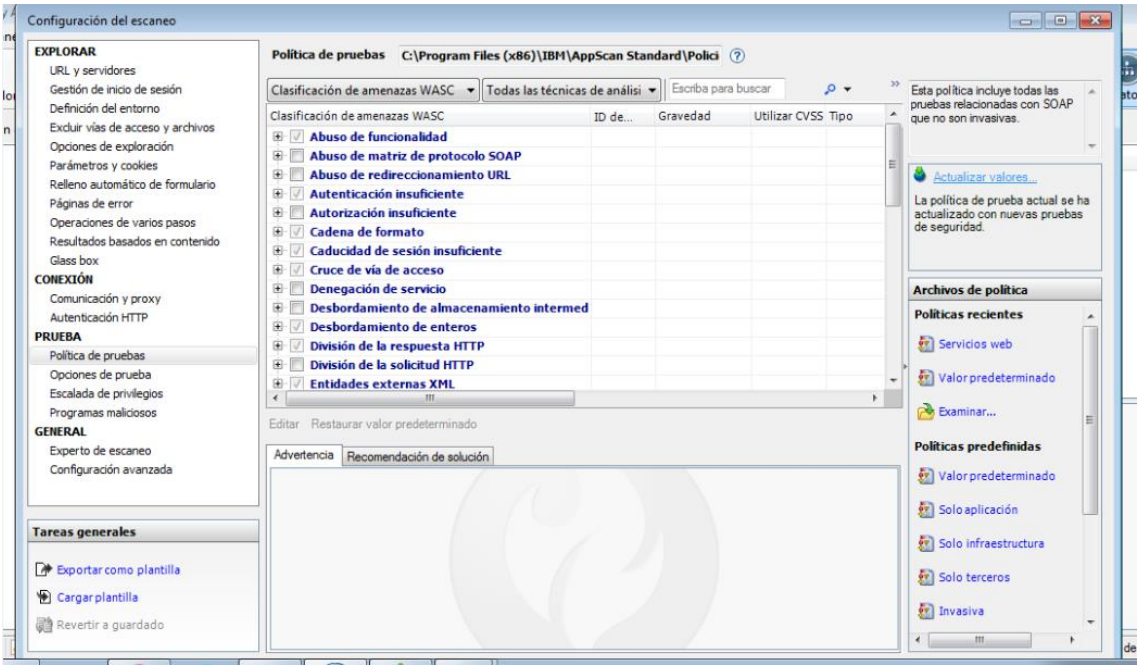


Figura 28. Política de Pruebas

Los casos de pruebas de intrusión que la aplicación permite realizar son:

XML External Entities, Information Leakage, Insufficient Authentication, SQL Injection, Cross Site Scripting, Directory Indexing, Abuse of functionality, Session Fixation, OS Commanding, Format String, Brute Force, Insecure Indexing, LDAP Injection, Content Spoofing, Remote File Inclusion, Null Byte Injection, SSI Injection, Insufficient Session Expiration, Insufficient Transport Layer Protection, HTTP Response Splitting, Path Traversal y XPath Injection.

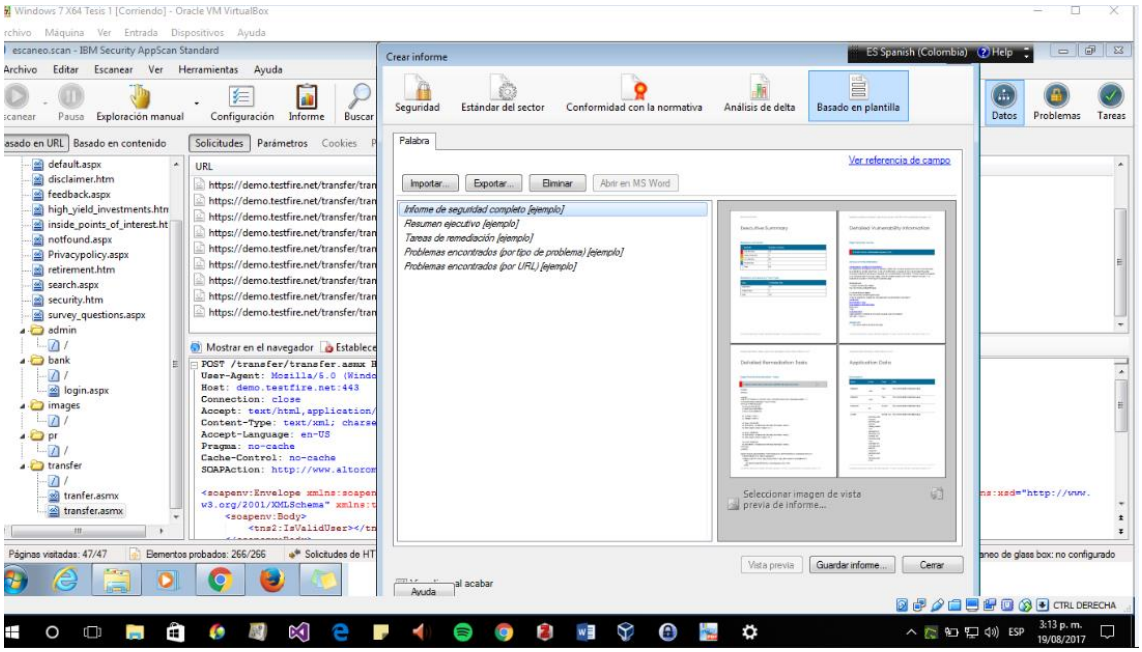
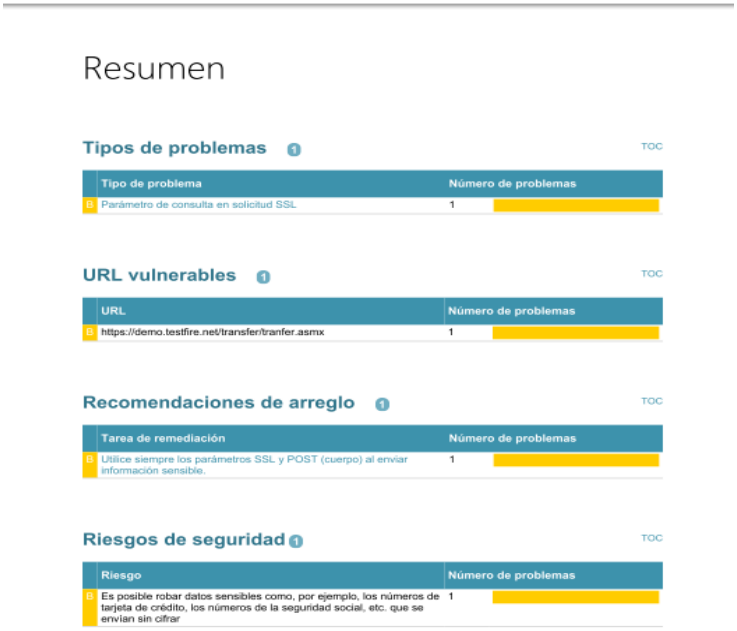


Figura 29. Plantillas Para Reportes de Seguridad

Para este ejemplo práctico no se pudo llevar a cabo la ejecución de todos estos casos sobre el archivo WSDL http://www.testfire.net/bank/ws.asmx?WSDL debido a que la aplicación exige una licencia para su uso completo, sin embargo cabe mencionar aspectos positivos encontrados como por ejemplo esta que la aplicación tiene plantillas basadas en OWASP para crear bien sea manualmente o automáticamente informes de seguridad de los resultados, esto es posible ya que la aplicación una vez terminada la ejecución de los escenarios crea automáticamente un informe completo y detallado con un lenguaje no tan técnico y más de negocio, dando a conocer aspectos como el cantidad de problemas encontrados, Recomendaciones para mitigar la vulnerabilidades encontradas, y una lista de riesgos que implica las vulnerabilidades encontradas, también detalla la causa de la vulnerabilidad.



Resumen

Tipos de problemas 1 TOC

Tipo de problema	Número de problemas
Parámetro de consulta en solicitud SSL	1

URL vulnerables 1 TOC

URL	Número de problemas
https://demo.testfire.net/transfer/tranfer.asmx	1

Recomendaciones de arreglo 1 TOC

Tarea de remediación	Número de problemas
Utilice siempre los parámetros SSL y POST (cuerpo) al enviar información sensible.	1

Riesgos de seguridad 1 TOC

Riesgo	Número de problemas
Es posible robar datos sensibles como, por ejemplo, los números de tarjeta de crédito, los números de la seguridad social, etc. que se envían sin cifrar	1

Figura 30. Resumen De Vulnerabilidades

A diferencia de los reportes que genera SoapUI este es un reporte más completo debido a que tiene una sección con lenguaje no técnico y una especialmente técnica.

7.4.9 HP Webinspect [23]

HP WebInspect es una aplicación de penetration testing muy popular desarrollada por la compañía Hewlet Packard. Esta aplicación se caracteriza por utilizar técnicas y ataques de hacking del mundo real que ayudan a realizar un análisis de seguridad más profundo sobre aplicaciones web y web services para identificar vulnerabilidades de seguridad.

WebInspect utiliza el componente Web Services Test Designer para cargar los archivos WSDL y sus respectivos métodos. Entre las principales ventajas que tiene la herramienta es que en el momento de cargar los métodos rellena automáticamente el valor de cada uno de los tipos de datos de los parámetros requeridos esto la diferencia de otras herramientas que necesitas

llenar manualmente los parámetros por cada método. (Panda, Web Services Penetration Testing Part 3: Automation with AppScan and Webinspect, 2013)

7.4.10 Pruebas de Caja Negra

Ejemplo Práctico

Dentro de la función Start a Web Service Scan se encuentra el modulo Web Services Test Designer permite cargar el archivo WSDL http://www.testfire.net/bank/ws.asmx?WSDL por medio de este módulo se cargan todos los métodos de este archivo, como se mencionó anteriormente una de las principales características de la herramienta es rellenar automáticamente los tipos de datos de los parámetros requeridos, pero cabe mencionar que la herramienta tiene una desventaja y es cuando la herramienta no puede rellenar estos parámetros de manera automática señala con una X en rojo aquellos métodos donde no pudo rellenar sus parámetros requeridos, para esta práctica la herramienta no pudo encontrar el parámetro para el método “IsValidUser”.

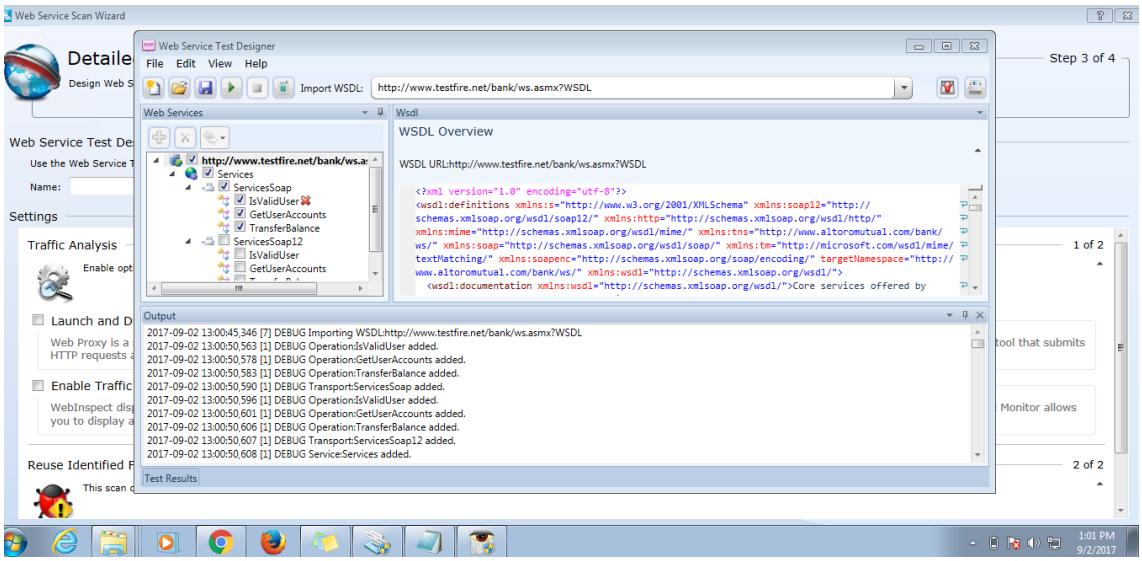


Figura 31. Web Services Test Designer

La herramienta tiene el componente Policy Manager en el cual se pueden seleccionar los test cases de seguridad que se necesiten.

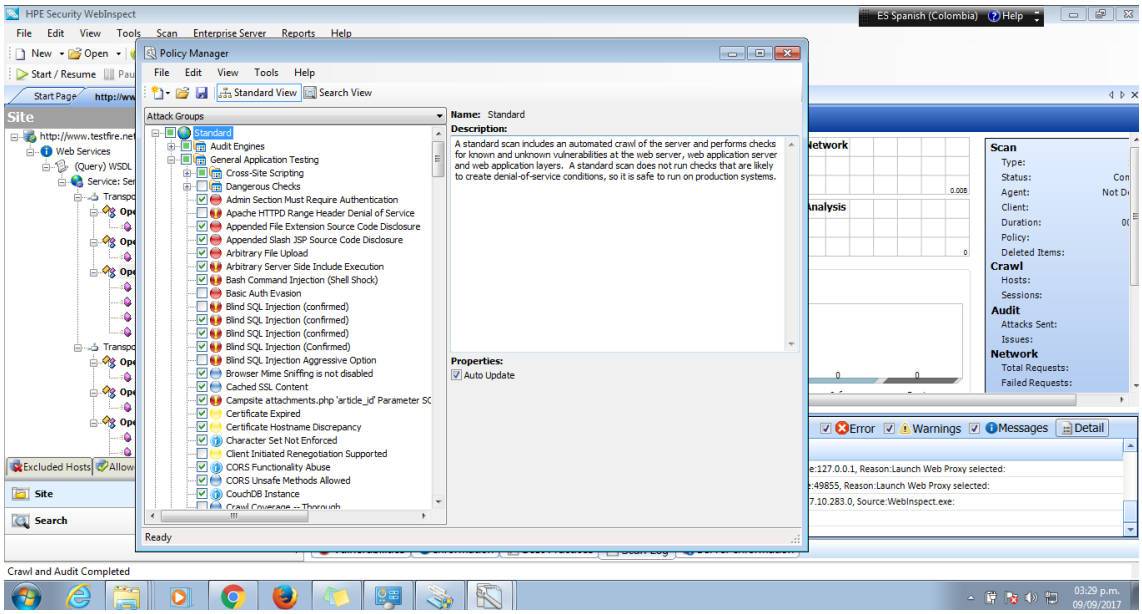


Figura 32. Policy Manager

Por cuestiones de licencia estos ataques no se pudieron realizar en el web services de prueba debido a que solamente se encuentra habilitado el modulo para escaneo de aplicaciones web en modo demo, sin embargo cabe mencionar aspectos positivos de la herramienta encontrados entre ellos está que posee un Scan Dashboard el cual muestra por medio de diagramas de barras estadísticos la cantidad de vulnerabilidades clasificadas según su severidad.

