

bugkuCTF平台逆向题第五道love题解

原创

Megrez 于 2019-03-26 14:15:31 发布 470 收藏

分类专栏: [ctf](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_43445167/article/details/88818579

版权



[ctf 专栏收录该内容](#)

3 篇文章 0 订阅

订阅专栏

看了看网上的writeup 这尼玛都不是人的思维, 自己写个writeup

1. 首先 第一件事 看标题 标题一般有提示 对吧。然而看不出个卵。
2. 运行一波 嗯哼 盲猜调strcmp进行比对, flag就会入栈, 打开od, 开始疯狂寻找

```
002F57FB 8380 54FFFFFF add ecx,dword ptr ss:[ebp-0x9C]
002F5801 8B95 54FFFFFF mov edx,dword ptr ss:[ebp-0x9C]
002F5807 888C15 6CFFFFFF mov byte ptr ss:[ebp+edx-0x94],cl
002F580E EB C0 jmp short reverse_.002F57D0
002F5810 8D85 6CFFFFFF lea eax,dword ptr ss:[ebp-0x94]
002F5816 50 push eax
002F5817 E8 ACB8FFFF call reverse_.002F10C8
002F581C 83C4 04 add esp,0x4
002F581F 8BF4 mov esi,esp
002F5821 50 push eax
002F5822 68 34A02F00 push reverse_.002FA034
002F5827 8D8D 6CFFFFFF lea ecx,dword ptr ss:[ebp-0x94]
002F582D 51 push ecx
002F582E FF15 84B12F00 call dword ptr ds:[<&ucrtbased.strncmp>] ucrtbase.strncmp
002F5834 83C4 0C add esp,0xC
002F5837 3BF4 cmp esi,esp
https://blog.csdn.net/qq_43445167
```

3. 卧槽 上面那串奇异的字符一定是flag。提交 你懂的
4. 开始思考 我发现我输入的字符串没有入栈, 这其中一定有蹊跷!

```
002F5810 8D85 6CFFFFFF lea eax,dword ptr ss:[ebp-0x94]
002F5816 50 push eax
002F5817 E8 ACB8FFFF call reverse_.002F10C8
002F581C 83C4 04 add esp,0x4
002F581F 8BF4 mov esi,esp
002F5821 50 push eax
002F5822 68 34A02F00 push reverse_.002FA034
002F5827 8D8D 6CFFFFFF lea ecx,dword ptr ss:[ebp-0x94]
002F582D 51 push ecx
002F582E FF15 84B12F00 call dword ptr ds:[<&ucrtbased.strncmp>] ucrtbase.strncmp
002F5834 83C4 0C add esp,0xC
002F5837 3BF4 cmp esi,esp
002F5839 E8 E9B8FFFF call reverse_.002F1127
002F583E 85C0 test eax,eax
002F5840 74 0F jc short reverse_.002F5851 这个要跳
Stack address=00F9FD28, (ASCII "MUIzNDU2")
ecx=00000055
https://blog.csdn.net/qq_43445167
```

5. 然后我选择去找输入函数, 跟踪我输入的字符串。

```
2F575D E8 CDBBFFFF call reverse_.002F132F
2F5762 83C4 04 add esp,0x4
2F5765 8D45 D8 lea eax,dword ptr ss:[ebp-0x28]
2F5768 50 push eax
2F5769 68 8C7B2E00 push reverse_.002F7B8C
```

2F577E	E8 02BCFFFF	call reverse_.002F1375	输入函数
2F5773	83C4 08	add esp,0x8	
2F5776	8BF4	mov esi,esp	
2F5778	6A 28	push 0x28	
2F577A	8D45 F4	lea eax,dword ptr ss:[ebp-0xC]	
2F577D	50	push eax	
2F577E	8D4D D8	lea ecx,dword ptr ss:[ebp-0x28]	
2F5781	51	push ecx	
2F5782	E8 41B9FFFF	call reverse_.002F10C8	极可能为加密函数
2F5787	83C4 04	add esp,0x4	
2F578A	50	push eax	
2F578B	8D55 D8	lea edx,dword ptr ss:[ebp-0x28]	
2F578E	52	push edx	
2F578F	E8 2AB9FFFF	call reverse_.002F10BE	
2F5794	83C4 0C	add esp,0xC	
2F5797	50	push eax	
2F5798	8D85 6CFFFFFF	lea eax,dword ptr ss:[ebp-0x94]	
2F579E	50	push eax	

eax=002BF8BC, (ASCII "12314560")

