

【二进制】【WP】MOCTF逆向题解

转载

[weixin_30684743](#) 于 2019-09-12 20:54:00 发布 117 收藏

文章标签: [数据结构与算法](#)

原文链接: <http://www.cnblogs.com/yichen115/p/11515136.html>

版权

moctf 逆向第一题: SOEASY

这个是个 64 位的软件, OD 打不开, 只能用 IDA64 打开, 直接搜字符串 (shift+F12) 就可以看到

moctf 逆向第二题: 跳跳跳

这个题当初给了初学逆向的我很大的成就感, 当时就学了改指令爆破, 根本不会分析算法, 这就能做出一道题还是很舒服的

打开程序, 是个猜数游戏

```
D:\anquan\学破解\CTF逆向题\moctf\dump.exe
Hello! 欢迎来到德莱。。。额, 跑错片场了。= =。
欢迎来到MOCTF! 你会摇骰子么? 赢6次你就能获得flag!
输入你心中的数字: 5
我的数字是: 6
回答错误, 游戏结束!
```

载入 OD 搜索字符串

```

00401C1 push dump.0040110E  闭+
00401C2 push dump.00472114 Hello! 欢迎来到德莱。。。额，跑错片场了。==。
00401C4 push dump.0040110E  闭+
00401C5 push dump.004720D4 欢迎来到MOCTF! 你会摇骰子么? 赢6次你就能获得flag!
00401C6 push dump.004720BC 输入你心中的数字:
00401CA push dump.0040110E  闭+
00401CA push dump.004720AC 我的数字是:
00401CD push dump.0040110E  闭+
00401CE push dump.00472090 回答正确, 继续努力!
00401D0 push dump.0040110E  闭+
00401D0 push dump.00472074 回答错误, 游戏结束!
00401D4 push dump.004720BC 输入你心中的数字:
00401D7 push dump.0040110E  闭+
00401D8 push dump.004720AC 我的数字是:
00401DB push dump.0040110E  闭+
00401DB push dump.00472090 回答正确, 继续努力!
00401DD push dump.0040110E  闭+
00401DE push dump.00472074 回答错误, 游戏结束!
00401E1 push dump.004720BC 输入你心中的数字:
00401E4 push dump.0040110E  闭+
00401E5 push dump.004720AC 我的数字是:
00401E8 push dump.0040110E  闭+
00401E9 push dump.00472090 回答正确, 继续努力!
00401EB push dump.0040110E  闭+
00401EB push dump.00472074 回答错误, 游戏结束!
00401EE push dump.004720BC 输入你心中的数字:
00401F2 push dump.0040110E  闭+
00401F2 push dump.004720AC 我的数字是:
00401F6 push dump.0040110E  闭+
00401F6 push dump.00472090 回答正确, 继续努力!
00401F8 push dump.0040110E  闭+
00401F8 push dump.00472074 回答错误, 游戏结束!