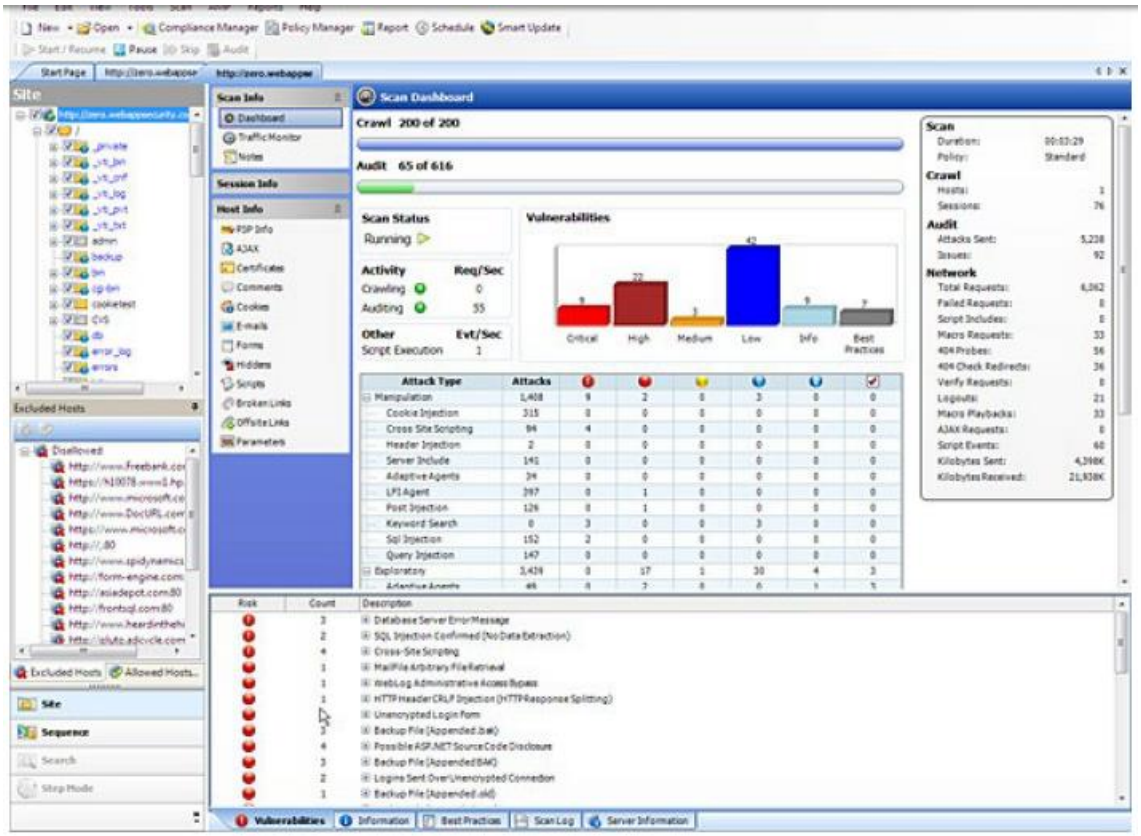


Figura 33. Scan Dashboard

Las vulnerabilidades que se consideren como falsos positivos se pueden clasificar y remover de la lista de vulnerabilidades detectadas, igualmente permite la generación de reportes y otra de las buenas características es una api que permite la integración con herramientas de pentestesting manual como Burp Suite.

7.4.11 OWASP ZAP Zed Attack Proxy [18]

Es una herramienta desarrollada por OWASP, y es caracterizada por ser una de las herramientas de seguridad open source más populares del mundo y diariamente es activamente recibe mantenimiento por cientos de usuarios voluntarios internacionales. Permite encontrar automáticamente vulnerabilidades de seguridad en aplicaciones web y es considerada como

una herramienta que deja una buena experiencia en el momento de realizar pruebas de seguridad manual.

Owasp Zap dentro de sus características no posee un componente propio para cargar archivos WSDL, en este caso actúa como proxy para escuchar las peticiones enviadas por otra herramienta. (OWASP, 2017)[18].

7.4.12 Pruebas de Caja Negra

Ejemplo Práctico

SoapUI juega un papel muy importante en horas de realizar pentesting en web services con Owasp Zap, para este ejemplo práctico se realizara con base al web services de pruebas http://www.testfire.net/bank/ws.asmx y se selecciona el método “TransferBalance”.

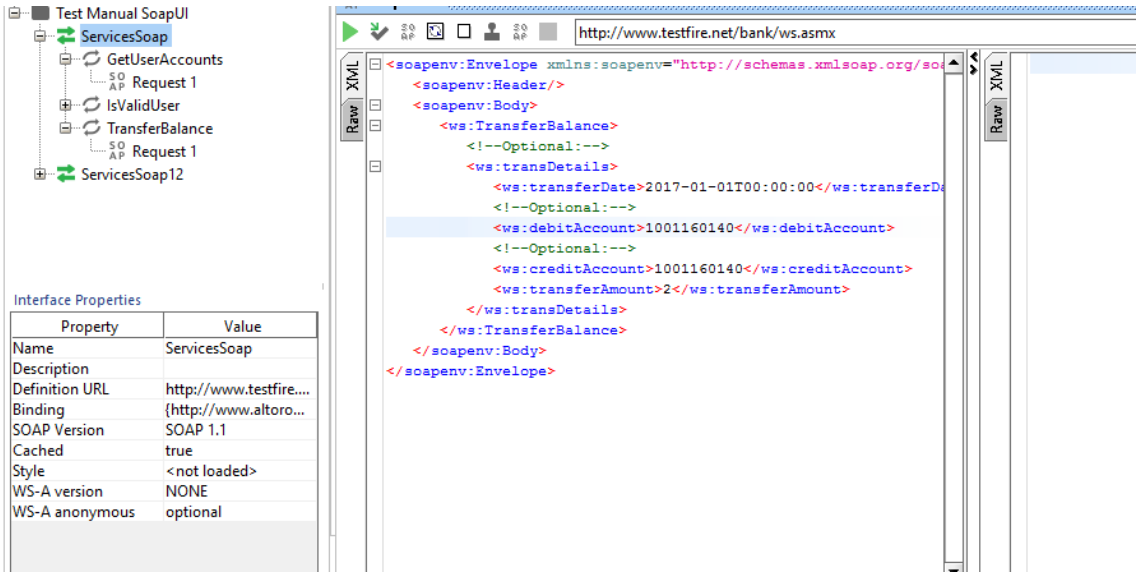


Figura 34. Método Soap TransferBalance

El proxy de ambas herramientas se ha configurado con el fin de que Owasp Zap reciba las respuestas del método “TransferBalance”.

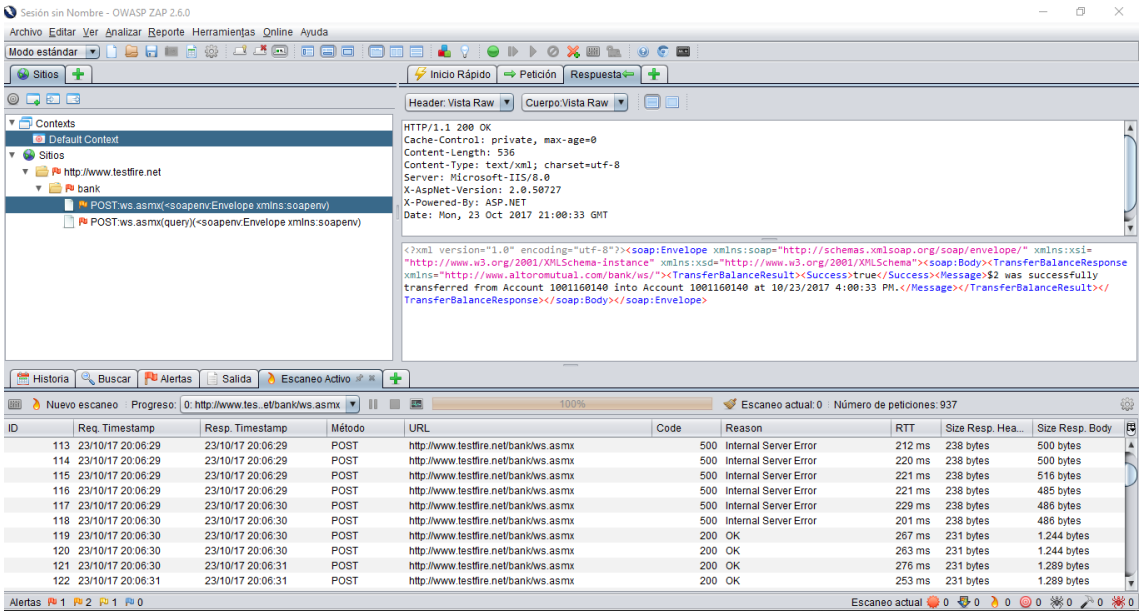


Figura 35. Integración SoapUI con Owasp Zap

Una vez que se obtiene la respuesta del web service, Owasp Zap permite armar un escaneo automático, donde es posible incluir todos los test cases de seguridad necesarios para descubrir las vulnerabilidades del método por medio del componente política, adicionalmente es posible ingresar vectores de entrada y vectores personalizados sobre los parámetros de entrada del método.

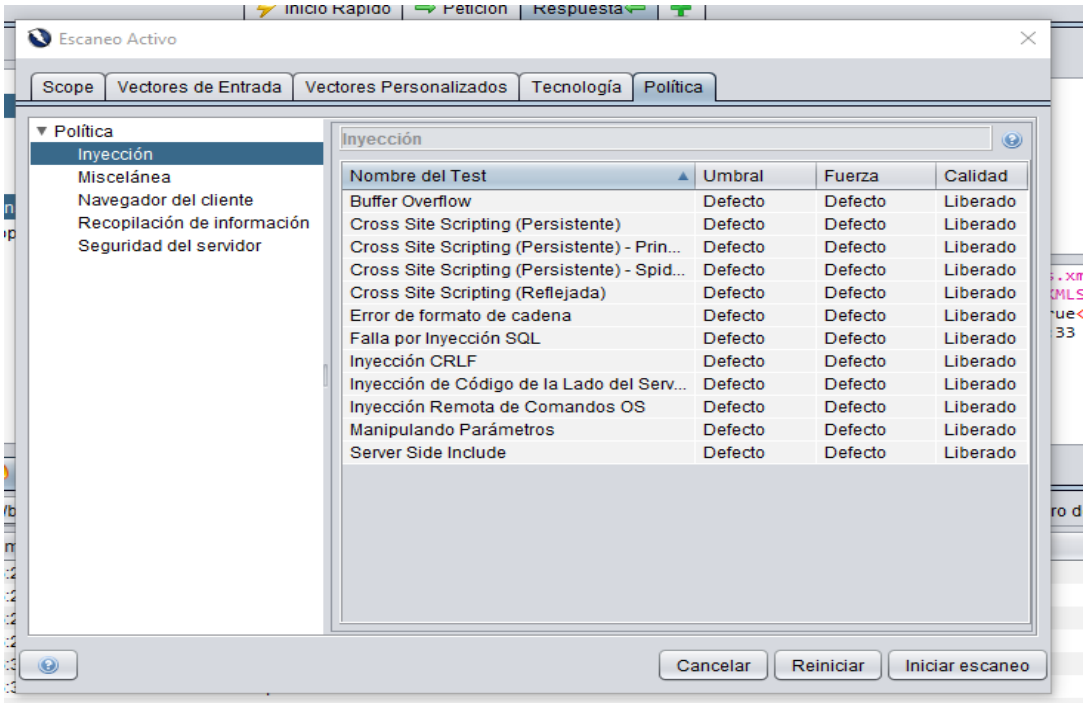


Figura 36. Lista de Pruebas de Seguridad

Al iniciar el escaneo se obtiene como resultado las siguientes alertas con su respectiva descripción, nivel de riesgo, una evidencia y las medidas preventivas solución:

Falla por Inyección SQL

Nivel de Riesgo → Alto

Solución: “No confíe en los valores de entrada del lado del cliente, incluso si en el lado del cliente se realice una validación.

En general, comprobar todos los datos de entrado en el servidor.

Si la aplicación usa JDBC, usar PreparedStatement o CallableStatement, con parámetros pasados por '?' Si la aplicación utiliza ASP, usar ADO Command Objects con una fuerte comprobación de tipos de consultas y parámetros.

Si la Base de Datos puede usar Stored Procedures (Procedimientos Almacenados), úselos.¡NO concatenar cadenas en los query (consultas) en el procedimientos almacenados, o utilizar 'exec', 'exec immediate', o su funcionalidad equivalente! No crear consultas SQL dinámicas usando una sencilla concatenación de cadenas.

Aplique aun lista blanca (whitelist) de caracteres permitidos, o una lista negra (blacklist) de caracteres no permitidos en la entrada (input) del usuario.

Aplique el privilegio mínimo posible al usuario de la base de datos de los privilegios usados.

En particular evitar el uso de los usuarios de base de datos 'sa' o 'db-owner'. Esto no elimina la inyección SQL, pero minimiza su impacto”.Conceder el mínimo acceso de base de datos que es necesario para la aplicación.

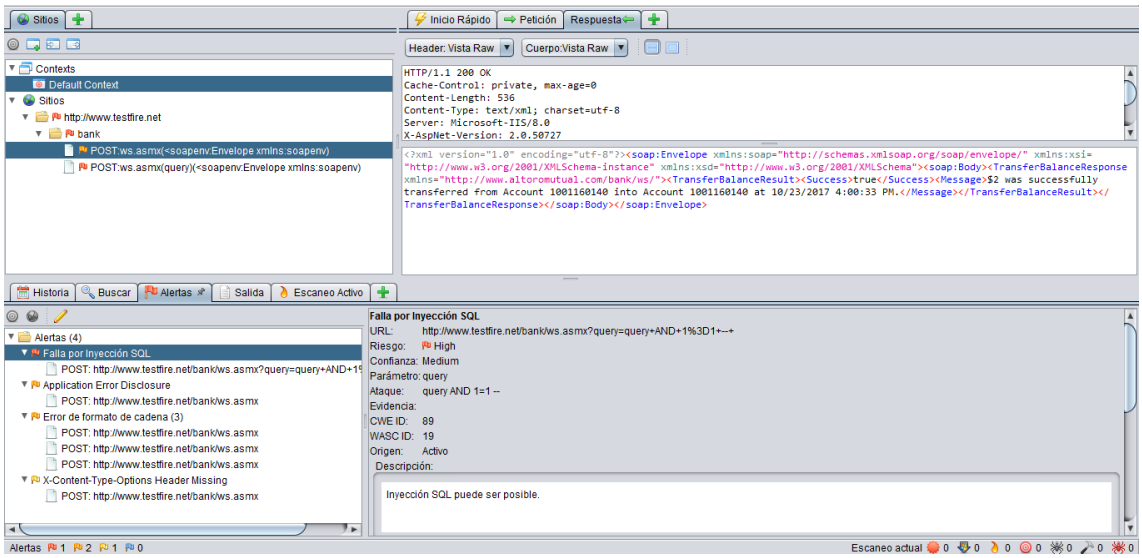


Figura 37. Resumen de Vulnerabilidad por Inyección SQL

Vulnerabilidad: Application Error Disclosure

Nivel de Riesgo → Medio

Solución: “Revisar el código fuente de esta página. Implementar páginas de error personalizadas. Considerar un mecanismo para proporcionar una referencia / identificador de error único al cliente (navegador) al registrar los detalles en el lado del servidor y no exponerlos al usuario”.

