




DDCTF2019逆向分析前俩题WriteUP

原创

秃桔子  于 2019-05-08 21:00:00 发布  351  收藏

微信搜索桔子科研，关注并领取更多有趣的源码。

本文链接：https://blog.csdn.net/qq_41482054/article/details/90706327

版权

DDCTF2019逆向分析前俩题WriteUP

DDCTF2019

笔者做了前俩道题。冷不丁过去一个月了、现在在此做一下WriteUp：题目链接：

1: [题目1](#)

2: [题目2](#)

reverse1:writeup:

1、程序打开后如下所示

□

2、查壳结果为UPX的壳。

□

3、UPXSHELL脱壳之后用GHIDRA打开后如下

□

这就程序大致的流程。我们的关键点在于确定其算法。

4、动态调试。

□

当我输入aaa时，程序内变成了===

□

当我输入bbbb时，程序内部变成了<<<<

□

经计算 我们输入的值在程序逆向思维出来是9E-该数即可。

5、解密

□

成功拿到flag

2、reverse2 ——WriteUP

1.题目打开后如下：所示

□

2.PEID查壳结果

□

ASP的壳，看样子需要手动脱壳.

3.用OD脱壳。

(1) 进程序之后按F9找到printf入口点。随后用ODdump直接脱下来

□

步骤-plugins->ODdumpEx->dump process

□

4.用IDA打开后查看大致的代码逻辑

□

看来这题是控制输入的字符串，来对v1进行赋值。并且将结果与DDCTF{reverse+}进行比对。

5.找到要分析函数模块。

□

经过分析，现然这两个代码块是我们需要主分析的地方。现在大致内容我们已经理清了，下一步是带壳调试。

6.找到输入函数的规则。

□

经调试，找到了scanf的输入点。这里我们去看一下对应的IDA函数的位置。

□

这里的函数过于复杂。引起生理上的不适，因此我选择动态调试分析。就是暴力方法测出输入函数的格式。

经多次检测，输入格式为 以下几个字符0123456789ABCDEF即可，其余的均会爆出invalid input错误。

因此到了下一阶段动态调试。

7.输入字符串后变成的样子。

这里有个需要注意的地方，当程序加壳的时候，会出现这种db 字符，要使用OD analysis重新分析下。

□

经多次分析，我发现输入的一串字符串变成了base64编码形式

□

经分析，我发现我输入的一串字符串会被变成16进制，并使用base64编码与结果中的reverse+进行比对。

那接下来就是reverse+解码的时候了。因为base64经编码之后的文本为乱码显示，无法了解内部。我们需要手动解码

□

8.BASE手动解码。找出REVERSE+对应的64位字符重新编码后，成功拿到FLAG

□

posted @ 2019-05-08 21:00 [godoforange](#) [阅读\(...\)](#) [评论\(...\)](#) [编辑](#) [收藏](#)