```

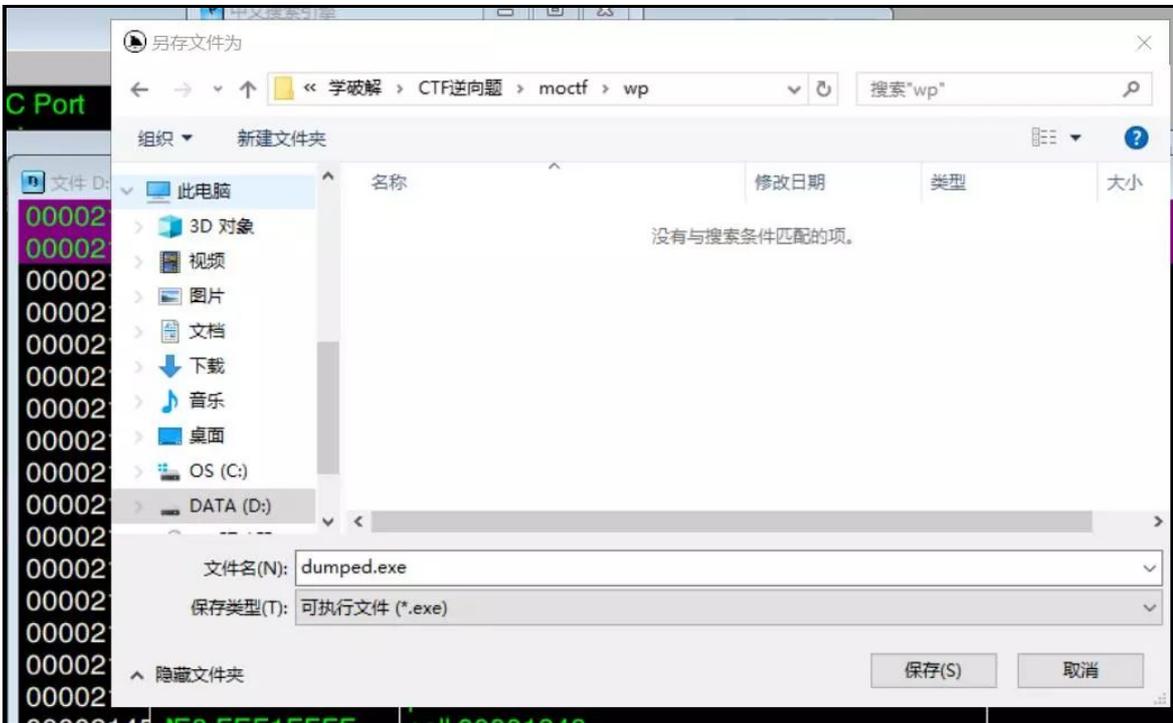
双击跟过去看看，很清晰了，那个 jnz 是关键，只要不让这个跳转实现就正确

00401CDA	3B45 F0	cmp eax,dword ptr ss:[ebp-0x10]	kernel32.75C68494
00401CDD	75 29	jnz short dump.00401D08	
00401CDF	68 0E114000	push dump.0040110E	闭+
00401CE4	68 90204700	push dump.00472090	回答正确, 继续努力!
00401CE9	68 98FE4700	push dump.0047FE98	
00401CEE	E8 55F6FFFF	call dump.00401348	
00401CF3	83C4 08	add esp,0x8	
00401CF6	8BC8	mov ecx, eax	
00401CF8	E8 92F5FFFF	call dump.0040128F	
00401CFD	8B4D EC	mov ecx,dword ptr ss:[ebp-0x14]	
00401D00	83C1 01	add ecx,0x1	
00401D03	894D EC	mov dword ptr ss:[ebp-0x14],ecx	dump.<ModuleEntryPoint>
00401D06	EB 39	jmp short dump.00401D41	
00401D08	68 0E114000	push dump.0040110E	闭+
00401D0D	68 74204700	push dump.00472074	回答错误, 游戏结束!
00401D12	68 98FE4700	push dump.0047FE98	
00401D17	E8 2CF6FFFF	call dump.00401348	
00401D1C	83C4 08	add esp,0x8	

那就直接把他 NOP 掉吧

3B45 F0	cmp eax,dword ptr ss:[ebp-0x10]	kernel132.75C68494
90	nop	
90	nop	
68 0E114000	push dump.0040110E	闭+
68 90204700	push dump.00472090	回答正确，继续努力！
68 98FE4700	push dump.0047FE98	
E8 55F6FFFF	call dump.00401348	
83C4 08	add esp,0x8	
8BC8	mov ecx,eax	
E8 92F5FFFF	call dump.0040128F	
8B4D EC	mov ecx,dword ptr ss:[ebp-0x14]	
83C1 01	add ecx,0x1	
894D EC	mov dword ptr ss:[ebp-0x14],ecx	dump.<ModuleEntryPoint>
EB 39	jmp short dump.00401D41	闭+
> 68 0E114000	push dump.0040110E	回答错误，游戏结束！
68 74204700	push dump.00472074	
68 98FE4700	push dump.0047FE98	
E8 2CF6FFFF	call dump.00401348	
83C4 08	add esp,0x8	
8BC8	mov ecx,eax	

保存出来



随便输入，最后还是成功出来 flag

```
Hello! 欢迎来到德莱。。。额，跑错片场了。= =。  
欢迎来到MOCTF! 你会摇骰子么? 赢6次你就能获得flag!  
输入你心中的数字: 5  
我的数字是: 1  
回答正确, 继续努力!  
输入你心中的数字: 3  
我的数字是: 2  
回答正确, 继续努力!  
输入你心中的数字: 1  
我的数字是: 2  
回答正确, 继续努力!  
输入你心中的数字: 4  
我的数字是: 6  
回答正确, 继续努力!  
输入你心中的数字: 5  
我的数字是: 6  
回答正确, 继续努力!  
输入你心中的数字: 2  
我的数字是: 4  
回答正确, 继续努力!  
哇呜~命运之子, 给你flag~  
ZHUtcF8xc19FYVN577yB  
10秒后自动退出
```

这个 flag 是 base64 加密的，之前在这里提交好多次总是不对，之后试了试才发现编码了

moctf 逆向第三题：暗恋的苦恼

当时可是费了好大劲才看懂这个伪代码的 (不知道现在还看不看得懂)，这次用 OD 分析的还算顺利，但是也花了一个多小时.....

