

【reversing.kr逆向之旅】 Replace的writeup

原创

iqiqiya 于 2018-11-10 11:26:47 发布 642 收藏

分类专栏: [我的逆向之路 -----reversing.kr](#) [我的CTF之路](#) 文章标签: [Replace reversing.kr](#) [逆向](#) [writeup](#) [reversing.kr](#)

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/xiangshangbashaonian/article/details/83927224>

版权



[我的逆向之路](#) 同时被 3 个专栏收录

108 篇文章 10 订阅

订阅专栏



[-----reversing.kr](#)

11 篇文章 0 订阅

订阅专栏



[我的CTF之路](#)

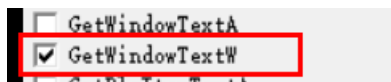
92 篇文章 5 订阅

订阅专栏

无壳 vc++程序

载入Olydbg动态调试分析就好

可以通过下API断点GetWindowTextW来找到关键代码



因为它就是用来捕获我们的输入的 随意输入(这里可以发现只可以输入数字 长度什么的倒是没有限制)

就像下面一样 可以看到地址很大 在系统领空 user32.dll 所以这里不用管

| | | | |
|----------|---------------|---------------------------------|-------------------|
| 766C386B | 8985 38FFFFFF | mov dword ptr ss:[ebp-0xC8],eax | 验证是否将输入比较完了 |
| 766C3871 | 85C0 | test eax,eax | |
| 766C3873 | 74 33 | je short user32.766C38A8 | 输入的长度<=9 大于时转移 |
| 766C3875 | 83C2 02 | add edx,0x2 | |
| 766C3878 | 8995 2CFFFFFF | mov dword ptr ss:[ebp-0xD4],edx | |
| 766C387E | 8D50 D0 | lea edx,dword ptr ds:[eax-0x30] | |
| 766C3881 | 85D2 | test edx,edx | |
| 766C3883 | 78 23 | js short user32.766C38A8 | |
| 766C3885 | 83FA 09 | cmp edx,0x9 | |
| 766C3888 | 7F 1E | jg short user32.766C38A8 | |
| 766C388A | 3BCE | cmp ecx,esi | |
| 766C388C | 0F83 8AC30200 | jnb user32.766EFC1C | |
| 766C3892 | 6BC9 0A | imul ecx,ecx,0xA | |
| 766C3895 | 899D 28FFFFFF | mov dword ptr ss:[ebp-0xD8],ebx | |
| 766C389B | 03CA | add ecx,edx | |
| 766C389D | 8B95 2CFFFFFF | mov edx,dword ptr ss:[ebp-0xD4] | |
| 766C38A3 | 0FB702 | movzx eax,word ptr ds:[edx] | |
| 766C38A6 | EB C3 | jnp short user32.766C386B | |
| 766C38A8 | 83BD 34FFFFFF | cmp dword ptr ss:[ebp-0xCC],0x0 | |
| 766C38AF | 75 31 | jnz short user32.766C38E2 | |
| 766C38B1 | 85FE | test edi,edi | |

user32.766CFDED

<https://blog.csdn.net/xiangshangbashaonian>

上面循环过去之后

就来到了这里

```

暂停
00401059 . 56      push esi
0040105A . FF15 9C504000 call dword ptr ds:[<&USER32.GetDlgItemInt]
00401060 . A3 D0844000 mov dword ptr ds:[0x4084D0],eax
00401065 . E8 05360000 call Replace.0040466F
0040106A . 33C0     xor eax,eax
0040106C . E9 1F360000 jmp Replace.00404690
00401071 > EB 11     jmp short Replace.00401084
00401073 . 68 34 60 40  ascii "h4`@",0
00401078 . 68 E9030000 push 0x3E9
0040107D . 56      push esi
0040107E . FF15 A0504000 call dword ptr ds:[<&USER32.SetDlgItemTextA]
00401084 > B8 01000000 mov eax,0x1
00401089 . 90      nop
0040108A . 90      nop
0040108B . 90      nop
0040108C . 90      nop
0040108D . 90      nop
0040108E . 90      nop
0040108F . 90      nop
00401090 . 5E      pop esi
00401091 . 5D      pop ebp
00401092 . C2 1000  retn 0x10
00401095 > 8B45 08   mov eax,dword ptr ss:[ebp+0x8]
00401098 . 6A 02   push 0x2
0040109A . 50      push eax
0040109B . FF15 A4504000 call dword ptr ds:[<&USER32.EndDialog>]
004010A1 . B8 01000000 mov eax,0x1
004010A6 . 5D      pop ebp
004010A7 . C2 1000  retn 0x10

hWnd = 0004071A ('Replace',class='#32770')
GetDlgItemInt
ControlID = 3E9 (1001.)
hWnd = 0004071A ('Replace',class='#32770')
SetDlgItemTextA
Correct!
Replace.0040469F
Replace.0040469F
Case 2 of switch 0040103A
Result = 0x2
hWnd = 60160646
EndDialog
Replace.0040469F
https://blog.csdn.net/xiangshangbashaonian

```

可以看到成功的标志Correct!

