[逆向][Writeup]EIS2016 chkflag - .NET程序逆向

转载

weixin_34009794 于 2017-02-16 10:00:00 发布 104 收藏
原文链接：http://www.cnblogs.com/gsharpsh00ter/p/6404396.html
版权
这是一道比较简单的逆向题，难点之处在于chkflag.exe是.NET编译生成的，如果用IDA进行逆向难度很大，因此本题主要考察.NET逆向工具的使用。题目中使用的二进制文件可以从我的github上下载：
https://github.com/gsharpsh00ter/reverse

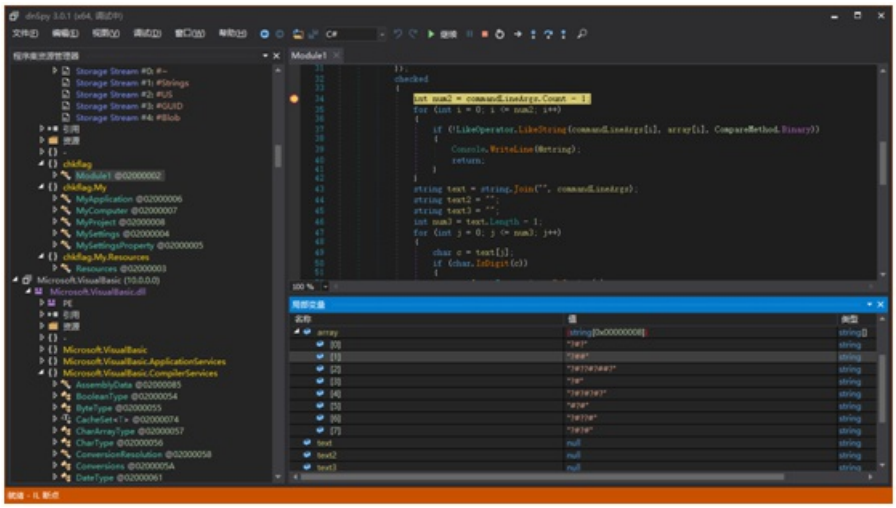
0x01 工具准备

在对.NET程序进行反编译时，常用的一个工具为.NET Reflector，但是目前该工具是收费的。本例中我们使用另一款开源的.NET反编译工具dnSpy。dnSpy是一款开源的基于ILSpy发展而来的.NET程序的编辑、反编译和调试神器，可对.NET程序进行反编译和动态调试，界面友好，功能强大。

dnSpy下载地址：

https://github.com/0xd4d/dnSpy/releases

下载后解压直接运行即可，但是需要先安装.NET框架。安装后进行逆向分析和调试的界面如下：



0x02 逆向

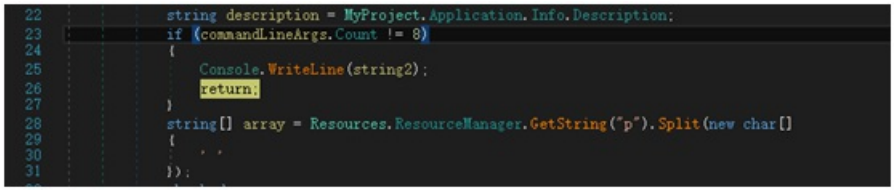
将chkflag.exe拖入dnspy，查看入口函数，如下：

```
// chkflag.Module1
// Token: 0x06000001 RID: 1 RVA: 0x00002048 File Offset: 0x00000248
[STAThread]
public static void Main()
{
    ReadOnlyCollection<string> commandLineArgs = MyProject.Application.CommandLineArgs;
    ulong num = 917137303011111000uL;
    string @string = Resources.ResourceManager.GetString("f");
    string string2 = Resources.ResourceManager.GetString("i");
    string description = MyProject.Application.Info.Description;
    //命令行必需有8个参数
    if (commandLineArgs.Count != 8)
    {
        Console.WriteLine(string2);
        return;
```

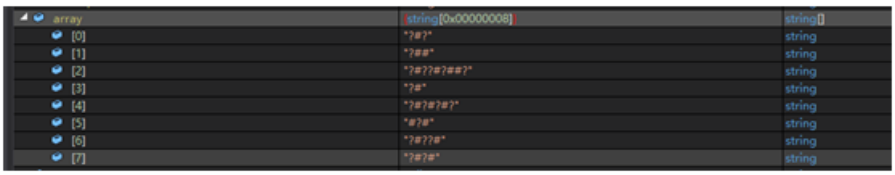
```
        return;
    }
    string[] array = Resources.ResourceManager.GetString("p").Split(new char[]
    {
        ' '
    });
    checked
    {
        int num2 = commandLineArgs.Count - 1;
        //命令行的每一个参数需要符合array中规定的格式
        for (int i = 0; i <= num2; i++)
        {
            if (!LikeOperator.LikeString(commandLineArgs[i], array[i], CompareMethod.Binary))
            {
                Console.WriteLine(@string);
                return;
            }
        }
        //将所有的命令行参数拼接成一个字符串
        string text = string.Join("", commandLineArgs);
        string text2 = "";
        string text3 = "";
        int num3 = text.Length - 1;
        //对拼接的字符串进行处理，如果某字符为数字，则拼接到text3，反之拼接到text2
        for (int j = 0; j <= num3; j++)
        {
            char c = text[j];
            if (char.IsDigit(c))
            {
                text3 += Conversions.ToString(c);
            }
            else
            {
                text2 += Conversions.ToString(c);
            }
        }
        //如果拼接后的text3逆序之后转化为整数为num(91713730301111000uL)，并且text2逆序之后为description，则将在命令行打印flag
        if (ulong.Parse(Strings.StrReverse(text3)) == num && Strings.StrReverse(text2).Equals(description))
        {
            Console.WriteLine("EIS{" + string.Join("_", commandLineArgs) + "}");
            return;
        }
        Console.WriteLine(@string);
    }
}
```