暗恋的苦恼

150

小米暗恋着小果，可小华和小果总是眉来眼去，他们甚至还有自己秘密通信的渠道。一天，实在忍受不了的小米偷偷潜入小华的宿舍，拷贝下了他们通信的加密器和一串小果刚发来的字符串“QWDRILDWNTW”，还有桌面上的密匙：“ilovemoctf”。回到宿舍的小米懵逼了，因为他不懂 RE。所以，你可以帮助他拿到明文吗？提交形式：moctf{明文}。明文有确切的意思。

[题目链接](#)

先来看一下流程：

密文: QWDRILDWNTW

密钥: ilovemoctf

D:\anquan\学破解\CTF逆向题\moctf\jiamiqi.exe

请输入您的明文:

reverse

请输入您的密钥:

ilovemoctf

ZPKZVFS

载入 OD, 通过字符串找到获取输入的地方, 在下图箭头所指的地方程序运行起来等待输入

00401378	8945 F0	mov dword ptr ss:[ebp-0x10],eax	jiamiqi.00439920
0040137B	8B4D F0	mov ecx,dword ptr ss:[ebp-0x10]	
0040137E	894D F8	mov dword ptr ss:[ebp-0x8],ecx	jiamiqi.00439924
00401381	68 2D104000	push jiamiqi.0040102D	
00401386	68 34204300	push jiamiqi.00432034	请输入您的明文:
0040138B	68 20994300	push jiamiqi.00439920	ASCII "P C"
00401390	E8 15FDFFFF	call jiamiqi.004010AA	
00401395	83C4 08	add esp,0x8	
00401398	8BC8	mov ecx,ecx	jiamiqi.00439920
0040139A	E8 DEFCFFFF	call jiamiqi.0040107D	
0040139F	8B55 FC	mov edx,dword ptr ss:[ebp-0x4]	
004013A2	52	push edx	
004013A3	E8 78810000	call jiamiqi.00409520	
004013A8	83C4 04	add esp,0x4	
004013AB	68 2D104000	push jiamiqi.0040102D	
004013B0	68 20204300	push jiamiqi.00432020	请输入您的密钥:
004013B5	68 20994300	push jiamiqi.00439920	ASCII "P C"
004013BA	E8 EBFCFFFF	call jiamiqi.004010AA	
004013BF	83C4 08	add esp,0x8	

可以看到, 经过一个 CALL 之后出现了加密后的字符串, 那就跟进这个 CALL 看一下

地址	HEX 数据	反汇编	注释	寄存器 (FPU)
004013A3	E8 78810000	call jiamiqi.00409520		EAX 00921046 ASCII "ZPKZVFS"
004013A8	83C4 04	add esp,0x4		ECX 00000007
004013AB	68 2D104000	push jiamiqi.0040102D		EDX 00000007
004013B0	68 20204300	push jiamiqi.00432020	请输入您的密钥:	EBX 0035E000
004013B5	68 20994300	push jiamiqi.00439920	ASCII "P C"	ESP 0019FED8
004013BA	E8 EBFCFFFF	call jiamiqi.004010AA		EBP 0019FF40
004013BF	83C4 08	add esp,0x8		ESI 00409BE0 jiamiqi.<ModuleEntryPoint>
004013C2	8BC8	mov ecx,ecx		EIP 004013E2 jiamiqi.004013E2
004013C4	E8 B4FCFFFF	call jiamiqi.0040107D		C 0 ES 002B 32位 0(FFFFFFFF)
004013C9	8B45 F8	mov eax,dword ptr ss:[ebp-0x8]		P 1 CS 0023 32位 0(FFFFFFFF)
004013CC	50	push eax		A 0 SS 002B 32位 0(FFFFFFFF)
004013CD	E8 4E810000	call jiamiqi.00409520		Z 1 DS 002B 32位 0(FFFFFFFF)
004013D2	83C4 04	add esp,0x4		S 0 FS 0053 32位 361000(FFF)
004013D5	8B4D F8	mov ecx,dword ptr ss:[ebp-0x8]	密钥放到了ecx	T 0 GS 002B 32位 0(FFFFFFFF)
004013D8	51	push ecx		D 0
004013D9	8B55 FC	mov edx,dword ptr ss:[ebp-0x4]	明文放到了edx	0 0 LastErr ERROR_SUCCESS (00000000)
004013DC	52	push edx		EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
004013DD	E8 78FCFFFF	call jiamiqi.0040105A	加密call	ST0 empty 0.0
004013E2	83C4 08	add esp,0x8		ST1 empty 0.0
004013E5	8945 FC	mov dword ptr ss:[ebp-0x4],eax		ST2 empty 0.0
004013E8	8B45 FC	mov eax,dword ptr ss:[ebp-0x4]		ST3 empty 0.0
004013EB	50	push eax		ST4 empty 0.0
004013EC	68 1C204300	push jiamiqi.0043201C	%s\n	ST5 empty 0.0
004013F1	E8 A8800000	call jiamiqi.004094A0		

