

攻防世界Reverse进阶区-流浪者-writeup

原创

y4ung 于 2020-10-06 10:40:14 发布 340 收藏 3

分类专栏: [ctf](#) 文章标签: [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_35056292/article/details/108937004

版权



[ctf 专栏收录该内容](#)

35 篇文章 0 订阅

订阅专栏

1. 介绍

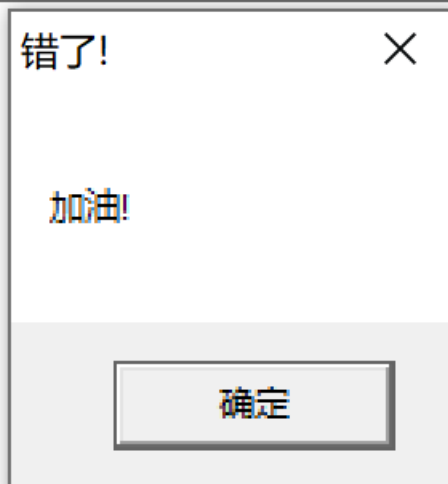
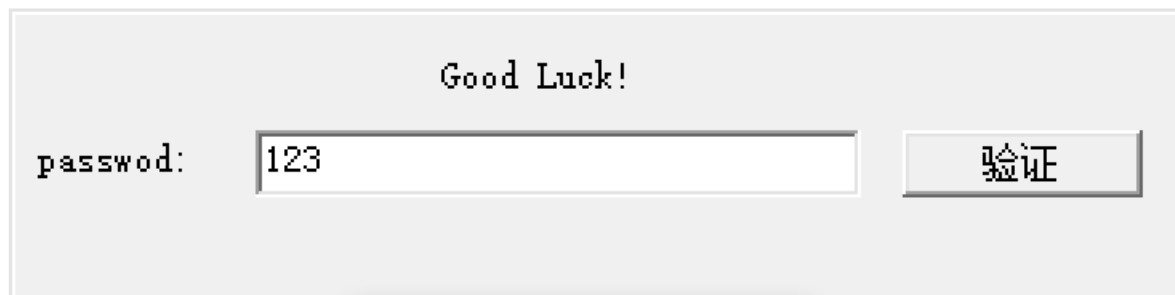
本题是xctf攻防世界中Reverse的进阶区的题流浪者

题目描述: 答案为flag{XXX}形式

2. 分析

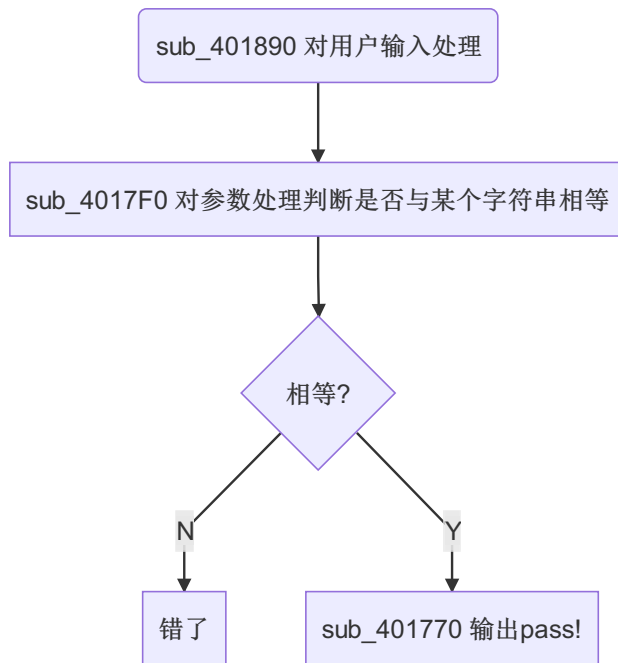
```
$ file cm.exe
```

```
cm.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```



https://blog.csdn.net/qq_35056292

根据IDA里的粗略分析, 函数的调用图如下:



IDA里strings window, 发现有个 `pass!`, 在sub_401770函数中的MessageBox中使用。

```

BOOL sub_401770()
{
    HANDLE hProcess; // ST5C_4

    MessageBoxA(0, "pass!", &Caption, 0);
    hProcess = GetCurrentProcess();
    return TerminateProcess(hProcess, 0);
}
  
```

sub_4017F0函数在sub_401890函数中被调用, 在sub_401890中, 通过GetWindowTextA读取了用户的输入保存到v1中。