0x03 分析

处理逻辑比较简单。首先，程序会检查命令行参数的个数，参数必须有8个，否则会打印“参数错误”而退出。



然后，程序会检查每一个参数是否符合指定的格式，参数格式在array中指定，通过动态调试，可以得到array的内容如下：



即：["?#?","?##","?#??#?##?","?#","?#?#?#?","#?#","?#??#","?#?#"]，其中'?'表示字母，'#'表示数字，比如"?#?"表示第一个参数应该是"字母+数字+字母"的形式。

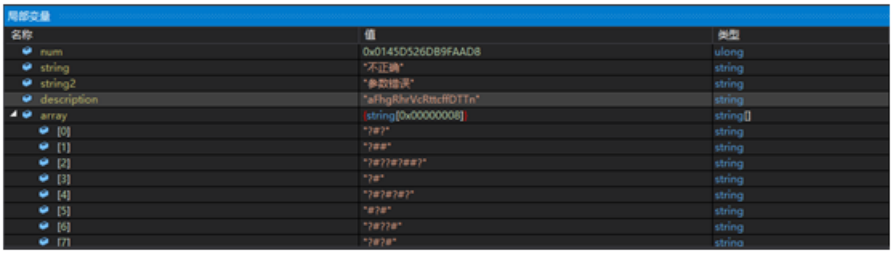
之后程序将命令行所有的参数进行拼接，成为一个字符串。

拼接完命令行参数后，会继续对拼接的字符串进行处理，如果某字符为数字，则拼接到text3，反之拼接到text2

最后，对text2和text3进行判断。如果text3逆序之后转化为整数为num，并且text2逆序之后为description，则将在命令行打印flag。

Num的值在反编译得到的代码中可以看到，为91713730301111000UL。

Description可以通过动态调试得到，为"aFhgRhrVcRttcffDTTn"



通过上述逻辑，用如下python代码进行逆向计算，即可得到我们向chkflag传递的命令行参数：

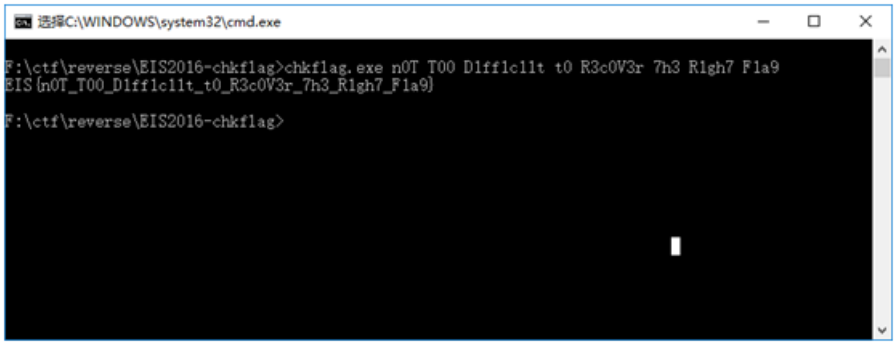
```
#!/usr/bin/python2

patterns     = ["?#?","?##","?#??#?##?","?#","?#?#?#?","#?#","?#??#","?#?#"]
num            = "91713730301111000"
desc         = "aFhgRhrVcRttcffDTTn"
args         = []
numidx          = 0
descidx      = 0
for pattern in patterns:
    arg = ""
    for k in xrange(0, len(pattern)):
        if pattern[k] == '?':
            arg += desc[::-1][descidx]
            descidx += 1
        else:
            arg += num[::-1][numidx]
            numidx += 1
    args.append(arg)
print args
```

打印出的命令行参数为：

['n0T', 'T00', 'D1ff1c11t', 't0', 'R3c0V3r', '7h3', 'R1gh7', 'F1a9']

在命令行运行chkflag，并传递上述参数，可得flag:



最终flag为：

EIS{n0T_T00_D1ff1c11t_t0_R3c0V3r_7h3_R1gh7_F1a9}

转载于:https://www.cnblogs.com/gsharpsh00ter/p/6404396.html