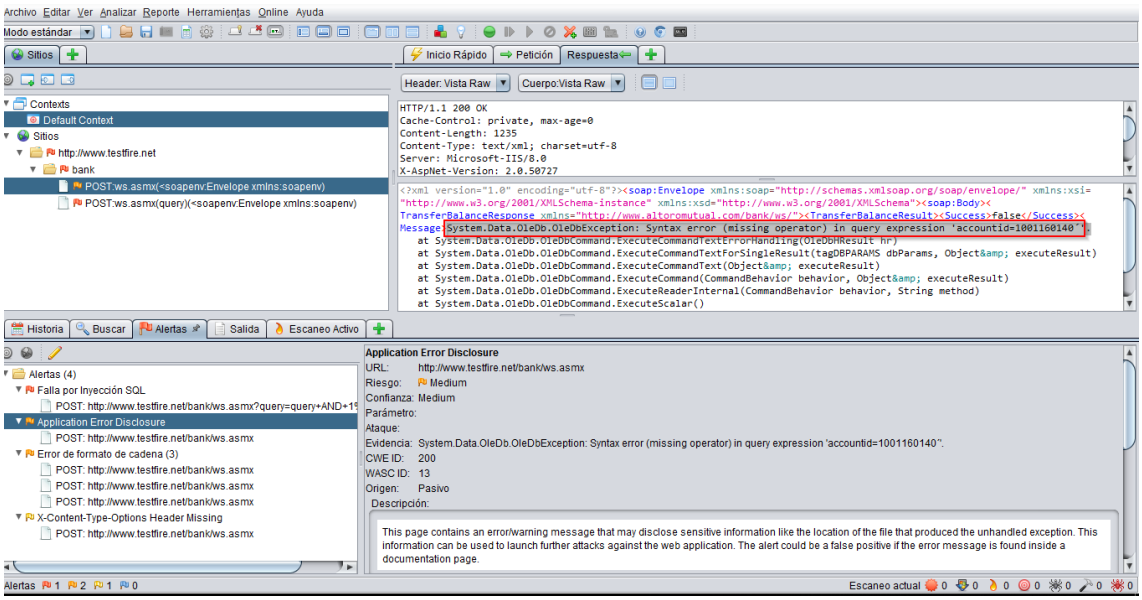


Figura 38. Resumen Error Application Error Disclosure

Vulnerabilidad: Error de Formato de Cadena

Nivel de Riesgo→ Medio

Solución: “Vuelvo a escribir el programa en segundo plano con la eliminación adecuada de las cadenas de caracteres incorrectos. Esto requerirá una recompilación del fondo ejecutable”.

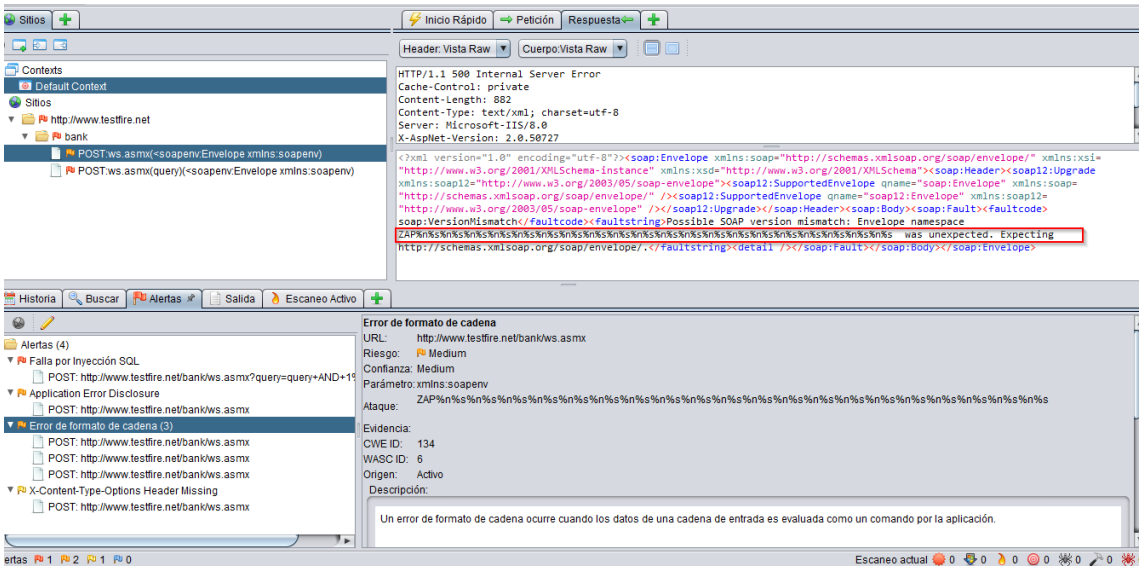


Figura 39. Resumen Error Formato de Cadena

Vulnerabilidad: X-Content-Type-Options Header Missing

Nivel de Riesgo Medio Bajo

Solución: “Asegúrar de que la aplicación / servidor web configure el encabezado de tipo de contenido de manera adecuada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web.

Si es posible, asegúrese de que el usuario final utilice un navegador web moderno y compatible con los estándares que no realice ningún tipo de detección MIME, o que la aplicación web / servidor web pueda indicarle que no realice el rastreo MIME”.

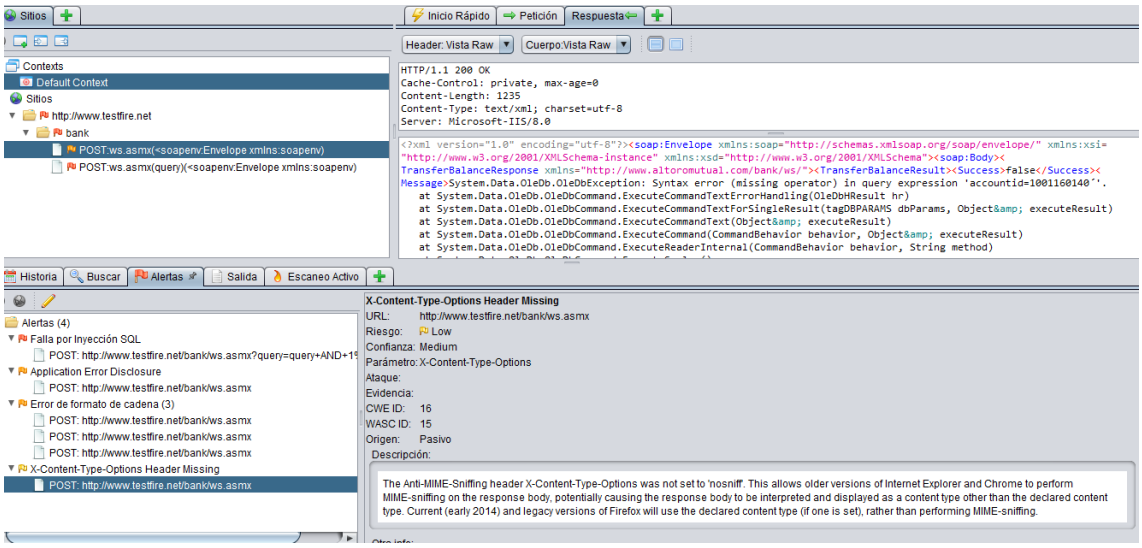


Figura 40. Resumen X-Content-Type Header Missing

Por ultimo Owasp Zap permite la generación de reportes.

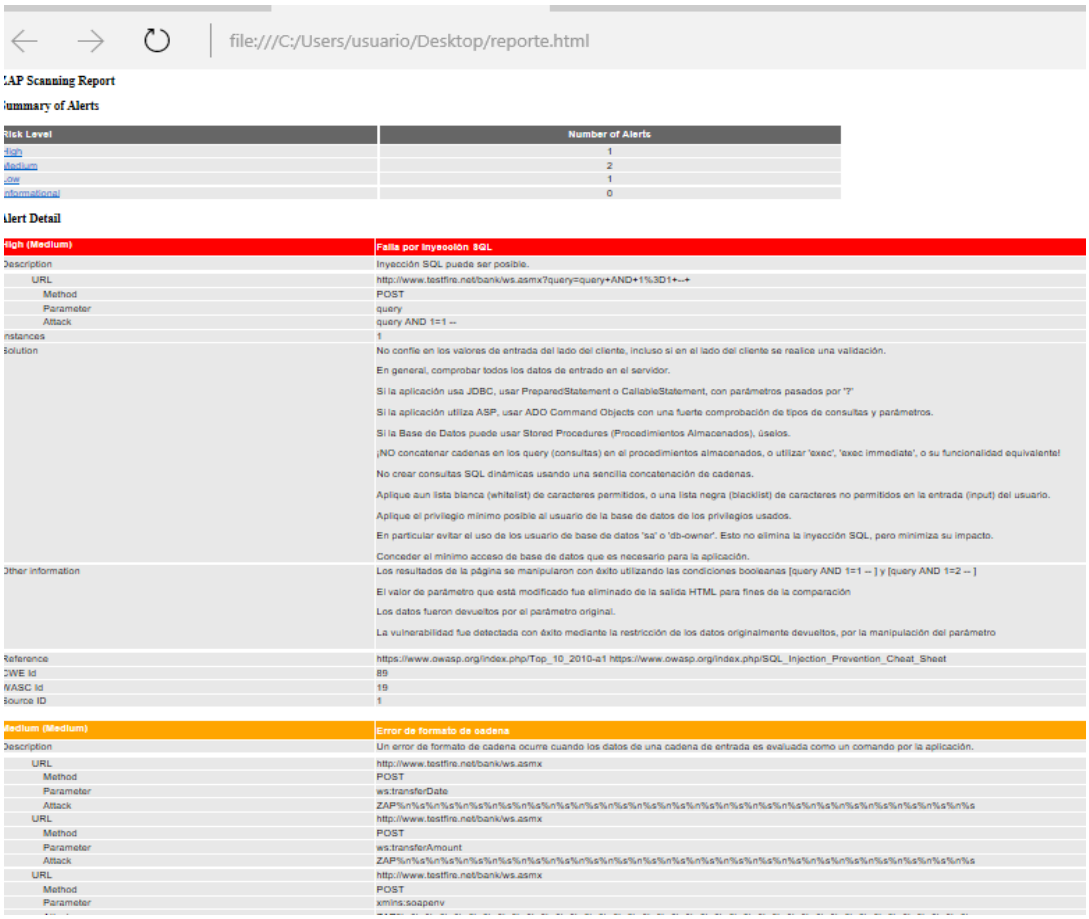


Figura 41. Owasp Zap Scanning Report

7.4.13 Testing de Seguridad Manual en Web Services

Las herramientas que automatizan las pruebas de penetration en web services no son suficientes para un efectivo análisis, por ende es necesario el uso de herramientas que permitan realizar un testing manual esto se debe a que al aplicar testing manual se pueden llegar a cubrir diferentes tipos de casos de pruebas no tradicionales que ayudan a un pentester a entender la funcionalidad y encontrar nuevos enfoques de pruebas que ayudan a que pueda pensar e innovar de manera que pueda ser libre de probar las diferentes vulnerabilidades de la lógica del negocio que literalmente son imposibles de probar por un escáner automático.

Para esta fase Owasp Cheat Sheets recomienda las siguientes herramientas con el fin de garantizar un análisis efectivo en horas de realizar un penetration testing de forma manual en web services, para ello se crea una serie de ejemplo prácticos basado en estas herramientas.

Ejemplos Prácticos

A continuación se realizaran ejemplos prácticos de la herramienta para las dos fases de pruebas manuales:

7.4.14 Test Manual SoapUI Free

En la fase del testing de seguridad manual una de las herramientas que recomienda Owasp Cheet Sheat es la versión open source SoapUI esto se debe a que esta versión es tan potente como su versión comercial a diferencia de que la versión comercial es usada para fase test de seguridad automatizado.

Una de las principales desventajas de esta herramienta en el caso de pruebas de caja negra, es el no mostrar los tipos de datos requeridos en horas de probar una solicitud en particular, para ello se deberá recurrir al fuzz o adivinación de parámetros de entrada o el uso de herramientas que permitan realizar este trabajo como es el caso de la herramieta SOA Client y por ultimo leer el archivo WSDL y entender los requisitos de la solicitud [26].

7.4.15 Fuzz de Parámetros de Entrada

En este primer análisis es necesario adivinar los valores que son necesarios para la comprensión de la solicitud, para este caso práctico se ha seleccionado la operación “TransferBalance” del archivo WSDL ubicado en la siguiente URL http://www.testfire.net/bank/ws.asmx?WSDL, para ello se ha creado dentro de la herramienta SoapUI (Open Source) un proyecto para realizar este tipo de pruebas manuales.

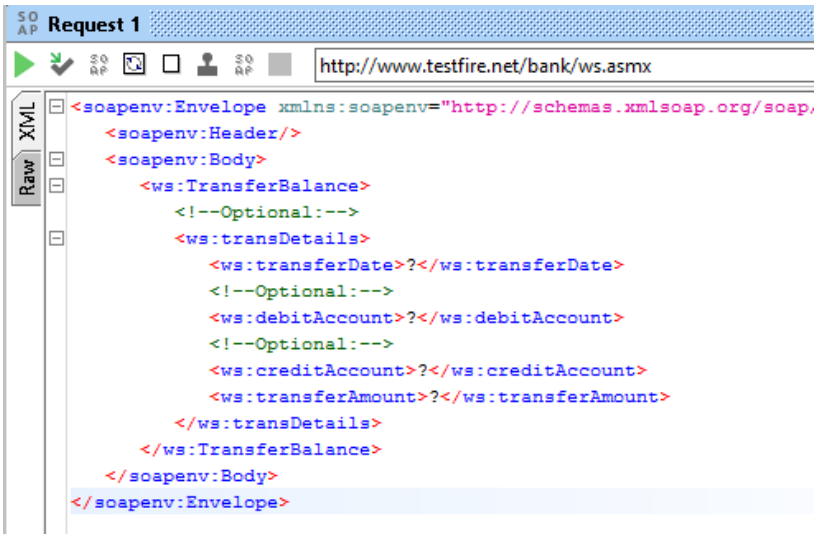


Figura 42. Operación TransferBalance

Dentro de esta operación se pueden apreciar cuatro parámetros del cual no se tiene información del tipo de dato que tiene cada uno, sin embargo

analizando cada uno de estos parámetros puede inferirse el tipo de dato que podría llegar a ser.