之后还得再进一个 CALL 才能发现程序算法 (程序是一个字一个字加密的刚开始会把输入的全部转换成大写)

004011B4	EB 4A	jmp short jiamiqi.00401200	
004011B6	> 0FBE45 08	movsx eax,byte ptr ss:[ebp+0x8]	放到eax
004011BA	83E8 41	sub eax,0x41	减去41
004011BD	8945 FC	mov [local.1],eax	放到local.1里面 (注意是16进制的)
004011C0	C745 F8 0000	mov [local.2],0x0	
004011C7	EB 09	jmp short jiamiqi.004011D2	
004011C9	> 8B4D F8	mov ecx,[local.2]	从零开始
004011CC	83C1 01	add ecx,0x1	加上一
004011CF	894D F8	mov [local.2],ecx	
004011D2	> 8B55 F8	mov edx,[local.2]	从零开始依次加一
004011D5	3B55 FC	cmp edx,[local.1]	与密文结果11相比
004011D8	7D 11	jge short jiamiqi.004011EB	
004011DA	8A45 0C	mov al,byte ptr ss:[ebp+0xC]	从第一个 I 密钥开始
004011DD	8845 0C	mov byte ptr ss:[ebp+0xC],al	
004011E0	8A4D 0C	mov cl,byte ptr ss:[ebp+0xC]	
004011E3	80C1 01	add cl,0x1	加上一
004011E6	884D 0C	mov byte ptr ss:[ebp+0xC],cl	
004011E9	EB DE	jmp short jiamiqi.004011C9	
004011EB	> 0FBE55 0C	movsx edx,byte ptr ss:[ebp+0xC]	
004011EE	83E8 50	sub edx,0x50	

程序算法:

1. 逐个取字符, 转成大写的, 然后用大写形式的 ASCII 码减去 41 (十六进制) 得到结果 (十六进制)
2. 然后用密钥的大写形式加上得到的结果

PS. 这里做了一些处理, 如果加上以后结果超出了大写字母 Z (0x5A), 那么会再减去 19 (十六进制)

004011E3	80C1 01	add cl,0x1	加上一
004011E6	884D 0C	mov byte ptr ss:[ebp+0xC],cl	
004011E9	EB DE	jmp short jiamiqi.004011C9	
004011EB	> 0FBE55 0C	movsx edx,byte ptr ss:[ebp+0xC]	
004011EF	83FA 5A	cmp edx,0x5A	
004011F2	7F 05	jg short jiamiqi.004011F9	
004011F4	8A45 0C	mov al,byte ptr ss:[ebp+0xC]	
004011F7	EB 07	jmp short jiamiqi.00401200	
004011F9	> 0FBE45 0C	movsx eax,byte ptr ss:[ebp+0xC]	
004011FD	83E8 19	sub eax,0x19	
00401200	> 5F	pop edi	
00401201	5E	pop esi	

然而还有个问题, 我发现给出的密文长度是比密钥长的, 那怎么办? 记得之前看的 writeup 说密钥循环使用的, 即: ilovemoctfilovemoctf, 这样一直排下去, 不管多出来多少密文都可以加密

为了验证这个问题, 我做了一个实验

这样输入的明文个数比密钥多一个, 而且多的是与第一个一样的, 我们看一下是不是按照那样循环是用的

```

mov [local.1], eax
mov [local.2], 0x0
jmp short jiamiqi.004011D3
mov ecx, [local.1]
add ecx, 0
mov [local.1], ecx
mov edx, [local.2]
cmp edx, [local.1]
jge short jiamiqi.004011D3
mov al, byte [local.1]
mov byte [local.2], al

```

加密结果让我很欣慰，证明我记性还不错 😊

```

call jiamiqi.00419590
xor eax, eax
pop edi
pop esi
pop ebx
add esp, 0x4
cmp ebp, esp
call jiamiqi.00419590
mov esp, ebp
pop ebp
retn