```

Int GetWindowText (HWND hWnd, LPTSTR lpString, Int nMaxCount);
hWnd:带文本的窗口或控制的句柄
lpString: 指向接收文本的缓冲区的指针
nMaxCount:指定要保存在缓冲区内的字符的最大个数, 其中包含NULL字符。如果文本超过界限, 它就被截断
  
```

先将this赋值给v8, 然后调用GetDlgItem函数获取子窗口句柄。调用GetWindowTextA(v2, v1), 获取用户的输入, 保存在v1。其中v1的初值为: `v1 = (CWnd *)((char *)this + 100)`。

接下来调用了一个函数: `v3 = sub_401A30((char *)v8 + 100);`, 跟进去看, 看不太懂。

用ollydbg调试看看, 可以看出sub_401A30用于获取用户输入的长度。

00401890	- 55	push ebp	
00401891	- 8BEC	mov ebp,esp	
00401893	- 81EC B4000000	sub esp,0xB4	
00401899	- 53	push ebx	
0040189A	- 56	push esi	
0040189B	- 57	push edi	
0040189C	- 894D FC	mov [local.1],ecx	
0040189F	- 8B45 FC	mov eax,[local.1]	
004018A2	- 83C0 64	add eax,0x64	
004018A5	- 50	push eax	
004018A6	- 68 EA030000	push 0x3EA	
004018A8	- 8B4D FC	mov ecx,[local.1]	
004018AE	- E8 D1060000	call <jmp.&MFC42.#3092>	
004018B3	- 8BC8	mov ecx,eax	
004018B5	- E8 C4060000	call <jmp.&MFC42.#3874>	GetWindowText
004018BA	- 8B4D FC	mov ecx,[local.1]	
004018BD	- 83C1 64	add ecx,0x64	
004018C0	- E8 6B010000	call cm.00401A30	获取用户输入的长度
004018C5	- 50	push eax	
004018C6	- 8B4D FC	mov ecx,[local.1]	
004018C9	- 83C1 64	add ecx,0x64	
004018CC	- E8 A7060000	call <jmp.&MFC42.#2915>	GetBuffer
004018D1	- 8945 F8	mov [local.2],eax	
004018D4	- 8B4D F8	mov ecx,[local.2]	
004018D7	- 51	push ecx	
004018D8	- E8 19070000	call <jmp.&MSUCRT.strlen>	s = "0&?" strlen
004018DD	- 83C4 04	add esp,0x4	
00401F7E	<jmp.&MFC42.#3874>		

寄存器 (FPU)	
EAX	00000004 ← 输入1234, 返回值为4
ECX	0019FE90
EDX	00000004
EBX	00000001
ESP	0019F3EC
EBP	0019F4AC
ESI	7B3F3FD8 mfc42.#4234
EDI	0019FE2C
EIP	004018C5 cm.004018C5
C 0	ES 002B 32位 0(FFFFFFFF)
P 0	CS 0023 32位 0(FFFFFFFF)
A 1	SS 002B 32位 0(FFFFFFFF)
Z 0	DS 002B 32位 0(FFFFFFFF)
S 0	FS 0053 32位 3C8000(FFF)
T 0	GS 002B 32位 0(FFFFFFFF)
D 0	
0 0	LastErrr ERROR_SUCCESS (00000000)
EFL	00000212 (NO,NB,NE,A,NS,PO,GE,G)
ST0	empty 0.0
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 0.0
ST5	empty 12.000000000000000000
ST6	empty 11.16666666666666664960
ST7	empty 0.0
3 2 1 0 E S P U O Z D I	
EST 0120 Cond 0 0 0 1 Err 0 0 1 0 0 0 0 0 (1)	

接下来遍历用户输入的每一个字符，根据每个字符的ASCII值，对数组v5赋值。

先不着分析这个流程，咱们先看v5应该取什么值。也就是跟进sub_4017F0函数看一下。

