Web Services: Enumeration and Profiling

Abstract

Web services hacking begins with the Web Services Definition Language or WSDL. A WSDL file is a major source of information for an attacker. Examining a WSDL description provides critical information like methods, input and output parameters. It is important to understand the structure of a WSDL file, based on which one should be able to enumerate web services. The outcome of this process is a web services profile or matrix. The scope of this paper is restricted to understanding this process. Once this is done, attack vectors for web services can be defined. The scope of attack vectors will be covered in the next paper.

Shreeraj Shah
Co-Author: "Web Hacking: Attacks and Defense" (Addison Wesley, 2002) and published several advisories on security flaws.

n e t - s q u a r e
http://www.net-square.com
shreeraj@net-square.com

Table of Contents

Acknowledgement

Lyra Fernandes for her help on documentation.

Introduction

Web services assessment can begin with a corporate name or some other such bit of information. This simple hint offers a wealth of information that needs to be unearthed. Focus first on locating single or multiple access points for a particular corporate. The methodology, which includes web services footprinting, discovery and search, is described in another paper (http://packetstormsecurity.org/papers/web/Defense_using_mod_security.pdf). Once an access point for a web service is uncovered, the next obvious step is to extract information from it.

Web services are deployed to invoke remote calls over HTTP/HTTPS. To make calls such as these, requires that information about the calls be shared with the end client. In the past, during the days of CORBA, developers used to share IDL (Interface Definition Language) files providing the required information over the network. Now, in the days of web services this has changed to WSDL (Web Services Definition Language). WSDL is major source for information and can help in the enumeration process. We shall go over the enumeration process in subsequent sections.

Example of Web Services

To demonstrate the process of enumeration let us consider a web store that provides web services to customers. The store has three objectives in offering web services to its customers:

1. Introducing the store (whoami, purpose, etc.)
2. Sharing product information (price, type, etc.)
3. Sharing information on product promotion

The structure of the code should resemble the snippet below if these sample web services are developed on a .Net framework.

[Code Snippet for store.asmx]

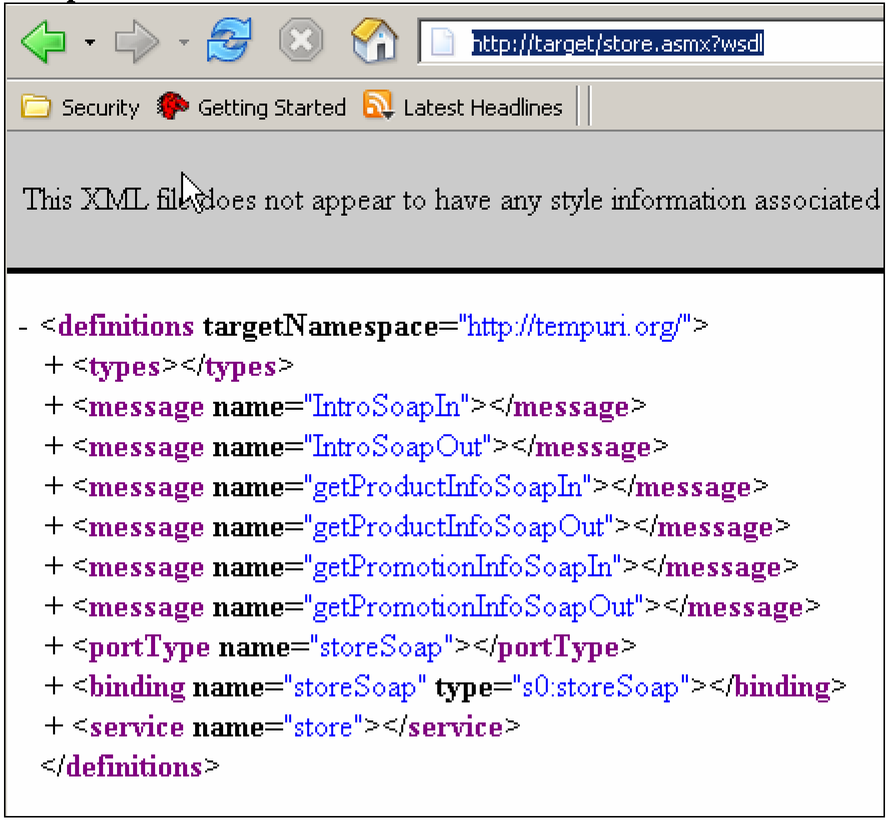
```
<%@ WebService Language="c#" Class="store" %>
using System;
using System.Web.Services;
using System.Data.SqlClient;
using System.IO;
public class store
{
[WebMethod]
        public string Intro()
        {
        return "This is web store's web services. Please use for required information";
        }


[WebMethod]
        public string getProductInfo(int id)
        {
                ......... Code......
                return productinfo;
        }

[WebMethod]
        public string getPromotionInfo(string filepath)
        {
                ......... Code......
                return promotionfile;
        }
}
```