```

参考脚本，这或许会帮你更好的认识这个程序的流程

```

l="QWDRILDWNTW"
key="ILOVEMOCTFI"
flag=""
for i in range(len(l)):
    for temp1 in range(65,91):
        temp2=temp1-65
        if(temp2+ord(key[i])>90):
            temp2-=25
        if(temp2+ord(key[i])==ord(l[i])):
            flag+=chr(temp1)
print(flag)

```

moctf 逆向第四题：crackme1

打开后啥都不要输入，出现一串字符，说是 flag，但是每次打开都不一样

载入 OD，通过字符串找到

地址	反汇编	文本字符串
00401588	mov dword ptr ss:[ebp-0x4],CrackMe1.0046E024	2410488
00401611	push CrackMe1.0046E024	怎么一直在变???
0040161F	push CrackMe1.0046E01C	flag:
00402015	push CrackMe1.00447BE9	溉 G
00402E93	push CrackMe1.0046E044	missing locale facet
004032B8	mov dword ptr ds:[ecx],CrackMe1.0046E2F4	阅,
004032C4	push CrackMe1.0046E060	C
00403370	mov dword ptr ds:[eax],CrackMe1.0046E25C	轲"
004033B9	mov dword ptr ds:[edx],CrackMe1.0046E25C	轲"
00403D66	push CrackMe1.0046E068	%f

但是因为不知道该怎么分析，干脆直接段首下断，然后单步跟踪

地址	HEX 数据	反汇编	注释
00401570	> 55	push ebp	
00401571	. 8BEC	mov ebp,esp	
00401573	. 83EC 5C	sub esp,0x5C	
00401576	. 53	push ebx	
00401577	. 56	push esi	CrackMe1.<M
00401578	. 57	push edi	
00401579	. 8D7D A4	lea edi,dword ptr ss:[ebp-0x5C]	
0040157C	. B9 17000000	mov ecx,0x17	
00401581	. B8 CCCCCCCC	mov eax,0xCCCCCCCC	
00401586	. F3:AB	rep stos dword ptr es:[edi]	
00401588	. C745 FC 38E04	mov dword ptr ss:[ebp-0x4],CrackMe1.0046E024	2410488
0040158F	. C745 F8 00000	mov dword ptr ss:[ebp-0x8],0x0	
00401596	. C745 F4 00000	mov dword ptr ss:[ebp-0xC],0x0	
0040159D	. C745 F0 00000	mov dword ptr ss:[ebp-0x10],0x0	
004015A4	. 6A 1C	push 0x1C	
004015A6	. E8 D5F30100	call CrackMe1.00420980	

找到算法部分，算法：

1. 逐个取定义好的“2410488”
2. 乘以 2 后减去 60 (十六进制)
3. 除以 4 之后加上 3
4. 对 10 取余，得到结果

004015E0	8B4D FC	mov ecx,dword ptr ss:[ebp-0x4]	
004015E3	034D F0	add ecx,dword ptr ss:[ebp-0x10]	
004015E6	0FBE11	mouxs edx,byte ptr ds:[ecx]	逐个放到edx
004015E9	8D4412 A0	lea eax,dword ptr ds:[edx+edx-0x60]	两倍的edx-60放到eax
004015ED	99	cdq	
004015EE	83E2 03	and edx,0x3	
004015F1	03C2	add eax,edx	
004015F3	C1F8 02	sar eax,0x2	右移两位，相当于除以4
004015F6	83C0 03	add eax,0x3	再加上3
004015F9	99	cdq	
004015FA	B9 0A000000	mov ecx,0xA	
004015FF	F7F9	idiv ecx	除以A
00401601	8B45 F0	mov eax,dword ptr ss:[ebp-0x10]	
00401604	8B4D EC	mov ecx,dword ptr ss:[ebp-0x14]	
00401607	891481	mov dword ptr ds:[ecx*eax+1],edx	

