

## A Message Box

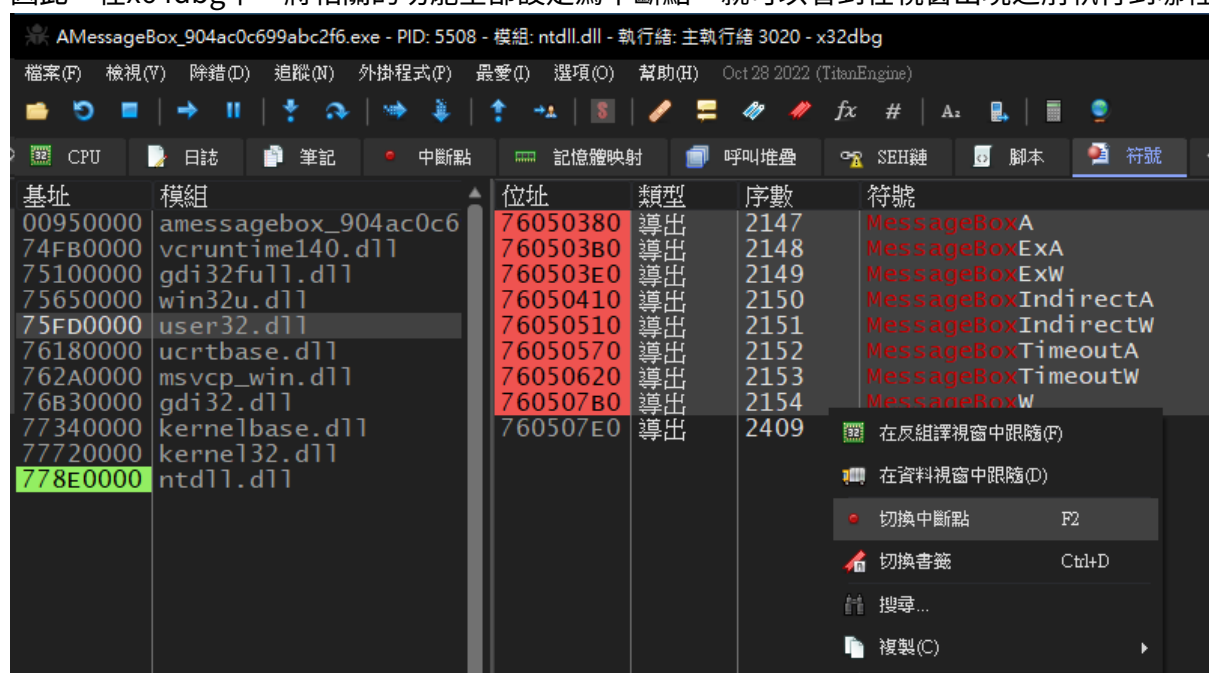
這題是windows逆向工程的題目，因此使用x64dbg來分析。

首先先試著執行程式，發現他會要求input flag，這邊先隨便給隨便給flag後跳出了一個提示視窗，說“wrong”