WSDL stores all critical information about web services and must be shared with rest of the world on the Internet. This is the only way others can invoke web services over a network.

[Snapshot of the WSDL file]



This XML file does not appear to have any style information associated

```
- <definitions targetNamespace="http://tempuri.org/">
  + <types></types>
  + <message name="IntroSoapIn"></message>
  + <message name="IntroSoapOut"></message>
  + <message name="getProductInfoSoapIn"></message>
  + <message name="getProductInfoSoapOut"></message>
  + <message name="getPromotionInfoSoapIn"></message>
  + <message name="getPromotionInfoSoapOut"></message>
  + <portType name="storeSoap"></portType>
  + <binding name="storeSoap" type="s0:storeSoap"></binding>
  + <service name="store"></service>
  </definitions>
```

The above snapshot displays a higher-level tag list only. All tags will be elucidated in detail as we walk through the entire WSDL.

Enumeration of web services

http://target/store.asmx?wsdl is our target access point returned by UDDI (Universal Description Discovery and Integration).

Next, we dissect the *.wsdl* file and enumerate required information. WSDL is like a jigsaw puzzle that we need to meticulously piece together.

Enumerating service from WSDL

Enumerating Service and Location

[WSDL snippet]

```
<service name="store">
  <port name="storeSoap" binding="s0:storeSoap">
    <soap:address location="http://target/store.asmx" />
  </port>
</service>
```

The <service> tag will provide name of the service and access location for the same. This information provides binding location for the client and the service to use with *invoke*. This information can be obtained from the following regex patterns “<service.*?>” and “<.*location.*[^>]>”.

Enumerating operations & methods

[WSDL snippet]

```
<portType name="storeSoap">
   <operation name="Intro">
     <input message="s0:IntroSoapIn" />
     <output message="s0:IntroSoapOut" />
   </operation>
   <operation name="getProductInfo">
     <input message="s0:getProductInfoSoapIn" />
     <output message="s0:getProductInfoSoapOut" />
   </operation>
   <operation name="getPromotionInfo">
     <input message="s0:getPromotionInfoSoapIn" />
     <output message="s0:getPromotionInfoSoapOut" />
   </operation>
</portType>
```

`<portType>` is the next important tag which contains names all methods that can be invoked remotely. It also presents the *type of invoke* supported. In our example, the name shown is “storeSoap” which indicates that the only type of invoke possible is SOAP. Similarly, sometimes web services also support the GET and POST methods.

<operation> represents the method name of *invoke*. The methods available are:

```
<operation name="Intro">
<operation name="getProductInfo">
<operation name="getPromotionInfo">
```

Here's the information that each operation provides –

- Intro – provides introduction to store.
- getProductInfo – provides product information for any product id specified.
- getPromotionInfo – this provides promotion detail file.