在刚开始一直不知道这里的数是怎么出来的，在心里把 [edx+edx-0x60] 当作一个地址了，后来发现，这他妈的不是存放的刚赋值的字符嘛！！

004015D2	83C2 01	add edx,0x1	
004015D5	8955 F0	mov dword ptr ss:[ebp-0x10],edx	
004015D8	> 8B45 F0	mov eax,dword ptr ss:[ebp-0x10]	
004015DB	3B45 E8	cmp eax,dword ptr ss:[ebp-0x18]	
004015DE	7D 2C	jge short CrackMe1.0040160C	
004015E0	8B4D FC	mov ecx,dword ptr ss:[ebp-0x4]	
004015E3	034D F0	add ecx,dword ptr ss:[ebp-0x10]	
004015E6	0FBE11	mouxs edx ,byte ptr ds:[ecx]	逐个放到edx
004015E9	8D4412 A0	lea eax,dword ptr ds:[edx+edx-0x60]	两倍的edx-60放到eax
004015ED	99	cdq	
004015EE	83E2 03	and edx,0x3	
004015F1	03C2	add eax,edx	
004015F3	C1F8 02	sar eax,0x2	右移两位，相当于除以4
004015F6	83C0 03	add eax,0x3	再加上3
004015F9	99	cdq	
004015FA	B9 0A000000	mov ecx,0xA	
004015FF	F7F9	idiv ecx	除以A
00401601	8B45 F0	mov eax,dword ptr ss:[ebp-0x10]	

写出脚本，跑出 flag: 4533577

moctf 逆向第五题: crackme2

通过字符串定位获取输入的位置

地址	反汇编	文本字符串
004015D6	mov dword ptr ss:[ebp-0x4],CrackMe2.0046D04C	10<1<>;?8:%w!##&#q./,x((
00401605	push CrackMe2.0046D04C	请输入 flag:
00401690	push CrackMe2.0046D028	Your get the flag,but it
004016A8	push CrackMe2.0046D01C	Not it..
00402F43	push CrackMe2.0046D07C	missing locale facet
00403070	mov dword ptr ds:[ecx],CrackMe2.0046D37C	闲
004030D0	mov dword ptr ds:[eax],CrackMe2.0046D37C	闲
004031F0	mov dword ptr ds:[ecx],CrackMe2.0046D37C	闲
0040330E	mov dword ptr ds:[ecx],CrackMe2.0046D288	槌)
00403320	push CrackMe2.0046D098	C
00403A70	mov dword ptr ds:[eax],CrackMe2.0046D288	槌)
00404188	push CrackMe2.00401118	棕0
004048FA	mov dword ptr ds:[edx+ecx],CrackMe2.0046D04C	pY@

然后按步跟踪找到加密函数

0040164E	. 3B55 F0	cmp edx,dword ptr ss:[ebp-0x10]	
00401651	~ 7D 1B	jge short CrackMe2.0040166E	
00401653	. 8B45 F8	mov eax,dword ptr ss:[ebp-0x8]	
00401656	. 0345 EC	add eax,dword ptr ss:[ebp-0x14]	
00401659	. 8A08	mov cl,byte ptr ds:[eax]	
0040165B	. 51	push ecx	
0040165C	. E8 6CFAFFFF	call CrackMe2.004010CD	逐个加密
00401661	. 83C4 04	add esp,0x4	
00401664	. 8B55 F4	mov edx,dword ptr ss:[ebp-0xC]	
00401667	. 0355 EC	add edx,dword ptr ss:[ebp-0x14]	
0040166A	. 8802	mov byte ptr ds:[edx],al	
0040166C	^ EB D4	jmp short CrackMe2.00401642	
0040166E	> 8B45 F4	mov eax,dword ptr ss:[ebp-0xC]	

步入加密 CALL，分析算法：

1. 依次取输入的字符串
2. 从 6 开始，与输入的字符串进行异或
3. 异或结果与定义好的字符串进行比较

00401578	. 57	push edi	
00401579	. 8D7D C0	lea edi,[local.16]	
0040157C	. B9 10000000	mov ecx,0x10	
00401581	. B8 CCCCCCCC	mov eax,0xCCCCCCCC	
00401586	. F3:AB	rep stos dword ptr es:[edi]	
00401588	. 0FB E45 08	movsx eax,byte ptr ss:[ebp+0x8]	输入放到eax
0040158C	. 3305 C05D4700	xor eax,dword ptr ds:[0x475DC0]	从6开始进行异或结果
00401592	. 8845 08	mov byte ptr ss:[ebp+0x8],al	
00401595	. 8B0D C05D4700	mov ecx,dword ptr ds:[0x475DC0]	
0040159B	. 83C1 01	add ecx,0x1	用来异或的数加一
0040159E	. 890D C05D4700	mov dword ptr ds:[0x475DC0],ecx	
004015A4	. 8A45 08	mov al,byte ptr ss:[ebp+0x8]	
004015A7	. 5F	pop edi	
004015A8	. 5E	pop esi	
004015A9	. 5B	pop ebx	