El parámetro de entrada transferDate, analizándose puede llegar a ser una tipo de dato fecha, debitAccount puede ser un tipo Integer al igual que el parámetro creditAccount y transferAmount se pude decir que es un tipo de dato double. Teniendo este análisis hecho es posible armar los datos de ingreso para solicitud.

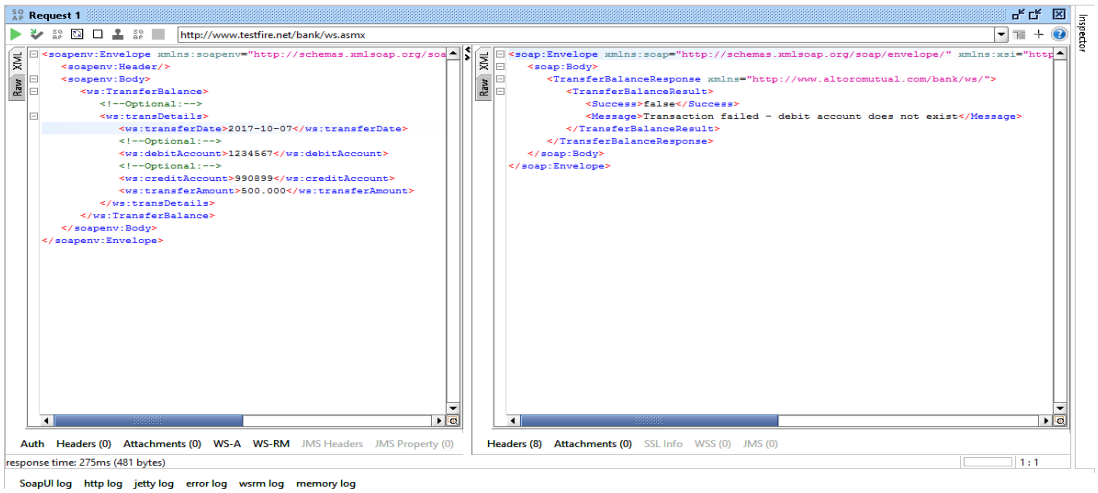


Figura 43. Transaction Failed

Al ejecutar la solicitud se puede apreciar que en el response aparece el Mensaje “Transaction failed - debit account does not exist” pero esta no indica realmente el si el response fallo o dio correcto, para ello SoapUI tiene la opción Raw que permite consultar el estado de la respuesta de la solicitud.

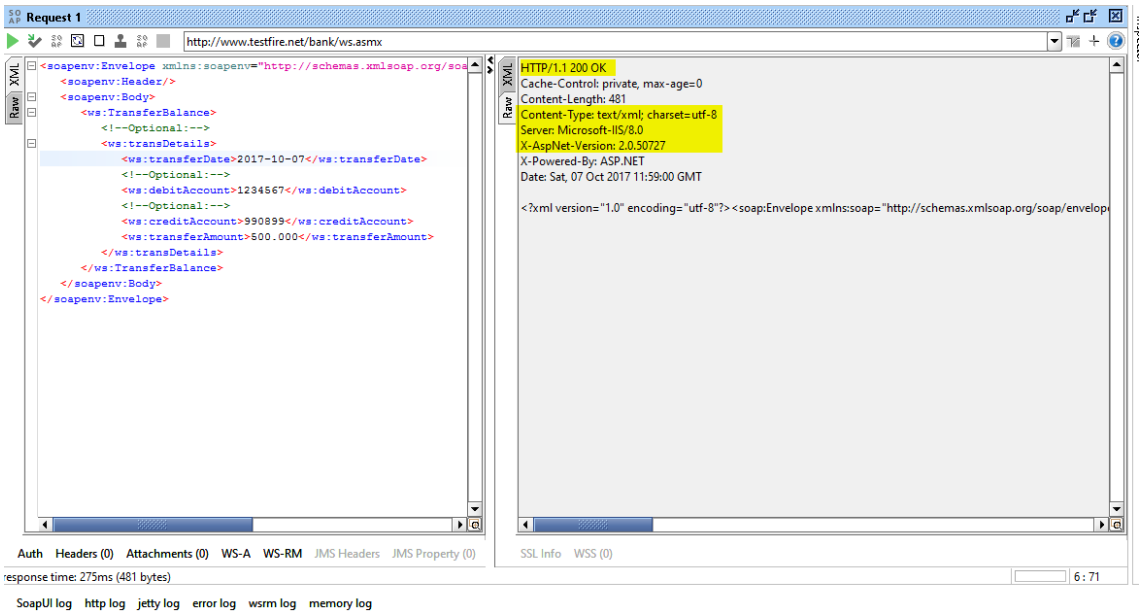


Figura 44. Response Headers

Como bien se sabe los mensajes Soap se transportan por medio del protocolo HTTP, en esta imagen es posible apreciar el status code 200 de la respuesta esto indica que la petición que realiza el servicio al servidor es correcta y además SoapUI entrega información del Soap Response Header como por ejemplo el tipo de formato que acepta el servidor, brinda información sobre el servidor de aplicaciones en el cual está montado el servicio y el tipo de lenguaje de programación en que está construido el web services.

Hasta el momento soapUI no da conocer detalles de los tipos de datos que se ingresaron previamente son correctos y para esto es opcional el uso de la herramienta SOA Client.

SOA Client es un plugin de Mozilla Firefox, la funcionalidad principal del plugin es servir como cliente portable para el acceder a web services y registros UDDI [24].

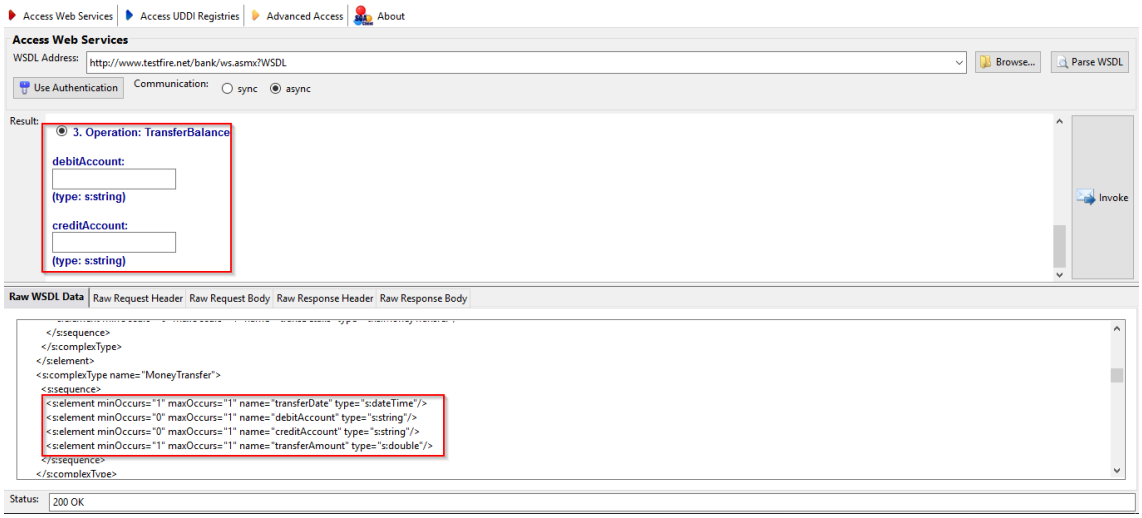


Figura 45. : SOA Client Plugin

Haciendo uso de esta plugin se obtiene el tipo de cada uno de los parámetros de la operación “transferBalance”, sin embargo SOA Client tiene una desventaja en el momento de parsear los parámetros por operación no puede parsearlos todos lo que significa que para este ejemplo de cuatro parámetros solo pudo parsear dos con respectivo tipo de dato.

Estos tipos de datos es posible verlos directamente ingresando al archivo WSDL a partir de cualquier browser sin necesidad del uso del plugin SOA Client todo depende de la comodidad del pentester.

Ahora si al conocer los tipos de datos de cada parámetro nuevamente se crea una solicitud en el servicio.

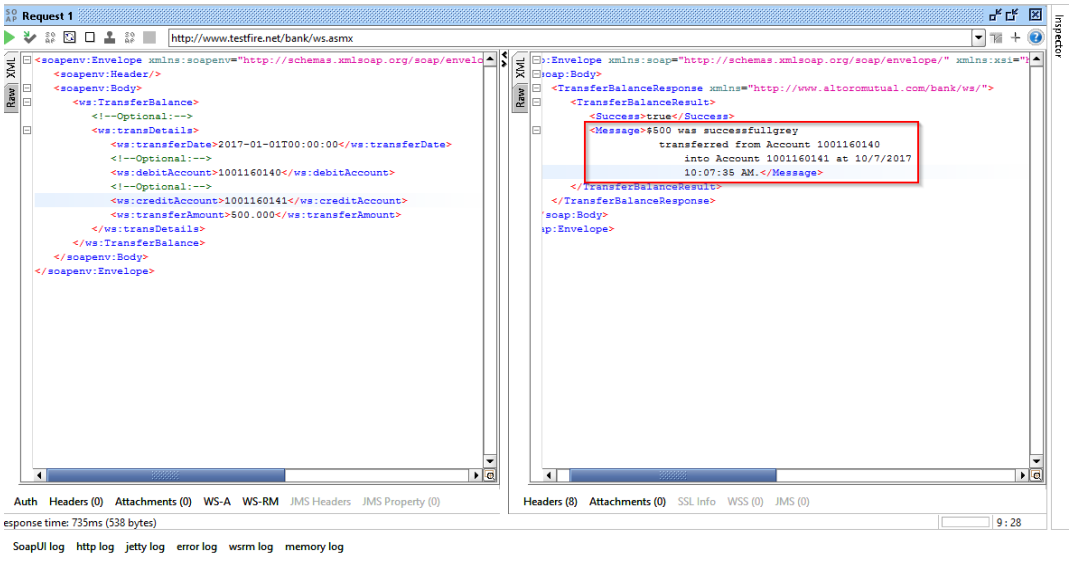


Figura 46. Transferencia Exitosa

Como se puede ver la solicitud se envió de forma satisfactoria transfiriendo el dinero a la cuenta que se ha parametrizado, de esta manera es posible usar estos parámetros para la creación de otros casos de pruebas.

7.4.16 Test Case I

Comprobar si la operación “TransferBalance” permite realizar una transferencia con un saldo negativo.

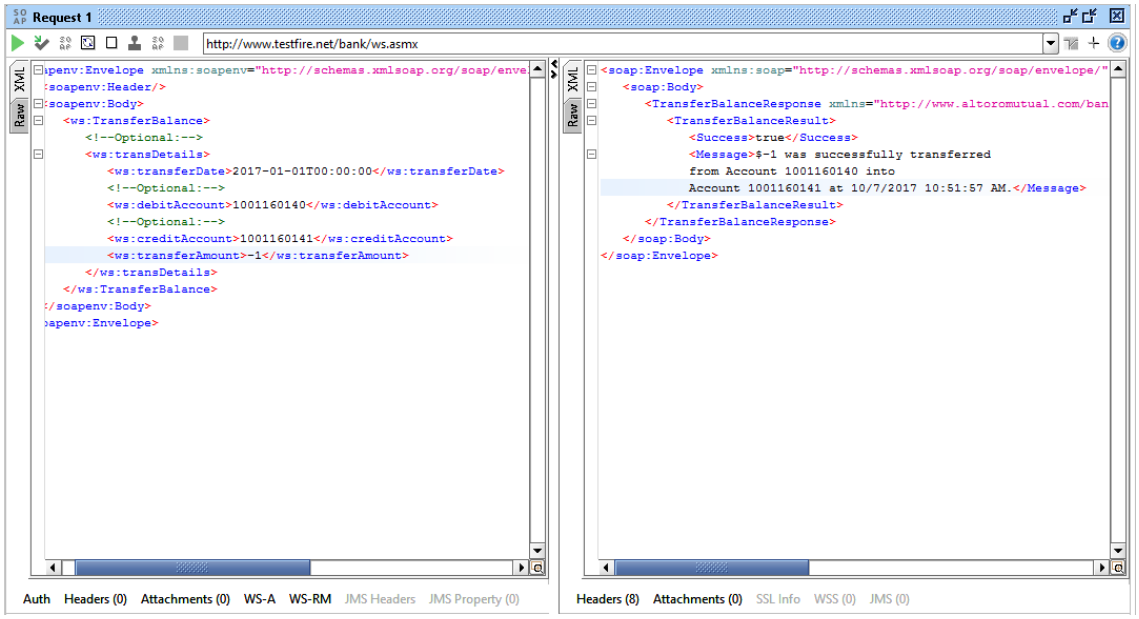


Figura 47. Transferencia Exitosa con Saldo Negativo

Para esta prueba la transferencia se pudo realizar, esto implica un vulnerabilidad critica en la lógica del negocio de la aplicación bancaria. Este tipo de pruebas son más complicadas para una herramientas de prueba automatizadas por consiguiente demuestra la importancia de las pruebas manuales y un complemento eficaz de las pruebas automáticas.

7.4.17 Test Case II

Comprobar si la operación TransferBalance permite realizar transferencia sobre la misma cuenta que debita.

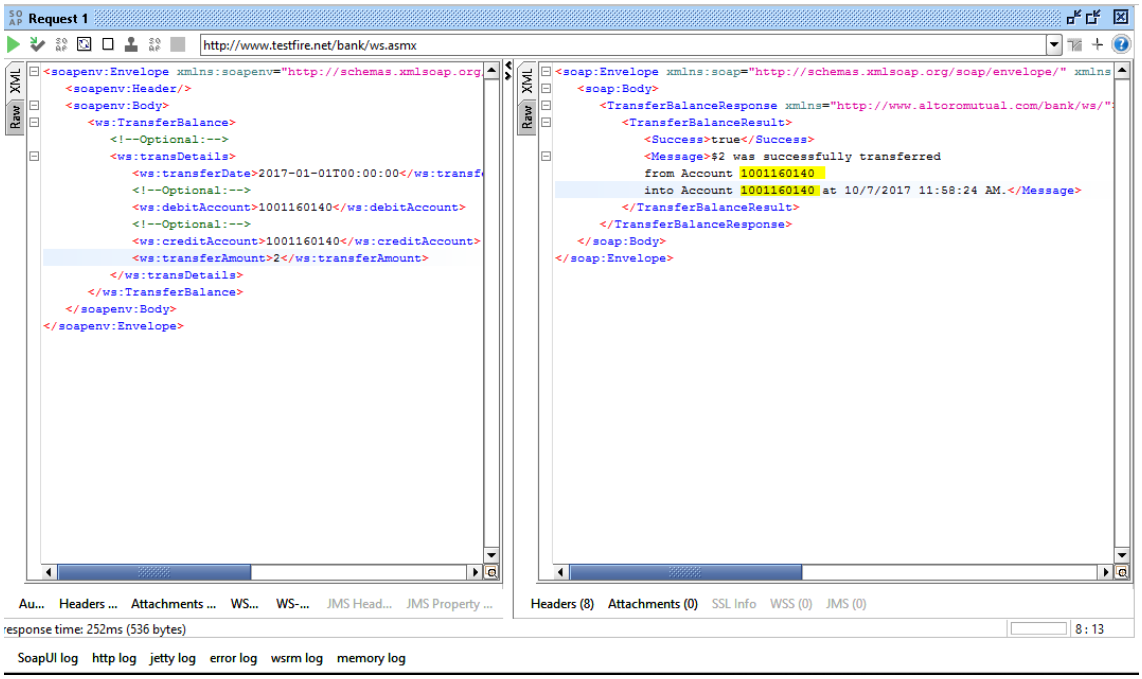


Figura 48.Transferencia con la Misma Cuenta que Debita

Al realizar esta prueba se puede apreciar como el sistema permite realizar transferencias cuando la cuenta que envía es la misma que la cuenta que recibe, otro error critico en la lógica de negocio del web services del aplicativo bancario.