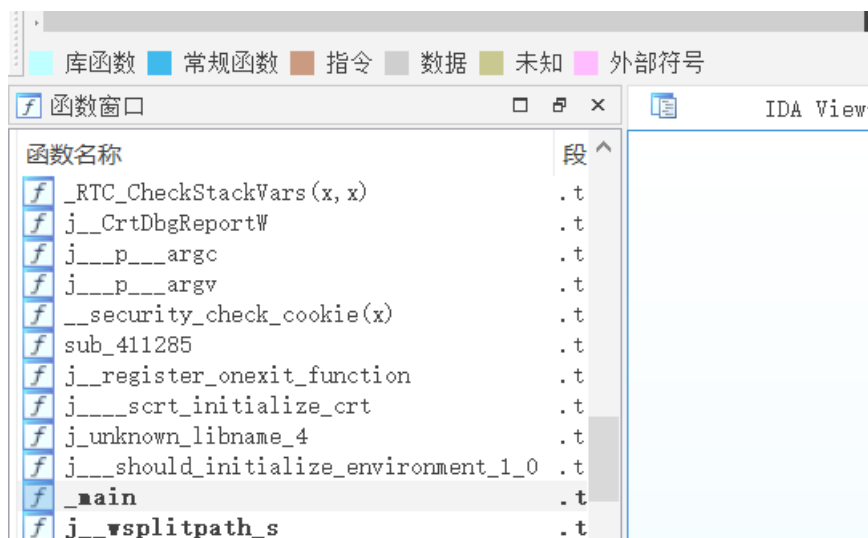
6. 如图 我发现我的字符串出现了 我就觉得它对我的字符串动了手脚 事实证明...我错了 是它的下一个函数对我的字符串动了手脚... (我就是f8看出来的)

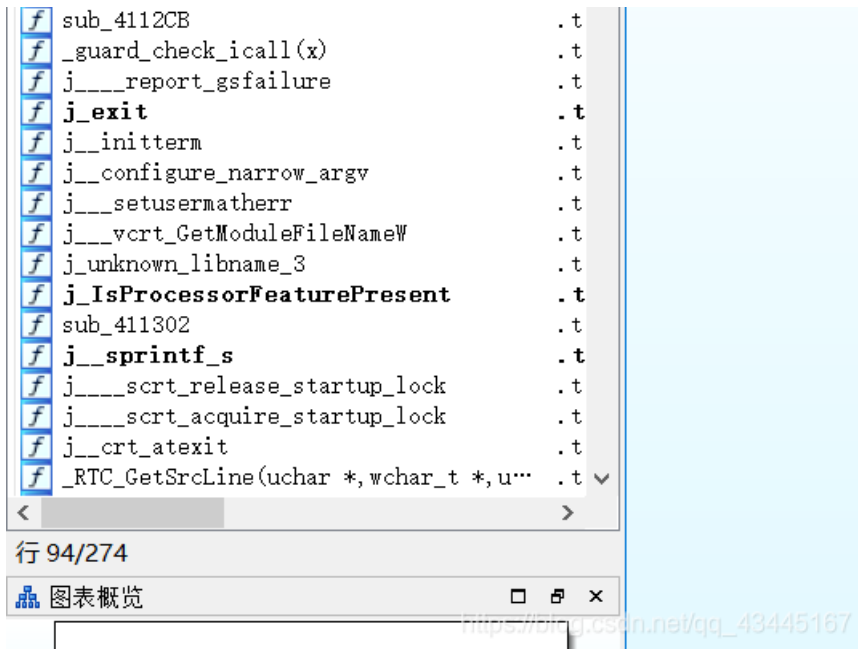
002F5778	6A 28	push 0x28	
002F577A	8D45 F4	lea eax,dword ptr ss:[ebp-0xC]	
002F577D	50	push eax	
002F577E	8D4D D8	lea ecx,dword ptr ss:[ebp-0x28]	
002F5781	51	push ecx	
002F5782	E8 41B9FFFF	call reverse_.002F10C8	极可能为加密函数
002F5787	83C4 04	add esp,0x4	
002F578A	50	push eax	
002F578B	8D55 D8	lea edx,dword ptr ss:[ebp-0x28]	
002F578E	52	push edx	
002F578F	E8 2AB9FFFF	call reverse_.002F10BE	
002F5794	83C4 0C	add esp,0xC	
002F5797	50	push eax	
002F5798	8D85 6CFFFFFF	lea eax,dword ptr ss:[ebp-0x94]	
002F579E	50	push eax	
002F579F	FF15 88812F00	call dword ptr ds:[<&ucrtbased.strncpy>	ucrtbase.strncpy
002F57A5	83C4 0C	add esp,0xC	

eax=027EAF0, (ASCII "HTIzMTQ1NjA=")

7.