红框圈出来的是定义好的字符串

0<1<>;?8:%w!##&#q./,x,(("

```

0040165B . 51          push ecx
0040165C . E8 6CFAFFFF call CrackMe2.004010CD
00401661 . 83C4 04     add esp,0x4
00401664 . 8B55 F4     mov edx,dword ptr ss:[ebp-0xC]
00401667 . 0355 EC     add edx,dword ptr ss:[ebp-0x14]
0040166A . 8802       mov byte ptr ds:[edx],al
0040166C . EB D4       jmp short CrackMe2.00401642
0040166E > 8B45 F4     mov eax,dword ptr ss:[ebp-0xC]
00401671 . 0345 EC     add eax,dword ptr ss:[ebp-0x14]
00401674 . C600 00     mov byte ptr ds:[eax],0x0
00401677 . 8B4D FC     mov ecx,dword ptr ss:[ebp-0x4]
0040167A . 51          push ecx
0040167B . 8B55 F4     mov edx,dword ptr ss:[ebp-0xC]
0040167E . 52          push edx
0040167F . E8 ACE50100 call CrackMe2.0041FC30
00401684 . 83C4 08     add esp,0x8
00401687 . 85C0       test eax,eax
00401689 . 75 20       jnz short CrackMe2.004016A8
堆栈 ss:[0019FF3C]=0046D05C (CrackMe2.0046D05C),ASCII 31,"0<1<>;?8:%w!##&#q./,x,(("
ecx=00000006
    
```

逐个加密

下面的那个 CALL 是进行对比的

```

0040166E > 8B45 F4     mov eax,dword ptr ss:[ebp-0xC]
00401671 . 0345 EC     add eax,dword ptr ss:[ebp-0x14]
00401674 . C600 00     mov byte ptr ds:[eax],0x0
00401677 . 8B4D FC     mov ecx,dword ptr ss:[ebp-0x4]
0040167A . 51          push ecx
0040167B . 8B55 F4     mov edx,dword ptr ss:[ebp-0xC]
0040167E . 52          push edx
0040167F . E8 ACE50100 call CrackMe2.0041FC30
00401684 . 83C4 08     add esp,0x8
00401687 . 85C0       test eax,eax
00401689 . 75 20       jnz short CrackMe2.004016A8
堆栈 ss:[0019FF34]=00AE0F60
edx=00000006
    
```

地址	HEX 数据	ASCII	地址	数值	注释
0019FF34	60 0F AE 00 C8 0E AE 00 5C D0 46 00 80 FF 19 00	???\u	0019FF2C	00000006	
0019FF44	A9 26 42 00 01 00 00 00 18 12 AE 00 90 12 AE 00	?	0019FF30	00000006	
0019FF54	C0 25 42 00 C0 25 42 00 00 00 34 00 00 00 00 00	?B.?B...	0019FF34	00AE0F60	
0019FF64	00 00 00 00 54 FF 19 00 00 00 00 00 CC FF 19 00	...T	0019FF38	00AE0EC8	ASCII "yichen"
0019FF74	2C E2 42 00 10 E1 46 00 00 00 00 00 94 FF 19 00	,舒\u	0019FF3C	0046D05C	ASCII 31,"0<1<>;?8:%w!##&#q./,x,(("
0019FF84	94 84 C6 75 00 00 34 00 70 84 C6 75 20 8C AA 38	\u	0019FF40	0019FF80	

可以看到我们输入的进行加密后是 7F，而规定的应该是 31

0041FC38	. F7C2 03000000	test edx,0x3
0041FC3E	.> 75 3C	jnz short CrackMe2.0041FC7C
0041FC40	> 8B02	mov eax,dword ptr ds:[edx]
0041FC42	. 3A01	cmp al,byte ptr ds:[ecx]
0041FC44	.> 75 2E	jnz short CrackMe2.0041FC74
0041FC46	. 0AC0	or al,al
0041FC48	.> 74 26	je short CrackMe2.0041FC70
0041FC4A	. 3A61 01	cmp ah,byte ptr ds:[ecx+0x1]
0041FC4D	.> 75 25	jnz short CrackMe2.0041FC74
0041FC4F	. 00E4	or ah,ah