7.4.18 Test Case III

Comprobar la validación de caracteres especiales dentro de los parámetros ingresados.

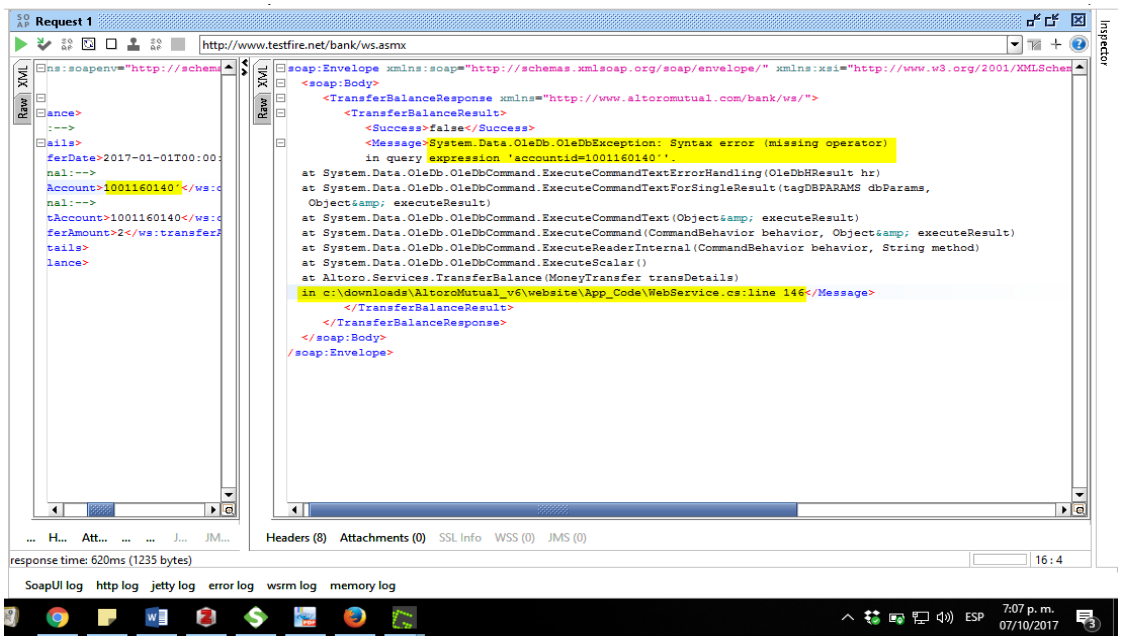


Figura 49. Ingreso de Caracteres Especiales

El realizar esta prueba el response expone información interna, en este caso tenemos un error que da a conocer el patrón de errores de la base de datos y la ruta exacta donde se encuentra alojado el código fuente del web services, información vital que puede ser útil para un atacante para poder acceder a información sensible del banco.

7.4.19 Test Manual Burp Suite [27]

Burp Suite es una herramienta muy utilizada en horas de realizar penetration testing manual en web services, esto se debe a que la herramienta actúa como complemento con otras herramientas.

SoapUI como se ha detallado previamente es una excelente herramienta para la fase de penetration testing manual en web services no obstante la herramienta no tiene la función de poder realizar un fuzz o adivinación de parámetros dado el caso que se quiera realizar un ataque de fuerza bruta al método de login de un Web Service, solo será posible si se integra con Burp Suite esto se debe a que esta herramienta provee una

interfaz que permite el fuzz de parámetros de las peticiones realizadas por SoapUI.

Burp suite es una plataforma integrada que permite realizar penetration testing sobre aplicaciones web y en la en la mayoría de los casos la herramienta permite usar las mismas pruebas para pruebas sobre web services y aplicaciones mobiles.

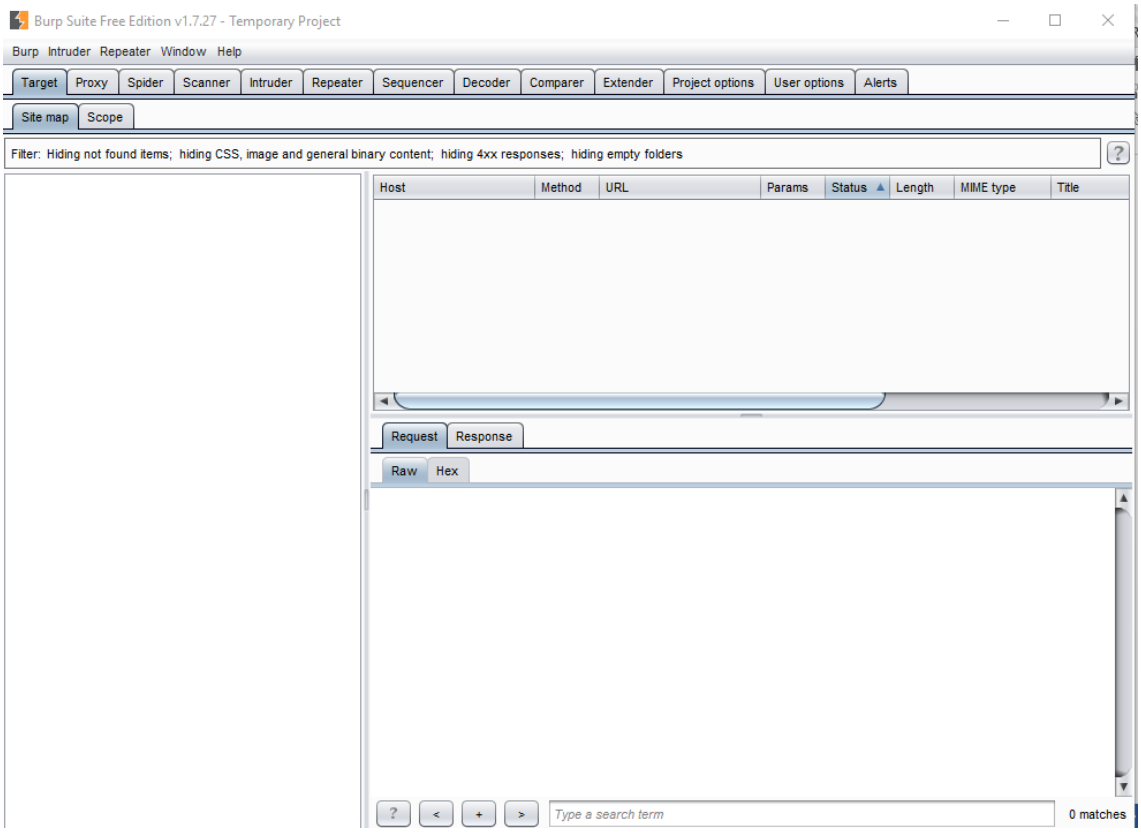


Figura 50. Interfaz Gráfica Burp Suite

Los componentes funcionales principales de Burp Suite son:

Proxy: se ejecuta en el puerto 8080 de manera predeterminada. Intercepta la solicitud y le permite inspeccionar y modificar el tráfico entre su navegador y la aplicación de destino.

Spider: se utiliza para rastrear contenido y funcionalidad mediante el envío automático de valores de formulario.

Scanner: se usa para automatizar la detección de numerosos tipos de vulnerabilidades. El tipo de escaneo puede ser pasivo, activo o dirigido por el usuario.

Intruder: se puede usar para varios propósitos, como realizar ataques personalizados, explotar vulnerabilidades, simular diferentes parámetros, etc.

El repetidor se usa para manipular y reenviar solicitudes individuales y para analizar las respuestas en todos los casos diferentes.

Sequencer: se usa principalmente para verificar la aleatoriedad de los tokens de sesión.

Decoder: se puede usar para decodificar y codificar diferentes valores de los parámetros.

Comparer: se usa para realizar una comparación entre dos solicitudes, respuestas o cualquier otra forma de datos.

Extender: permite escribir fácilmente sus propios complementos para realizar tareas complejas y altamente personalizadas dentro de Burp. O también puede incluir diferentes tipos de extensiones de Burp creadas por diferentes desarrolladores o profesionales de seguridad. Por medio del componente proxy de Burp Suite es posible realizar la integración con SoapUI.

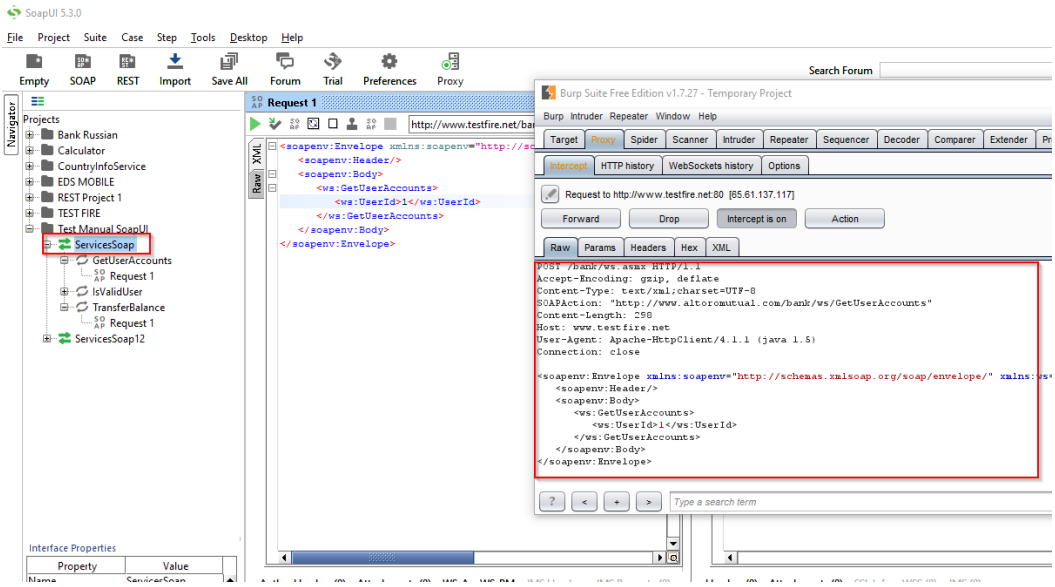


Figura 51. Integración de SOAP UI con BurpSuite

Por medio de Burp Suite se ha interceptado el request "GetUserAccounts" del servicio para este ejemplo práctico se seguirá utilizando el servicio http://www.testfire.net/bank/ws.asmx?WSDL usado en ejemplos prácticos previos, para este caso se ha configurado el proxy de ambas herramientas.

Al ejecutar el request en la herramienta Burp Suite esta nos da a conocer información del servicio así como la versión del servidor donde está alojado.

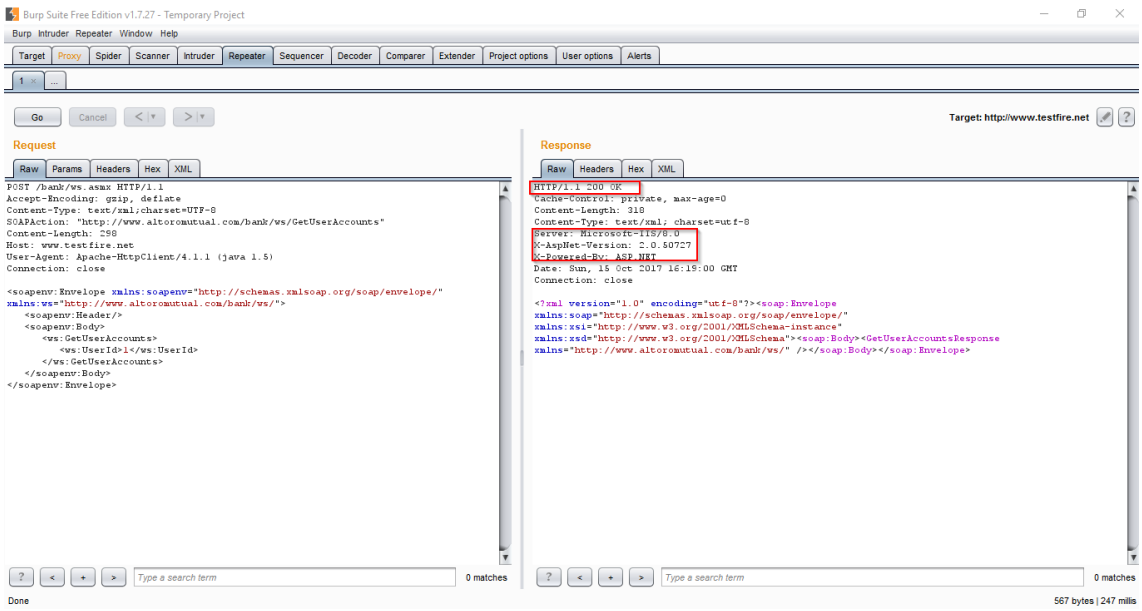


Figura 52. Información de Servicio

Es necesario saber el tipo del parámetro “UserId” para ello es necesario leer el archivo WSDL, en él se encuentra que el tipo del parámetro “UserId” es de tipo Integer y que al proporcionar el valor apropiado se puede obtener un matriz de datos de cuentas.