Input and output messages for particular method are provided through *input* and *output* tags. Inputs and outputs are obtained for specific methods.

```
<input message="s0:IntroSoapIn" />
<output message="s0:IntroSoapOut" />
```

Deriving data types for methods

[WSDL snippet]

```
<operation name="getProductInfo">
    <input message="s0:getProductInfoSoapIn" />
    <output message="s0:getProductInfoSoapOut" />
</operation>
```

```
<message name="getProductInfoSoapIn">
   <part name="parameters" element="s0:getProductInfo" />
  </message>
  <message name="getProductInfoSoapOut">
   <part name="parameters" element="s0:getProductInfoResponse" />
</message>
```

```
<s:element name="getProductInfo">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="id" type="s:int" />
        </s:sequence>
      </s:complexType>
</s:element>

<s:element name=" getProductInfoResponse ">
      <s:complexType>
        <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="getProductInfoResult"
            type="s:string" />
        </s:sequence>
      </s:complexType>
</s:element>
```

As shown above, we can derive information for an operation name and use this derived information to get message blocks mentioned in the *WSDL* file. These blocks point us to element tags. As shown, an element tag has a relative data type for each message, for either input or output.

To elaborate, the method ***getProductInfo*** has the following inputs and outputs:

Input:

Variable:	**id**
Type:	**int**

Output:

Variable:	**getProductInfoResult**
Type:	**string**

In similar fashion, an entire profile for all methods and variables can be created. See matrix below.

Web services matrix for web store (profile):

Method	Input	Output
Intro	-	-
getProductInfo	Int (id)	String
getPromotionInfo	String (file)	String

This enumerated information can be very useful in the web services assessment process. On the basis of this, assessment test plans can be built. Test plans for web services assessment is not covered in the scope of this paper but handled in another one.

Extracting information using *wsdl.exe*

`WSDL.exe` is a small utility that is bundled with the .Net framework. This utility can be used to build an enumeration matrix. The purpose of this utility is to create proxy code from this *wsdl* file itself.

Example:
Our target is http://target/store.asmx

Let us run the wsdl tool against this.

```
C:\Documents and Settings\Administrator>wsdl /namespace:storeproxy /language:cs
http://localhost/store.asmx
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.0.3705.0]
Copyright (C) Microsoft Corporation 1998-2001. All rights reserved.

Writing file 'C:\Documents and Settings\Administrator\store.cs'.

C:\Documents and Settings\Administrator>
```

This command creates a proxy code in the C# language with *storeproxy* as the namespace. This proxy code can then be used to enumerate information like we did in the preceding section. An output file *store.cs* is created in the same folder. See directory listing below.

```
03/11/2005  03:59p            4,316 store.cs
```

Let us look at this proxy code in order to derive key information.

[Code Snippet]

```
namespace storeproxy {
   using System.Diagnostics;
   using System.Xml.Serialization;
   using System;
   using System.Web.Services.Protocols;
   using System.ComponentModel;
   using System.Web.Services;
```

As shown above proxy code for storeproxy is created by the *WSDL* tool. Some critical information can be gathered by closely inspecting the public methods.

[Code Snippet]

```
public store() {
            this.Url = "http://localhost/store.asmx";
        }

public string Intro()
public string getProductInfo(int id)
public string getPromotionInfo(string file)
```

The three functions help us in deriving this table.

Method	Input	Output
Intro	-	-
getProductInfo	Int (id)	String
getPromotionInfo	String (file)	String

We now have a complete set of inputs for web services deployed on the corporate network. i.e. our sample web store. With this information in place, we can commence web services assessment with a different set of attack vectors.

Conclusion

The ease which WSDL allows for deploying web services on a network has brought into sharp focus the need to address security concerns that arise by having web services assessments included in the scope of other security assessment assignments. This calls for a fair degree of comprehension of WSDL, the basic constituent in the web services domain. Enumerating and profiling web services then becomes an extremely simple task.

This paper has sought to explain the methodology of enumerating and profiling web services in a simple, straightforward manner. Various other methods of extracting the same information about a web service's methods, input and output may exist, and users are free to follow methodologies of their choice.