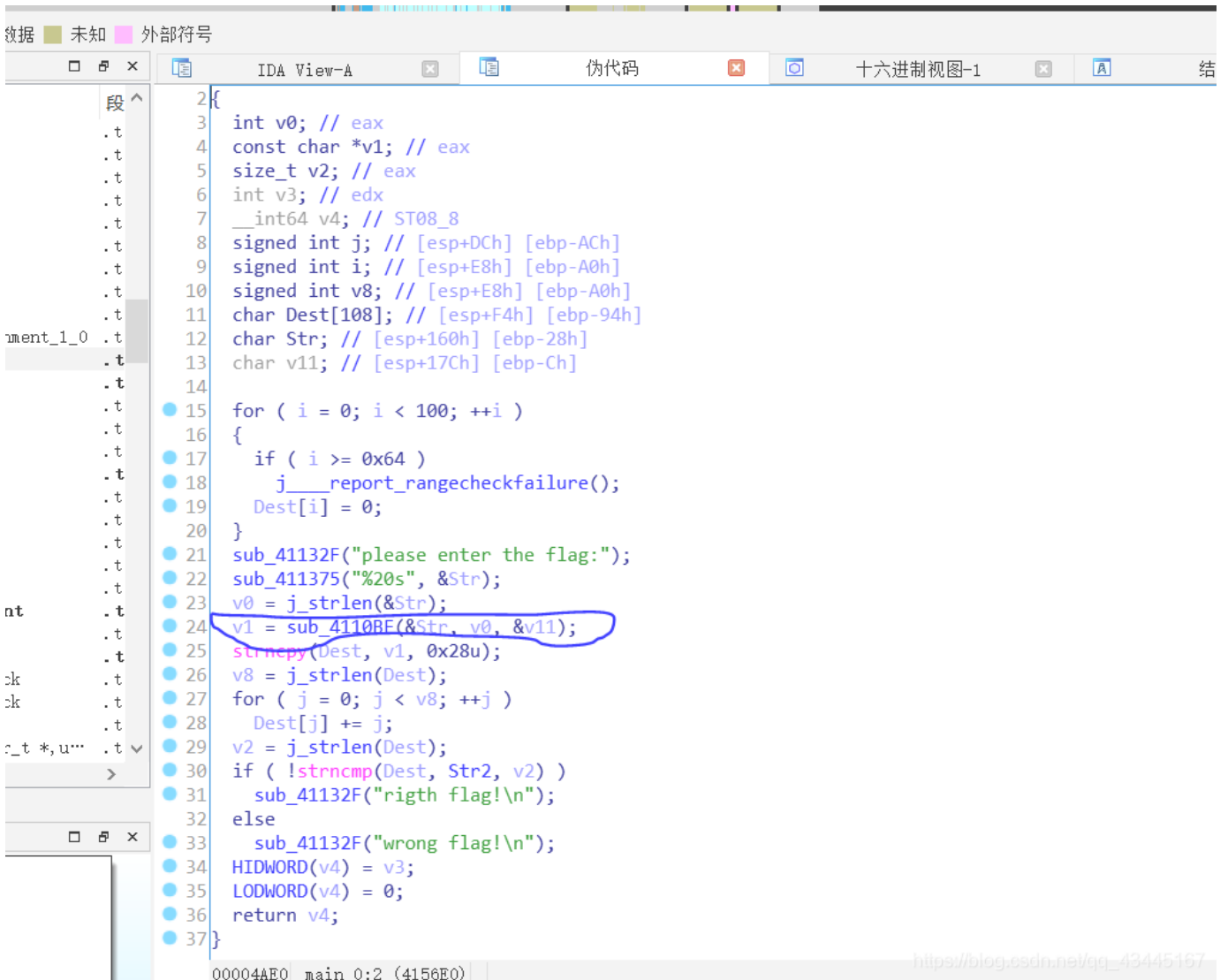
8. 这个字符串与图2中的字符串像的一匹，尼玛脉络出来了。题目将输入的flag加密后与另一个字符串比较。开IDA！逆向算





法！（先找主函数）

9. 直接f5看伪码，往里跟



10.自己分析一边就知道这个函数有问题了，往里跟就能看到加密函数。

然后 算法逆向不出来。。。。。。。。。

去看writeup了，base64，气得我吐血。就当熟悉base64的加密过程了。



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)