## Windows API

跳出來的視窗，其實是Windows的API，是user32.dll中提供的

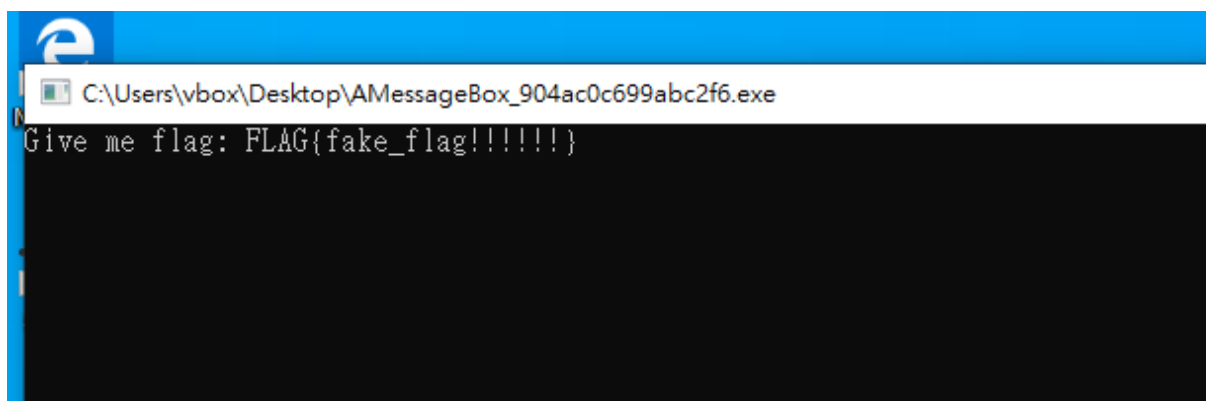
因此，在x64dbg中，將相關的功能全部設定為中斷點，就可以看到在視窗出現之前執行到哪裡。



## Run

設好中斷點後接下來就是試著執行看看了。

先執行到他可以輸入flag為止，並隨便輸入一個flag後按enter:



接下來遇到的中斷點，應該會有有用的資訊，因為是MessageBoxA的中斷點：

EIP	7605037B	CC	int3	
	7605037C	CC	int3	
	7605037D	CC	int3	
	7605037E	CC	int3	
	7605037F	CC	int3	
	76050380	8BFF	mov edi,edi	MessageBoxA
	76050382	55	push ebp	
	76050383	8BEC	mov ebp,esp	
	76050385	6A FF	push FFFFFFFF	
	76050387	6A 00	push 0	
	76050389	FF75 14	push dword ptr ss:[ebp+14]	
	7605038C	FF75 10	push dword ptr ss:[ebp+10]	
	7605038F	FF75 0C	push dword ptr ss:[ebp+C]	
	76050392	FF75 08	push dword ptr ss:[ebp+8]	
	76050395	E8 D6010000	call <user32.MessageBoxTimeoutA>	
	7605039A	5D	pop ebp	
	7605039B	C2 1000	ret 10	
	7605039E	CC	int3	
	7605039F	CC	int3	

找到了之後按x64dbg的call stack的tab，看看在執行到這邊之前是在哪裡：

執行緒ID	位址	返回到	來自	大小	Pa	註解
7872	0094F650	0095110B	76050380	18		user32.76050380
	0094F668	77968CC2	0095110B	24		amessagebox_904ac0c699abc2f6.0095110B
	0094F68C	77968C94	77968CC2	C8		ntdll.77968CC2
	0094F754	77954136	77968C94	510		ntdll.77968C94
	0094FC64	777360C9	77954136	10		ntdll.77954136
	0094FC74	77947A94	777360C9	5C		kernel32.777360C9
	0094FCD0	77947A64	77947A94	10		ntdll.77947A94
	0094FCE0	00000000	77947A64			ntdll.77947A64
5716	00DC5CFC	777360C9	7795396C	10		ntdll.7795396C