ds:[0046D05C]=31 ('1')
al=7F

写出脚本跑出应该输入的字符串

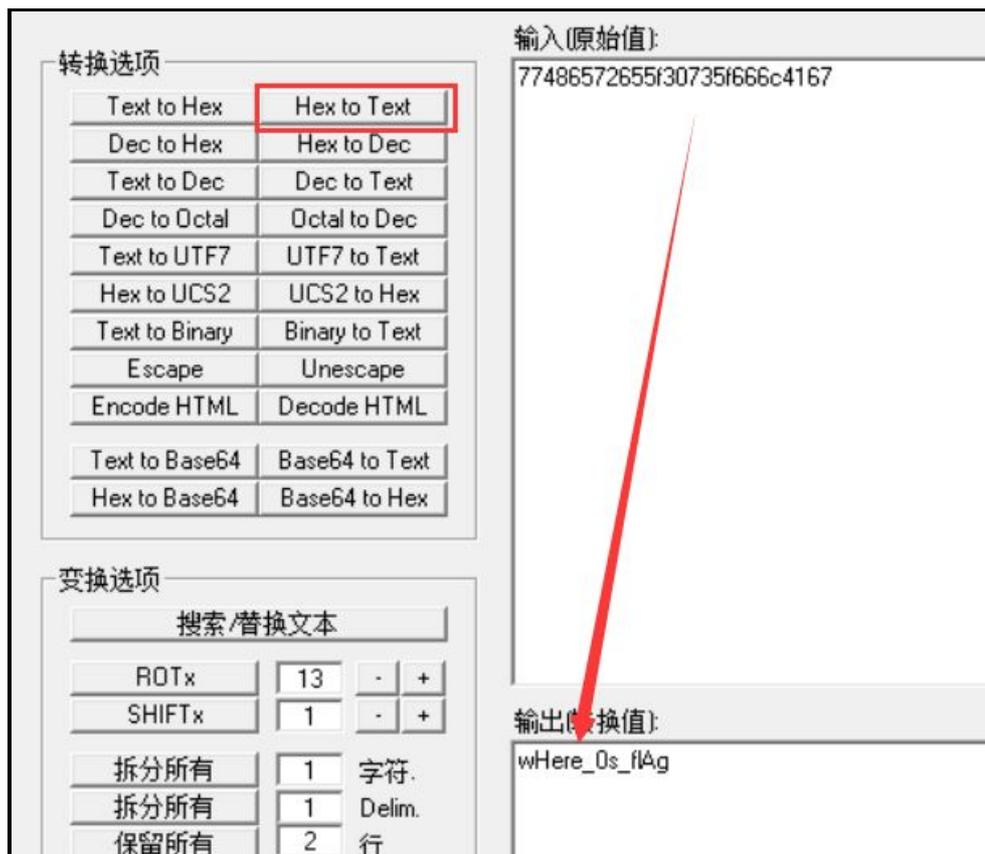
1	a = 6	77486572655
2	s = "10<1<>;?8:%w!##&#q./,x(,(("	77486572655f
3	flag = ""	77486572655f3
4		77486572655f30
5	for i in range(len(s)):	77486572655f307
6	result_temp = ord(s[i])^a	77486572655f3073
7	a += 1	77486572655f30735
8	flag += chr(result_temp)	77486572655f30735f
9	print(flag)	77486572655f30735f6
		77486572655f30735f66
		77486572655f30735f666c
		77486572655f30735f666c4
		77486572655f30735f666c41
		77486572655f30735f666c416
		77486572655f30735f666c4167

输入正确的后在进行测试，nice !!!

0041FC34	. 8B4C24 08	mov ecx,dword ptr ss:[esp+0x8]
0041FC38	. F7C2 03000000	test edx,0x3
0041FC3E	.> 75 3C	jnz short CrackMe2.0041FC7C
0041FC40	> 8B02	mov eax,dword ptr ds:[edx]
0041FC42	. 3A01	cmp al,byte ptr ds:[ecx]
0041FC44	.> 75 2E	jnz short CrackMe2.0041FC74
0041FC46	. 0AC0	or al,al
0041FC48	.> 74 26	je short CrackMe2.0041FC70
0041FC4A	. 3A61 01	cmp ah,byte ptr ds:[ecx+0x1]
0041FC4D	.> 75 25	jnz short CrackMe2.0041FC74
0041FC4F	. 0AE4	or ah,ah
0041FC51	.> 74 1D	je short CrackMe2.0041FC70
0041FC53	. C1E8 10	shr eax,0x10

ds:[0046D05C]=31 ('1')
al=31 ('1')

然而，这并不是 flag，需要将它转换成字符串形式



ps. 这是 N 个月前在公众号发的文章，转过来博客，主要是希望能够让更多人看见 😊

转载于:<https://www.cnblogs.com/yichen115/p/11515136.html>