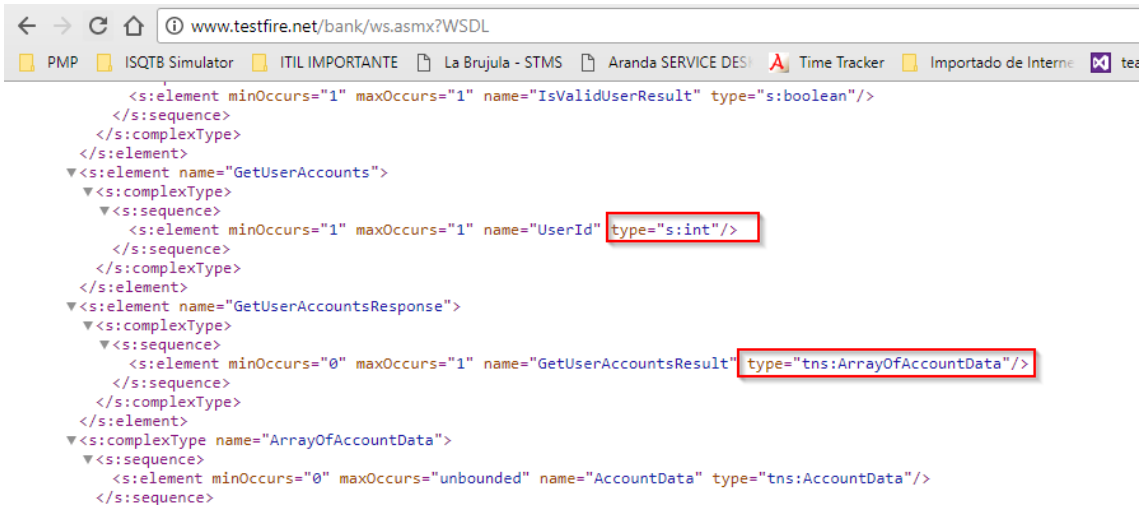


Figura 53. Análisis de Archivo WSDL

Es necesario utilizar realizar un fuzz sobre el valor de entrada con el fin de obtener esta matriz de cuentas, Burp Suite tiene la función de Intruder que permite cargar un conjunto de payloads de parámetros.

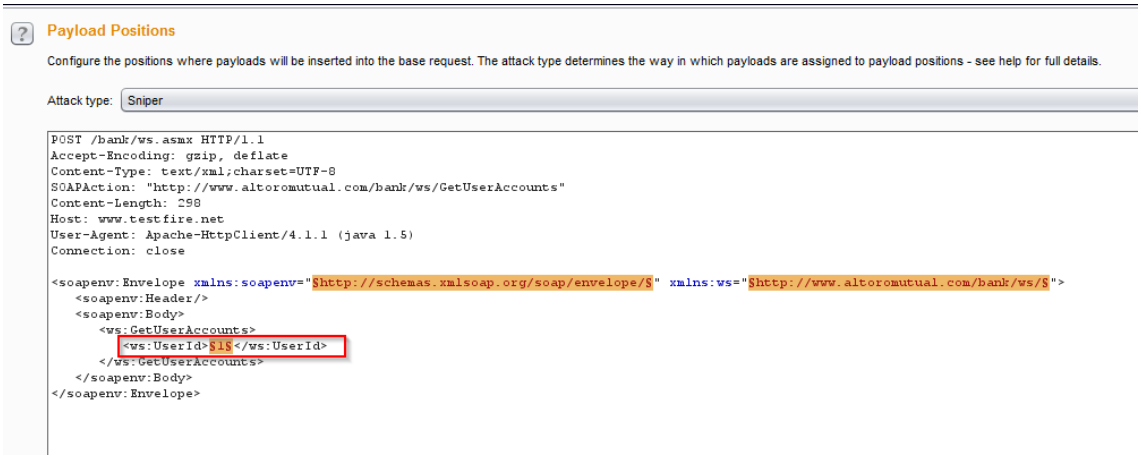


Figura 54. Carga de Payloads

Los payloads son cargados de manera manual 1 a 30 fuzz de parámetros “UserID”

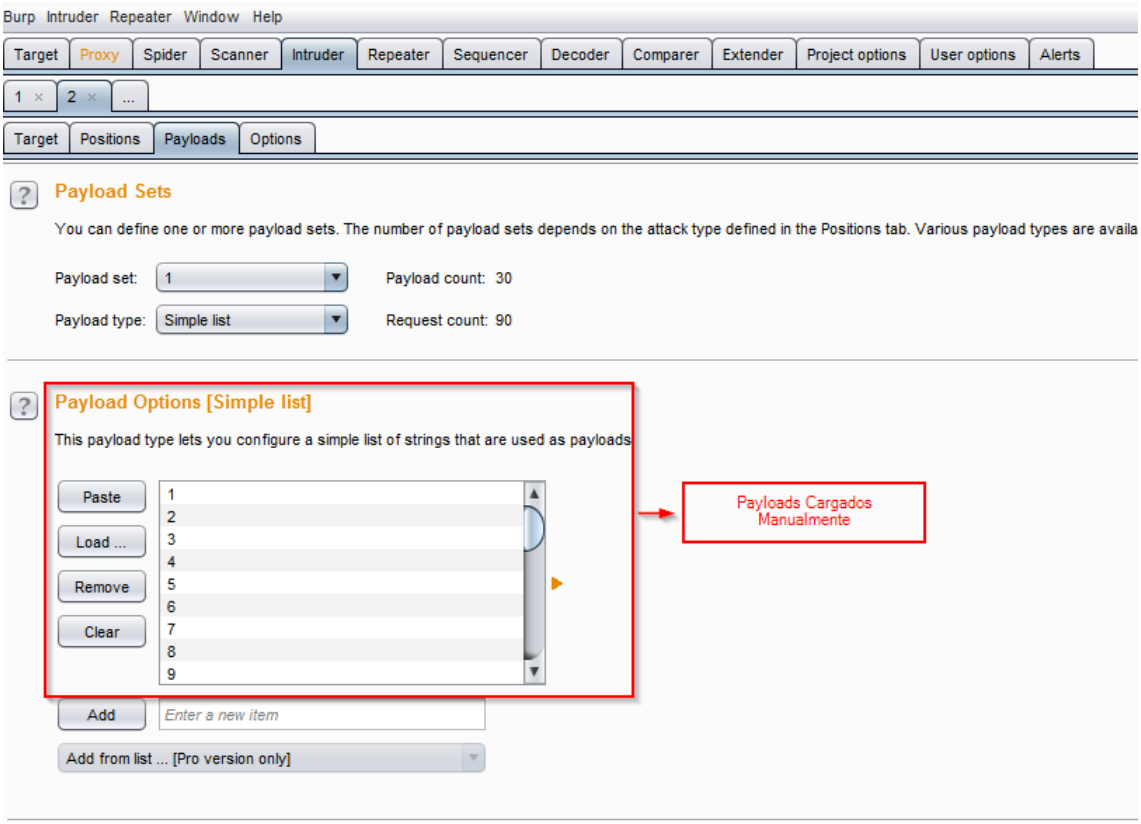


Figura 55. Payloads Manualmente

Al cargar todos los valores se puede empezar el ataque, en este caso el componente Intruder carga el payloads con todos los valores ingresados en el request “UserId”.

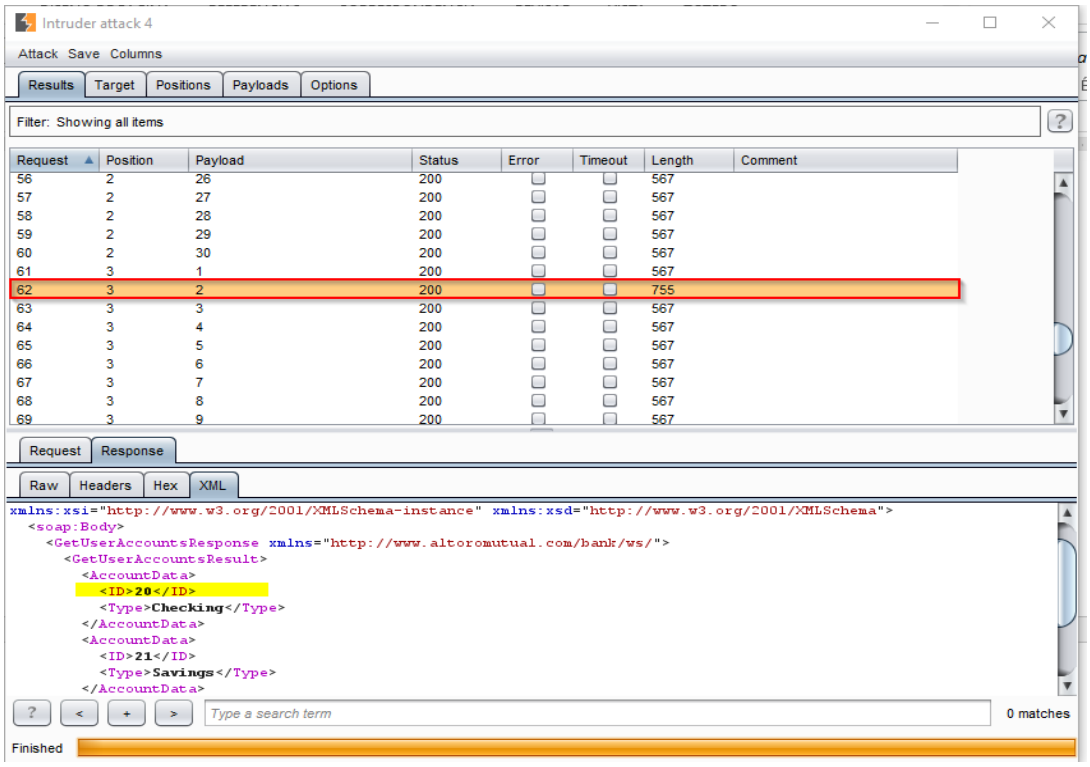


Figura 56. Intruder Attack

Después de llevarse a cabo el ataque, el componente intruder arroja un lista de resultados, el cual se puede apreciar el status code de request enviado y el length del response, la estrategia es buscar los responses que cuyo status code sea 200 y los que en su length difieran en este caso de 567.

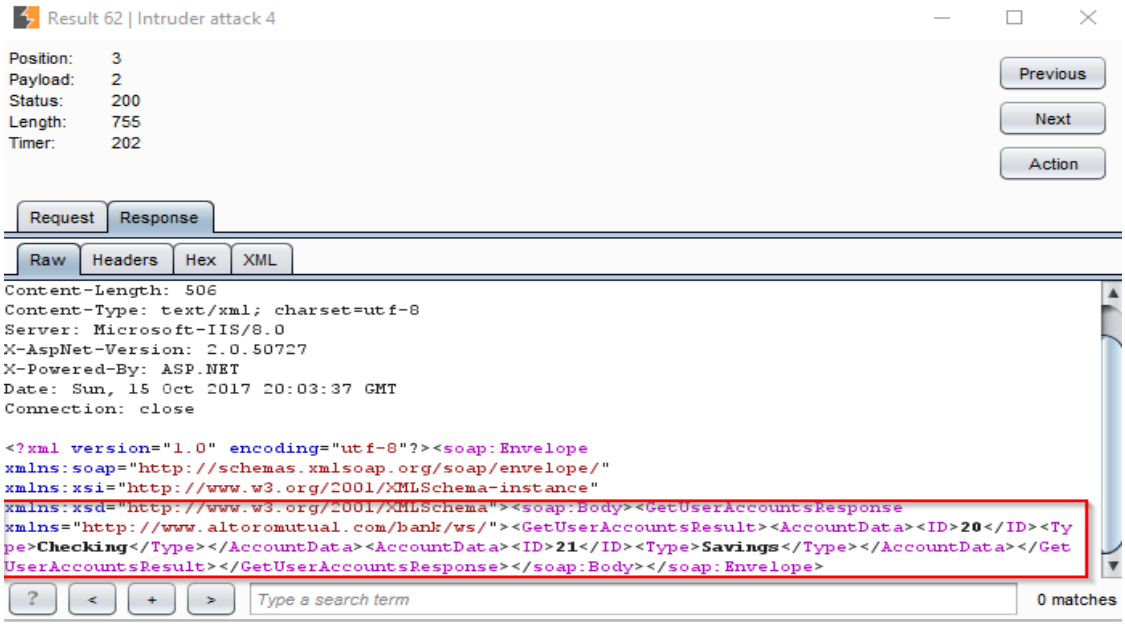


Figura 57. Análisis de Intrudder Attack

Como resultado se obtiene que el cliente con identificación 2 tiene dos cuentas bancarias, entre ellas están una cuenta de cheques “Checking” y cuenta de ahorros “Savings” y de esta forma se ha vulnerado por medio del fuzzing el web services.

8. Tablas Comparativas (Fuente Internet).

Las herramientas para el penetration testing en Web Services recomendadas por Owasp Cheat Sheet han sido puestas a pruebas por medio de Aplicaciones Web Vulnerables como es el caso de WAVSEP - Web Application Vulnerability Scanner Evaluation Project que se encuentra del Proyecto de Directorio de Aplicaciones Web Vulnerables recomendados por OWASP (OWASP Vulnerable Web Applications Directory Project/Pages/Offline, 2016) [21].

“WAVSEP es una aplicación web vulnerable diseñada para ayudar a evaluar las características, la calidad y la precisión de los escáneres de vulnerabilidades de aplicaciones web” (Chen, WAVSEP, 2014) [7].

La metodología que se llevó a cabo para evaluar las herramientas, estuvieron basados en las siguientes pruebas:

8.1 Test Análisis de Vectores de Entrada [8].

Las herramientas se deben evaluar de acuerdo a los vectores o parámetros de entrada, en caso de faltar esta característica el escáner no funcionara en lo absoluto, o simplemente proporcionara poco valor si no lo soporta debido a que la gran mayoría de ataques dependen de la entradas maliciosas que se envían a través de las entradas a la aplicación, por lo tanto una escáner que no pueda enviar esos valores a todos los puntos de entrada del servidor de aplicaciones no será una buena opción.

Varias Herramientas comerciales y Open Sources de penetration testing fueron sometidas a esta prueba obteniéndose el siguiente resultado:

Rank	Input Vectors	Chart	Vulnerability Scanner
1	19		Burp Suite Professional
2	17		IBM AppScan
3	16		NTOSpider
4	13		WebInspect
5	9		Netsparker, ScanToSecure
6	7		Acunetix WVS, Ammonite, Syhunt Dynamic
7	6		N-Stalker, QualysGuard WAS
8	3		JSky (Commercial Edition), ParosPro, WebCruiser Enterprise Edition

Tabla 1. Análisis de Vector de Entrada Herramientas Comerciales

Fuente: (Chen, Security Tools Benchmarking, 2014) [6]

Rank	Input Vectors	Chart	Vulnerability Scanner
1	13		IronWASP
2	11		ZAP
3	9		Netsparker Community Edition
4	8		W3AF
5	7		arachni
6	5		Acunetix WVS Free Edition
7	4		N-Stalker 2012 Free Edition, SkipFish, SQLiX, sqlmap, XSSer
8	3		safe3wvs (limited free edition), Vega, Wapiti, WebCruiser Free Edition
9	2		Andiparos, Gamja, Grabber, Grendel Scan, Mini MySqlat0r, N-Stalker 2009 Free Edition, Oedipus, openAcunetix, Paros Proxy, PowerFuzzer, ProxyStrike, Sandcat Free Edition, ScreamingCSS, Secubat, Syhunt Mini (Sandcat Mini), Uber Web Security Scanner, VulnDetector, WATOBO, WebScarab, WebSecurify (Opensource Version), WSTool, Xcobra, XSSploit, XSSS
10	1		aidSQL, crawlfish, Damn Small SQLi Scanner (DSSS), iScan, JSky Free Edition, LoverBoy, Priamos, Scrawlr, SQID (SQL Injection Digger), Web Injection Scanner (WIS)

Tabla 2. Análisis de Entrada Vectores Herramientas Open Source

Fuente: (Chen, Security Tools Benchmarking, 2014) [6]

Las herramientas comerciales recomendadas por Owasp Cheat Sheet, se destacaron en el resultado de esta primera prueba, Burp Suite Professional es la que ocupa el primer puesto en el la lista destacándose por permitir mayor número de vectores de entrada, le sigue IBM AppScan en el segundo puesto y por ultimo nos queda HP WebInspect en el puesto octavo de la lista.