發現在MessageBoxA之前是在0095110B的地方，點兩下追蹤到該位置。

到0095110B之後，可以看到一些關鍵的程式邏輯：

009510BE	75 37	jne amessagebox_904ac0c699abc2f6.9510F7	
009510C0	33C0	xor eax,eax	
009510C2	85D2	test edx,edx	
009510C4	74 23	je amessagebox_904ac0c699abc2f6.9510E9	
009510C6	6666:0F1F8400 00000000	nop word ptr ds:[eax+eax],ax	eax+9533C0:"FLAG{fake_flag!!!!!!}"
009510D0	8A88 C0339500	mov cl,byte ptr ds:[eax+9533C0]	
009510D6	C0C1 03	rol cl,3	
009510D9	80F1 87	xor cl,87	
009510DC	3A88 18309500	cmp cl,byte ptr ds:[eax+953018]	
009510E2	75 13	jne amessagebox_904ac0c699abc2f6.9510F7	
009510E4	40	inc eax	
009510E5	3BC2	cmp eax,edx	
009510E7	72 E7	jb amessagebox_904ac0c699abc2f6.9510D0	
009510E9	6A 00	push 0	
009510EB	68 10219500	push amessagebox_904ac0c699abc2f6.952110	952110:"Result"
009510F0	68 18219500	push amessagebox_904ac0c699abc2f6.952118	952118:"Correct"
009510F5	EB 0C	jmp amessagebox_904ac0c699abc2f6.951103	
009510F7	6A 00	push 0	
009510F9	68 10219500	push amessagebox_904ac0c699abc2f6.952110	952110:"Result"
009510FE	68 20219500	push amessagebox_904ac0c699abc2f6.952120	952120:"Wrong"
00951103	6A 00	push 0	
00951105	FF15 34209500	call dword ptr ds:[<MessageBoxA>]	
0095110B	8B45 10	mov eax,dword ptr ss:[ebp+10]	
0095110E	8380 B8000000 02	add dword ptr ds:[eax+88],2	
00951115	33C0	xor eax,eax	
00951117	5D	pop ebp	
00951118	C3	ret	

最重要的就是這幾行：

```
mov cl, byte ptr ds:[eax+9533C0] ; "FLAG{fake_flag!!!!!!}"
rol cl, 3
xor cl, 87
cmp cl, byte ptr ds:[eax+953018]
jne ...
```

可以看的出來這邊就是在做比對了，因此只要找到資料位置，並還原出來就可以獲得flag了。額外補充的是，rol指令是rotate left的意思，也就是shift並將移出去的bits放到右邊。

## Encrypted flag

追蹤到存放加密過的flag的位置，可以看到：

00953017	00B5 E58DBD5C	add byte ptr ss:[ebp+5CBD8DE5],dh
0095301D	46	inc esi
0095301E	36:4E	dec esi
00953020	4E	dec esi
00953021	1E	push ds
00953022	0E	push cs
00953023	26:A4	movsb
00953025	1E	push ds
00953026	0E	push cs
00953027	4E	dec esi
00953028	46	inc esi
00953029	06	push es
0095302A	16	push ss
0095302B	AC	lodsb
0095302C	B4 3E	mov ah,3E
0095302E	4E	dec esi
0095302F	16	push ss
00953030	94	xchg esp,eax
00953031	3E:94	xchg esp,eax
00953033	8C948C 9C4EA48C	mov word ptr ss:[esp+ecx*4-735BB164],ss
0095303A	2E:46	inc esi
0095303C	8C6C00 00	mov word ptr ds:[eax+eax],qs

雖然他解析成了指令，不過沒關係，這些二進位數值就是加密過的flag了。

---

## Solve

依照上面的流程，寫出對應的python程式:

```
ror = lambda val, r_bits, max_bits: \
    ((val & (2**max_bits-1)) >> r_bits%max_bits) | \
    (val << (max_bits-(r_bits%max_bits)) & (2**max_bits-1))

a = [0xB5, 0xE5, 0x8D, 0xBD, 0x5C, 0x46, 0x36, 0x4E, 0x4E,
     0x1E, 0x0E, 0x26, 0xA4, 0x1E, 0x0E, 0x4E, 0x46, 0x06,
     0x16, 0xAC, 0xB4, 0x3E, 0x4E, 0x16, 0x94, 0x3E, 0x94, 0x8C,
     0x94, 0x8C, 0x9C, 0x4E, 0xA4, 0x8C, 0x2E, 0x46, 0x8C, 0x6C]
a = [i ^ 0x87 for i in a]
a = [ror(i, 0x3, 8) for i in a]
a = [chr(i) for i in a]

print("".join(a))
```

## Result

執行程式碼，可以得到結果:

```
$ python3 sol.py
FLAG{8699314d319802ef792b7babac9da58a}
```