```

int __thiscall sub_401890(CWnd *this)
{
    struct CString *v1; // ST08_4
    CWnd *v2; // eax
    int v3; // eax
    int v5[26]; // [esp+4Ch] [ebp-74h]
    int i; // [esp+B4h] [ebp-Ch]
    char *Str; // [esp+B8h] [ebp-8h]
    CWnd *v8; // [esp+BCh] [ebp-4h]

    v8 = this;
    v1 = (CWnd *)((char *)this + 100);
    v2 = CWnd::GetDlgItem(this, 1002); // 返回子窗口句柄
    CWnd::GetWindowTextA(v2, v1); // 将指定窗口的标题条文本（如果存在）拷贝到一个缓存区内
    v3 = sub_401A30((char *)v8 + 100); // 获取用户输入的长度
    Str = CString::GetBuffer((CWnd *)((char *)v8 + 100), v3);
    if ( !strlen(Str) )
        return CWnd::MessageBoxA(v8, &byte_4035DC, 0, 0);
    for ( i = 0; Str[i]; ++i )
    {
        if ( Str[i] > 57 || Str[i] < 48 )
        {
            if ( Str[i] > 122 || Str[i] < 97 )
            {
                if ( Str[i] > 90 || Str[i] < 65 )
                    sub_4017B0();
                else
                    v5[i] = Str[i] - 29;
            }
            else
            {
                v5[i] = Str[i] - 87;
            }
        }
        else
        {
            v5[i] = Str[i] - 48;
        }
    }
    return sub_4017F0((int)v5);
}

```

查看sub_401770函数在sub_4017F0函数中被调用。

a1为v5 数组的首地址。 `*(_DWORD *)(a1 + 4 * v4)` 相当于是a1[v4]。因为这里是DWORD类型，也就是4个字节，每次循环+4，相当于是取数组的下一个元素。因此，while循环的条件是 `0 <= a1[v4] <= 62` 。

循环中，对Str1赋值： `Str1[v4] = abcdefghiabcde[a1[v4]]` 。

最后，可以看出，要能够弹出 `pass!` 的MessageBox，需要让Str1与 `KanXueCTF2019JustForhappy` 相等。

```

// a1为v5 数组的首地址
BOOL __cdecl sub_4017F0(int a1)
{
    BOOL result; // eax
    char Str1[28]; // [esp+D8h] [ebp-24h]
    int v3; // [esp+F4h] [ebp-8h]
    int v4; // [esp+F8h] [ebp-4h]

    v4 = 0;
    v3 = 0;
    while ( *(_DWORD *)(a1 + 4 * v4) < 62 && *(_DWORD *)(a1 + 4 * v4) >= 0 )// 0<= a1[v4] <= 62
    {
        Str1[v4] = aBcdefghiabcde[*(_DWORD *)(a1 + 4 * v4)];// abcdefghiABCDEFGHIJKLMNjklmn0123456789opqrstuvwxyzOP
        Qrstuvwxyz
        ++v4;
    }
    Str1[v4] = 0;
    if ( !strcmp(Str1, "KanXueCTF2019JustForhappy") )
        result = sub_401770();
    else
        result = sub_4017B0();
    return result;
}

```

写个python脚本求一下a1数组的内容:

```

src = "abcdefghijklmnopqrstuvwxyzOPQRSTUVWXYZ"
dst = "KanXueCTF2019JustForhappy"
a1 = []

for each in dst:
    a1.append(src.index(each))

print(a1)

```

得到a1的内容, 即v5的内容为: [19, 0, 27, 59, 44, 4, 11, 55, 14, 30, 28, 29, 37, 18, 44, 42, 43, 14, 38, 41, 7, 0, 39, 39, 48]

接下来, 再写个脚本, 就能求出Str, 也就是用户的输入了。

需要注意的是, 不能直接照着反汇编的代码写。比如v5[0]为19, 如果单纯按反汇编的代码逻辑来看的话, 最终会调用sub_4017B0()函数。

而实际上, 我们需要根据v5[i]的值判断我们应该进入那个if条件中。

```
v5 = a1
Str = []

for each in v5:
    tmp1 = each + 29
    tmp2 = each + 87
    tmp3 = each + 48

    if 65 <= tmp1 <= 90:
        Str.append(tmp1)
    elif 97 <= tmp2 <= 122:
        Str.append(tmp2)
    elif 48 <= tmp3 <= 57:
        Str.append(tmp3)

Str_str = ""
for each in Str:
    Str_str += chr(each)

print(f"flag{{{Str_str}}}")
```

最终得到flag: `flag{j0rXI4bTeustBiIGHeCF70DDM}`

3. 总结

1. 有窗口的这种题目，可以从字符串入手，找到对应的函数，然后再找一找跟用户输入（需要找到获取用户输入的函数）的联系。
2. 对于有一些函数，不知道啥作用的，可以用动态调试的方法，测试不同的输入，查看该函数的返回值。猜测函数的作用。