#	Logo	Vulnerability Scanner		COUNT	GET	POST	COOKIE	HEADER	SECRET	PName	XML	XmlATT	XmlTAG	JSON	NetENC	AMF	JavaSER	NetSER	WCF	WCF-Bin	WebSock	DWR	Custom	DIR	FILE	Path	NestXML	NestJSON	JsonPName	Multipart	GWT	ODataID	ODataFilt
1		Burp Suite Professional		20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
2		IBM AppScan		17	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗
3		Acunetix WVS		16	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓	✓	✓	✗	✗
4		AppSpider		16	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗
5		Netsparker		16	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗
6		Netsparker Cloud		16	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✗	✗
7		Tinfoil Security		15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
8		WebInspect		13	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗

Tabla 3. Análisis de Vector de Entrada Herramientas Comerciales Recomendadas Por Owasp

Fuente: (Chen, SECTOOL, 2015) [6]

#	Logo	Vulnerability Scanner	COUNT	GET	POST	COOKIE	HEADER	SECRET	PName	XML	XmlATT	XmlTAG	JSON	.NetENC	AMF	JavaSER	.NetSER	WCF	WCF-Bin	WebSock	DWR	Custom	DIR	FILE	Path	NestXML	NestJSON	JsonPName	Multipart	GWT	ODataID	ODataFilt
1		IronWASP	13	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✘	✔	✔	✘	✘	✔	✘	✘	✘
2		arachni	11	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✘	✔	✘	✘	✘	✘	✘	✘	✘
3		ZAP	11	✔	✔	✔	✔	✘	✘	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔	✔	✔
4		Netsparker Community Edition	9	✔	✔	✔	✔	✘	✘	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✘	✘	✔	✘	✘	✘
5		W3AF	8	✔	✔	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔	✔	✘	✘	✘	✘	✘	✘	✘

Tabla 4. Análisis de Vectores de Entrada Herramienta Open Source Recomendadas por Owasp

Fuente: (Chen, SECTOOL, 2015)[6]

En esta lista tenemos a IronWASP, Arahni junto con OWASP ZAP recomendado por OWASP ocupando uno de los tres primeros puestos.

Alias	General Feature	Description	References
GET	HTTP Query String Parameters	Input parameters sent in the URL	1
POST	HTTP Body Parameters	Input parameters sent in the HTTP body	1
COOKIE	HTTP Cookie Parameters	Input parameters sent in the HTTP cookie	1
HEADER	HTTP Headers	HTTP request headers used by the application	1
SECRET	Secret HTTP Parameters	Non-visible valid HTTP parameters (such as GET to POST, etc)	
PName	HTTP Parameter Names	HTTP parameter names used by the application	
XML	XML Element Content	The content of XML elements	1
XmlATT	XML Attributes	XML attributes	1
XmlTAG	XML Tags	The names of XML tags	1
JSON	JSON Parameters	Parameters sent in JSON format	1
.NetENC	.Net PostBack Encoded Parameters	Parameters sent after undergoing .net PostBack encoding	1
AMF	Flash Action Message Format	Parameters sent in Flash AMF format	1
JavaSER	Java Serialized Objects	Parameters sent within Java serialized objects	1
.NetSER	.Net Serialized Objects / Remoting	Parameters sent within .Net serialized objects / remoting	1
WCF	.Net WCF Objects	Parameters sent in WCF requests	1
WCF-Bin	.Net Binary WCF Objects	Parameters sent in binary WCF requests	1
WebSock	HTML5 WebSockets	Direct Socket Browser-Server Communication	1
DWR	Java Direct Web Remoting	Parameters sent in DWR format	1
Custom	Custom Input Vector	Support for defining custom input vectors in the HTTP request	
DIR	Directory Name Input Vector	Support for scanning the directory section in the HTTP URL	
FILE	File Name Input Vector	Support for scanning the file name section (without extension) in the HTTP URL	
Path	HTTP Path Input Vector	Support for appending to and scanning the HTTP path	
NestXML	Nested XML In Parameter Input Vector	Support for scanning XML components which are nested in other parameters	
NestJSON	Nested JSON In Parameter Input Vector	Support for scanning JSON components which are nested in other parameters	
JsonPName	JSON Parameter Name Input Vector	Support for scanning JSON parameter names	
Multipart	Multipart Input Vector	Support for scanning Multipart values	1
GWT	GWT Input Vector	Support for scanning input sent in GWT (Google Web Toolkit) format	1
ODataID	OData Id Input Vector	Support for scanning OData ID Values	1
ODataFilt	OData Filter Input Vector	Support for scanning OData Filter Values	1

Tabla 5. Vectores de Entrada

Fuente: (Chen, SECTOOL, 2015)

8.2 Test Cobertura de Rastreo [6]

Las herramientas se deben evaluar de acuerdo a su cobertura de rastreo, cada aplicación tiene diversos métodos de descubrimiento que les ayuda a aumentar la superficie de ataque entre ello localización de recursos adicionales y métodos de entrega para atacar.

En esta prueba se evaluó la capacidad de rastreo automatizado y la cobertura de extracción de vectores de entradas debido a que esta característica es bastante importante en una aplicación para los usuarios que no son expertos en seguridad y que no pueden manipular una aplicación de forma manual.

Para probar esta característica se utiliza el proyecto llamado VIWET[8], este proyecto tiene como objetivo el análisis estadístico de los extractores de enlaces web, mide la cantidad de vectores de entrada extraidos por cada herramienta de pentesting que escanea el sitio web de VIWET.

El proyecto VIWET mide la exactitud del rastreo que realiza una herramienta de pentesting de igual modo revisa la efectividad en el momento de localizar los vectores entrada.

Se somete a estas pruebas las herramientas y este fue el resultado obtenido:

[8] **VIWET** Web Input Vector Extractor Teaser (Urgun, 2014)

Rank	Detection Accuracy	Chart	Vulnerability Scanner
1	96.00%		WebInspect
2	94.00%		Acunetix WVS, NTOSpider, Syhunt Dynamic
3	92.00%		IBM AppScan, Netsparker, QualysGuard WAS
4	44.00%		JSky (Commercial Edition)
5	19.00%		ParosPro
6	16.00%		Burp Suite Professional, N-Stalker

Tabla 6. Herramientas Comerciales Cobertura de Rastreo

Fuente: (Chen, Security Tools Benchmarking, 2014) [6]

WebInspect en cabeza en la lista con una precisión en la detección del 96% y en el tercer lugar tenemos a IBM AppScan con una precisión en la detección de 92% y de sexto lugar nos queda nuestra última aplicación comercial recomendad por OWASP Burp Suite Professional con un precisión del 16% por este motivo Owasp Cheat Sheat recomienda a IBM AppScan y WebInspect como las herramientas especiales para las fase de pruebas automatizadas y para la fase pruebas manuales Burp Suite Professional.

Rank	Detection Accuracy	Chart	Vulnerability Scanner
1	92.00%		Acunetix WVS Free Edition
2	91.00%		Netsparker Community Edition
3	73.00%		ZAP
4	50.00%		Vega
5	48.00%		SkipFish
6	44.00%		Wapiti
7	39.00%		ProxyStrike
8	28.00%		WebSecurify (Opensource Version)
9	19.00%		arachni, W3AF
10	16.00%		N-Stalker 2012 Free Edition
11	14.00%		Grendel Scan
12	10.00%		Andiparos, Paros Proxy
13	1.00%		WATOBO, XSSer

Tabla 7. Herramientas Open Source Cobertura de Rastreo

Fuente: (Chen, Security Tools Benchmarking, 2014)[6]

En esta ocasión la herramienta ZAP queda en puesto número 3 con una precisión de detección de 73%.

8.3 Test Detección de Redireccionamiento Malicioso (Phishing) [6]

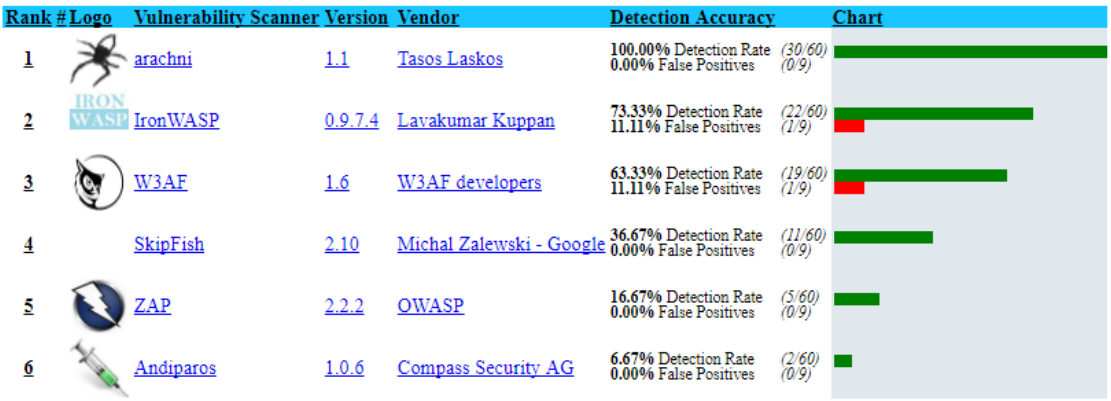
Las herramientas se deben evaluar con el fin de determinar la precisión en la detección de Redireccionamiento Maliciosos (Phishing) que se basa en el la redirección de sitios web que redirigen el navegador a direcciones controladas por un atacante.

Rank	#	Logo	Vulnerability Scanner	Version	Vendor	Detection Accuracy	Chart
1			Netsparker	4.1.1.0	Netsparker Ltd	100.00% Detection Rate (30/60) 0.00% False Positives (0/9)	
1			Netsparker Cloud	2015-06-16	Netsparker Ltd	100.00% Detection Rate (30/60) 0.00% False Positives (0/9)	
1			Tinfoil Security	X	Tinfoil Security	100.00% Detection Rate (30/60) 0.00% False Positives (0/9)	
2			Acunetix WVS	10.5	Acunetix	100.00% Detection Rate (30/60) 11.11% False Positives (1/9)	
2			N-Stalker	X	N-Stalker	100.00% Detection Rate (30/60) 11.11% False Positives (1/9)	
3			Burp Suite Professional	1.7.03	PortSwigger	76.67% Detection Rate (23/60) 0.00% False Positives (0/9)	
4			WebInspect	10.1.177.0	HP Application Security Center	50.00% Detection Rate (15/60) 0.00% False Positives (0/9)	
5			IBM AppScan	9.0.0.999 / 8.8.0.0	IBM Security Systems Division	36.67% Detection Rate (11/60) 11.11% False Positives (1/9)	
6			AppSpider	6.0	Rapid7	33.33% Detection Rate (10/60) 0.00% False Positives (0/9)	
7			QualysGuard WAS	2014-01-21	Qualys, Inc.	3.33% Detection Rate (1/60) 0.00% False Positives (0/9)	

Tabla 8. Herramientas Open Source Cobertura de Rastreo

Fuente: (Chen, SECTOOL, 2015)[8]

La barra verde indica el porcentaje de los casos de prueba vulnerables a Pishing descubiertos por las aplicaciones y la barra roja indica el porcentaje de casos de pruebas que indicaron falsos positivos, en esta lista WebInspect ocupa el cuarto lugar con un porcentaje de 50% de casos de pruebas vulnerables descubiertos y un 0% de falsos positivos, IBM AppScan le sigue con 36.7% y un 11.11 % de falsos positivos.



Rank	#	Logo	Vulnerability Scanner	Version	Vendor	Detection Accuracy	Chart
1			arachni	1.1	Tasos Laskos	100.00% Detection Rate (30/60) 0.00% False Positives (0/9)	
2			IronWASP	0.9.7.4	Lavakumar Kuppan	73.33% Detection Rate (22/60) 11.11% False Positives (1/9)	
3			W3AF	1.6	W3AF developers	63.33% Detection Rate (19/60) 11.11% False Positives (1/9)	
4			SkipFish	2.10	Michal Zalewski - Google	36.67% Detection Rate (11/60) 0.00% False Positives (0/9)	
5			ZAP	2.2.2	OWASP	16.67% Detection Rate (5/60) 0.00% False Positives (0/9)	
6			Andiparos	1.0.6	Compass Security AG	6.67% Detection Rate (2/60) 0.00% False Positives (0/9)	

Tabla 9. Herramientas Open Source Phishing

Fuente: (Chen, SECTOOL, 2015)[8]

Para el análisis de herramientas OpenSource Owasp Zap queda en la posición número cinco con un 16.7% de casos vulnerables descubierto y 0% de falsos positivos.

8.4 Test Detección de Inyección SQL [6]

Entre los ataques más importantes según el top 10 de Owasp, las herramientas se deben someter a pruebas con el fin de medir la precisión en la detección de este ataque.

Dentro de los resultados cabe mencionar que la barra verde representa la precisión en la detección, y la roja representa los falsos positivos detectados por la herramienta.

Rank	#	Logo	Vulnerability Scanner	Version	Vendor	Detection Accuracy	Chart
1			Acunetix WVS	10.5	Acunetix	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
1			IBM AppScan	9.0.0.999 / 8.8.0.0	IBM Security Systems Division	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
1			Netsparker	4.1.1.0	Netsparker Ltd	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
1			Netsparker Cloud	2015-06-16	Netsparker Ltd	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
1			Tinfoil Security	X	Tinfoil Security	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
1			WebInspect	10.1.177.0	HP Application Security Center	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
2			Burp Suite Professional	1.7.03	PortSwigger	100.00% Detection Rate (136/136) 10.00% False Positives (1/10)	
3			Syhunt Dynamic	5.0.0.7	Syhunt	100.00% Detection Rate (136/136) 50.00% False Positives (5/10)	
4			AppSpider	6.0	Rapid7	97.06% Detection Rate (132/136) 0.00% False Positives (0/10)	
5			N-Stalker	X	N-Stalker	96.32% Detection Rate (131/136) 0.00% False Positives (0/10)	

Tabla 10. Herramientas Comerciales Inyección SQL

Fuente: (Chen, SECTOOL, 2015)[8]

En estos resultado tenemos a IBM AppScan con un buena precisión, al igual que HP WebInpect, en esta también se puede observar como Burp Suite Professional tiene un 10% de falsos positivos.