但是CPU窗口有两个jmp指令 也就是说无条件跳转 肯定无法成功 向上向下跟踪 可以看到只有地址00401071处有一个跳转进来 可以使我们走向成功

接着向下单步 来到下图所示

```

0040466E . 00      db 00
0040466F $ C600 90 mov byte ptr ds:[eax],0x90
00404672 ? C3     retn
00404673 ? 0081 05D08441 add byte ptr ds:[ecx+0x4084D005],al
00404679 ? 00C7   add bh,al
0040467B ? 05 16604000 add eax,Replace.00406016
00404680 ? EB 60   jmp short Replace.004046E2
00404682 ? 90      nop
00404683 ? 61     popad
00404684 . E8 00000000 call Replace.00404689
00404689 $ FF05 D0844000 inc dword ptr ds:[0x4084D0]
0040468F . C3     retn
00404690 > A1 D0844000 mov eax,dword ptr ds:[0x4084D0]
00404695 . 68 9F464000 push Replace.0040469F
0040469A . E8 EAffffff call Replace.00404689
0040469F . C705 6F464000 mov dword ptr ds:[0x40466F],0xC39000C6
004046A9 . E8 C1FFFFFF call Replace.0040466F
004046AE . 40     inc eax
004046AF . E8 BBFFFFFF call Replace.0040466F
004046B4 . C705 6F464000 mov dword ptr ds:[0x40466F],0x6E8
004046BE . 58     pop eax
004046BF . B8 FFFFFFFF mov eax,-0x1
004046C4 . E9 A8C9FFFF jmp Replace.00401071
004046C9 . 00     db 00
004046CA . 00     db 00

0x90就相当于nop指令
UNICODE "惫怜"
Replace.0040469F
https://blog.csdn.net/xiangshangbashaonian

```

可以发现 程序不管输入什么 总是走到0x004046A9这里就停止运行了

可以看到这个call跳转的地址 是将eax所对应的地址进行nop

连续两个这样的call 中间inc eax 就是eax加1 也就是说连续nop两个地址

再来看下边那个向上跳转的jmp 跳转的地址0x00401071 对应其中一个jmp 挨着的下一条指令正好是个空指令

那么我们就可以利用这两个call 将这两条指令进行nop 即可走向Correct!

这时候重点就在eax是如何与我们的输入进行运算的

输入123时 $eax=0x60160646$ 把123转成十六进制是0x7b $0x60160646-0x7b=0x601605cb$



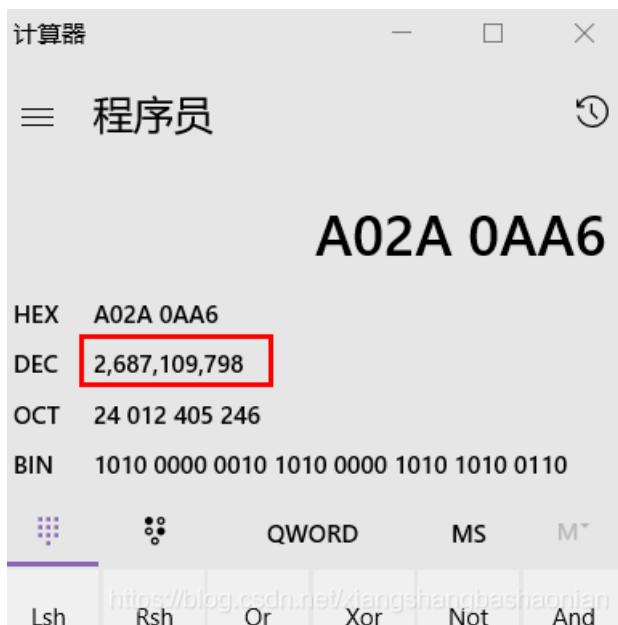
重新输入1234时 $eax=0x60160A9D$ 把1234转成十六进制是0x4d2 $0x60160A9D-0x4d2=0x601605cb$

多次输入 可以发现最终结果都是0x601605cb

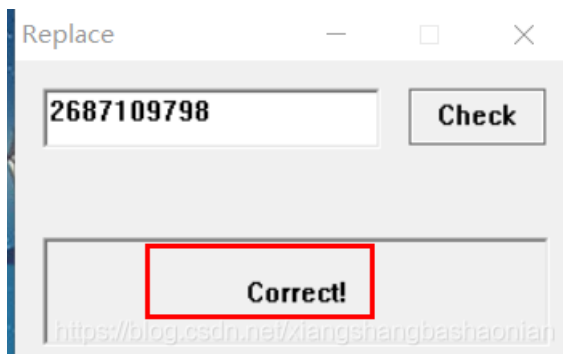
那么我们正确的输入应该要满足 $0x00401071-?=0x601605cb$

可以发现需要eax溢出才可能成立

所以 $? = 0x100401071-0x601605cb = 0xa02a0aa6$



验证2687109798 成功



参考链接:

<http://www.mottoin.com/article/reverse/88447.html>

<https://www.cnblogs.com/xiao-zhang/articles/5602504.html>