Unified List | Commercial Scanners | **Free / Open Source Scanners**

Rank	#	Logo	Vulnerability Scanner	Version	Vendor	Detection Accuracy	Chart
1			arachni	1.1	Tasos Laskos	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
1			sqlmap	1.0	sqlmap developers	100.00% Detection Rate (136/136) 0.00% False Positives (0/10)	
2			Vega	1.0	Subgraph	100.00% Detection Rate (136/136) 20.00% False Positives (2/10)	
2			Wapiti	2.3.0	OWASP	100.00% Detection Rate (136/136) 20.00% False Positives (2/10)	
3			ZAP	2.2.2	OWASP	100.00% Detection Rate (136/136) 30.00% False Positives (3/10)	
4			Syhunt Mini (Sandcat Mini)	4.4.3.0	Syhunt	100.00% Detection Rate (136/136) 50.00% False Positives (5/10)	
5			IronWASP	0.9.7.4	Lavakumar Kuppan	99.26% Detection Rate (135/136) 0.00% False Positives (0/10)	
6			WATOBO	0.9.19	Andreas Schmidt	83.09% Detection Rate (113/136) 60.00% False Positives (6/10)	
7			Andiparos	1.0.6	Compass Security AG	77.21% Detection Rate (105/136) 40.00% False Positives (4/10)	
7			Paros Proxy	3.2.13	MileSCAN Technologies	77.21% Detection Rate (105/136) 40.00% False Positives (4/10)	
8			SkipFish	2.10	Michal Zalewski - Google	76.47% Detection Rate (104/136) 0.00% False Positives (0/10)	
9			Netsparker Community Edition	3.1.6.0	Netsparker Ltd	72.06% Detection Rate (98/136) 30.00% False Positives (3/10)	
10			Sandcat Free Edition	4.0.0.1	Syhunt	58.82% Detection Rate (80/136) 20.00% False Positives (2/10)	

Tabla 11. Herramientas Open Source Inyección SQL

Fuente: (Chen, SECTOOL, 2015)[8]

8.5 Test Configuración, Usabilidad, Estabilidad, Performance y Report [6]

Las herramientas se deben evaluar con respecto a que tan sencilla pude llegar a ser su configuración, interfaz amigable y fácil de usar, que tan estable es, que tan rápido puede llegar hacer en el momento del análisis de seguridad y de acuerdo a la normativa PCI si la herramienta soporta la generación de reportes donde se indique el certificado de conformidad de la

exploración, un resumen de vulnerabilidades donde se enumere vulnerabilidades por componentes y muestre si cada componente escaneado recibió una puntuación de aprobación y cumplió con los requisitos de análisis y por último el reporte de detalles de vulnerabilidades que es la lista general de vulnerabilidades que muestra cumplimiento estado (paso / error) y detalles de todas las vulnerabilidades detectadas durante la exploración. (ASV Program Guide v3.0, 2017)

#	Logo	Vulnerability Scanner		GUI	Config	Usage	Stability	Performance	Report	ScanLog	Pause	Session
1		Acunetix WVS		✓	Very Simple	Very Simple	Very Stable	Fast	✓	✓	✓	✓
2		Ammonite		✓	Very Simple	Very Simple	Stable	Fast	✓	✓	✓	✓
3		AppSpider		✓	Very Simple	Very Simple	Stable	Fast	✓	✓	✓	?
4		Burp Suite Professional		✓	Very Simple	Very Simple	Very Stable	Very Fast	✓	✓	✓	✓
5		IBM AppScan		✓	Simple	Simple	Very Stable	Fast	✓	✓	✓	✓
6		JSky (Commercial Edition)		✓	Very Simple	Simple	Stable	Fast	✓	✗	✓	✓
7		Netsparker		✓	Very Simple	Very Simple	Very Stable	Fast	✓	✓	✓	✓
8		Netsparker Cloud		✓	Very Simple	Very Simple	Very Stable	Fast	✓	✓	✗	✓
9		N-Stalker		✓	Very Simple	Very Simple	Very Stable	Very Fast	✓	✗	✗	?
10		ParosPro		✓	Simple	Simple	Stable	Fast	✓	✗	✗	?
#	Logo	Vulnerability Scanner		GUI	Config	Usage	Stability	Performance	Report	ScanLog	Pause	Session
11		QualysGuard WAS		✓	Simple	Simple	Very Stable	Very Fast	✓	✓	✓	✓
12		Syhunt Dynamic		✓	Simple	Simple	Stable	Fast	✓	✓	✓	✓
13		Tinfoil Security		✓	Very Simple	Very Simple	Very Stable	Fast	✓	✓	✗	✓
14		WebCruiser Enterprise Edition		✓	Very Simple	Very Simple	Stable	Very Fast	✓	✗	✓	?
15		WebInspect		✓	Very Simple	Very Simple	Stable	Fast	✓	✓	✓	✓

Tabla 12. Herramientas Comerciales Configuración Usabilidad

Fuente: (Chen, SECTOOL, 2015)[8]

#	Logo	Vulnerability Scanner	GUI	Config	Usage	Stability	Performance	Report	ScanLog	Pause	Session
41		WebCruiser Free Edition	✔	Very Simple	Simple	Stable	Fast	✔	✖	✔	?
42		WebScarab	✔	Very Simple	Very Simple	Stable	Fast	✔	✔	✖	?
43	WS	WebSecurify (Opensource Version)	✔	Very Simple	Very Simple	Very Stable	Fast	✔	✖	✔	?
44		WSTool	✔	Complex	Complex	Unstable	Fast	✔	✖	✖	?
45		Xcobra	✔	Simple	Simple	Stable	Fast	✔	✖	✖	?
46		XSSer	✔	Very Simple	Simple	Stable	Fast	✔	✔	✖	?
47		XSSploit	✔	Very Simple	Very Simple	Stable	Fast	✔	✖	✔	?
48		XSSS	✖	Simple	Complex	Fragile	Fast	✖	✖	✖	?
49		ZAP	✔	Very Simple	Very Simple	Very Stable	Fast	✔	✔	✔	✔

Tabla 13. Herramientas Open Sources Configuración Usabilidad

Fuente: (Chen, SECTOOL, 2015)[8]

De acuerdo a estos resultados se destaca en la lista Burp Suite Professional como una de las que mejor cumple todas estas características, IBM AppScan y WebInspect también se encuentran.

9.0 Conclusiones

Las organizaciones, los desarrolladores y los evaluadores deben otorgar a los servicios web una importancia equivalente a las aplicaciones web, dado que hoy en día no es un campo que se explote mucho en el área de seguridad. Muchas de las metodologías o buenas prácticas que se encuentras son pobremente diseñadas y las herramientas disponibles no son lo suficiente efectivas para probar y encontrar las vulnerabilidades que pueden afectar a un web services en el mundo real.

En el momento de analizar una herramienta para penetration testing en web services la ausencia de entornos de pruebas se hace notoria, un pentester que quiera expandir sus conocimientos y aprender más técnicas de pentesting sobre aplicaciones web tiene a disposición una amplia variedad de aplicaciones que son vulnerables para prácticas. Algunos ejemplos de estas aplicaciones WebGoat by OWASP, Mutillidae by Adrian “irongeek” Crenshaw y Dan Vulnerable Web Aplication (DVWA) by Ryan Dewhurst. Estas aplicaciones proveen a los penetration tester una serie de páginas web que contienen un gran conjunto de vulnerabilidades. Un penetration tester puede practicar sus habilidades haciendo uso de estas aplicaciones e investigar sobre herramientas que permitan hacer un análisis efectivo.

Actualmente, los sistemas que proveen web services vulnerables para pruebas de seguridad son muy escasos, esto significa que aquellos pentester que estén buscando la forma de mejorar sus habilidades, ampliar sus conocimientos y buscar e investigar sobre herramientas que permitan hacer un análisis efectivo sobre web services están limitados en temas de ambientes de pruebas, lo que implica que muchos recurran a los web services que están en ambientes productivos o por ultimo construyan sus propios web services de pruebas.

En la fase de automatización en el penetration testing en web services cualquier herramienta automatizada ayudara a conseguir abarcar una buena cobertura de pruebas, obtener buenos resultados y reducir horas de trabajo, sin embargo estas herramientas tienden a reportar falsos positivos y falsos

negativos, de modo que siempre será necesario realizar una fase manual de pruebas que permitan garantizar un correcto análisis de seguridad, dado que hoy día no existen herramientas que brinden una cobertura en profundidad para las pruebas de web services desde una perspectiva de seguridad, esto se debe a que la mayoría de herramientas pruebas de web services han sido construidas únicamente para asegurar la calidad funcional.

10 .0 Bibliografía Especifica

[1] About The Open Web Application Security Project. (08 de 09 de 2017). Obtenido dehttps://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project

[2] About The Open Web Application Security Project. (08 de 09 de 2017). Obtenido de OWASP ORG: https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project

[3] ASV Program Guide v3.0. (Febrero de 2017). Obtenido de PCI Security Standars Council: https://www.pcisecuritystandards.org/documents/ASV_Program_Guide_v3.0.pdf

[4] Bochumm, U. R. (s.f.). WS-Attacker. Obtenido de RUB-NDS/WS-Attacker: https://github.com/RUB-NDS/WS-Attacker

[5] C., B. G. (3 de Agosto de 2004). UDDI (Universal Description Discovery and Integration). Obtenido de desarrolloweb.com: https://desarrolloweb.com/articulos/1589.php

[6] Chen, S. (6 de Febrero de 2014). Security Tools Benchmarking. Obtenido de sectooladdict.blogspot: http://sectooladdict.blogspot.com.ar/2014/02/wavsep-web-application-scanner.html

[7] Chen, S. (2014). WAVSEP. Obtenido de sectooladdict/wavsep: https://github.com/sectooladdict/wavsep

[8] Chen, S. (2015). SECTOOL. Obtenido de sectoolmarket: http://www.sectoolmarket.com/price-and-feature-comparison-of-web-application-scanners-unified-list.html

[9] Cross-site Scripting (XSS). (2016). Obtenido de OWASP ORG: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[10] Falkenberg, A. (31 de Octubre de 2015). Soap Array Attack. Obtenido de ws-attacks.org: http://www.ws-attacks.org/Soap_Array_Attack

[11] Falkenberg, A. (31 de Octubre de 2015). SOAPAction Spoofing. Obtenido de ws-attacks.org: http://www.ws-attacks.org/SOAPAction_Spoofing

[12] Falkenberg, A. (31 de 10 de 2015). XML Injection. Obtenido de ws-attacks.org: http://www.ws-attacks.org/XML_Injection

[13] Falkenberg, A. (31 de 10 de 2015). Xpath Injection. Obtenido de ws-attacks.org: http://www.ws-attacks.org/Xpath_Injection

[14] Fuzzing Scan. (2017). Obtenido de Soap UI by SmartBear: https://www.soapui.org/security-testing/security-scans/fuzzing-scan.html

[15] IBM. (20 de Junio de 2017). Altoro Mutual. Obtenido de testfire: http://www.testfire.net/

[16] Moral, L. G. (2014). Curso de Ciberseguridad y Hacking Ético 2013. Punto Rojo Libros.

[17] Owasp. (3 de 12 de 2012). Inyección SQL. Obtenido de Owasp: https://www.owasp.org/index.php/Inyecci%C3%B3n_SQL

[18] OWASP. (14 de Noviembre de 2017). OWASP Zed Attack Proxy Project. Obtenido de owasp: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[19] OWASP Cheat Sheet Series. (13 de 11 de 2017). Obtenido de OWASP Cheat Sheet Series: https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series

[20] OWASP Testing Guide v3 Table of Contents. (4 de 7 de 2013). Obtenido de Testing: WS Information Gathering (OWASP-WS-001): https://www.owasp.org/index.php/Testing:_WS_Information_Gathering_(OWASP-WS-001)

[21] OWASP Vulnerable Web Applications Directory Project/Pages/Offline. (29 de Diciembre de 2016). Obtenido de owasp: https://www.owasp.org/index.php/OWASP_Vulnerable_Web_Applications_Directory_Project/Pages/Offline

[22] Panda, N. (2013). Web Services Penetration Testing Part 1. Obtenido de INFOSEC INSTITUTE: http://resources.infosecinstitute.com/web-services-penetration-testing-part-1-2/

[23] Panda, N. (2013). Web Services Penetration Testing Part 3: Automation with AppScan and Webinspect. Obtenido de Infosec Institute: http://resources.infosecinstitute.com/web-services-pen-test-part-3-automation-appscan-webinspect/

[24] Panda, N. (2013). Web Services Penetration Testing Part 4: Manual Testing with SOA Client. Obtenido de Infosec Institute: http://resources.infosecinstitute.com/web-services-pen-test-part-4-manual-testing-soa-client/

[25] Panda, N. (2013). Web Services Penetration Testing Part 5: Manual Testing with soapUI. Obtenido de Infosec Institute: http://resources.infosecinstitute.com/web-services-penetration-testing-part-5-manual-testing-soapui/

[26] Panda, N. (2013). Web Services Penetration Testing, Part 2: An Automated Approach With SoapUI Pro. Obtenido de Infosec Institute: http://resources.infosecinstitute.com/web-services-penetration-testing-part-2-automated-approach-soapui-pro/

[27] Panda, N. (2013). Web Services Penetration Testing, Part 6: Fuzzing Parameters with Burp. Obtenido de Infosec Institute: http://resources.infosecinstitute.com/web-services-penetration-testing-part-6-fuzzing-parameters-burp/

[28] Panda, N. (s.f.). Web Services Penetration Testing Part 7: More Fuzzing with Burp. Obtenido de Infosec Institute: http://resources.infosecinstitute.com/web-services-penetration-testing-part-7-fuzzing-burp/

[29] SMARTBEAR. (s.f.). SQL Injection. Obtenido de https://www.soapui.org/security-testing/security-scans/sql-injection.html

[30] Szuster, P. M. (s.f.). Seguridad en SOA. Obtenido de IBM Security Forum V Foro de Seguridad y Tecnologia: https://www.ibm.com/uy/news/events/securityforumv/pdf/websphere_datapower.pdf

[31] Vera, M. (2013). I2B Intelligence to Business. Obtenido de I2B Intelligence to Business: http://www.i2btech.com/blog-i2b/tech-deployment/que-se-entiende-por-soa-y-cuales-son-sus-beneficios/

[32] W3C. (s.f.). Obtenido de https://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

[33] W3C. (s.f.). Obtenido de W3C: http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML

[34] Web Service Security Testing Cheat Sheet. (5 de 11 de 2016). Obtenido de Web Service Security Testing Cheat Sheet: https://www.owasp.org/index.php/Web_Service_Security_Testing_Cheat_Sheet#Web_Services_Security_Testing_Cheat_Sheet_Introduction

[35] XML security: Preventing XML bombs. (Febrero de 2006). Obtenido de TechTarget: http://searchsoftwarequality.techtarget.com/answer/XML-security-Preventing-XML-bombs