

Computer Security HW0x2

R08946007 資科所 蔡昀達

Nickname : r08946007

1. 使用 ida pro & hex ray decompiler 先 decompile IDAmudamudamuda.exe 可以得到原始碼大概理解這隻程式的功能

```
1 int sub_13312F0()
2 {
3     int v1; // [esp+0h] [ebp-2Ch]
4     char v2; // [esp+4h] [ebp-28h]
5
6     sub_1331000();
7     puts("Oh, it's another ez Reversing Challenge");
8     puts("Leverage all you learned in class to solve this one");
9     puts("2 step to get the flag: \n\n\n");
10    puts("First, give me the seed: ");
11    scanf("%d", &v1);
12    sub_1331070(v1);
13    puts("\n\n\nOK then whats the flag ?");
14    scanf("%s", &v2);
15    if ( sub_1331270(&v2) )
16    {
17        puts("OH yeah, you got the flag\n\n\n");
18    }
19    else
20    {
21        puts("That's not right, go back to review the slides on http");
22        puts("Hint: file format, calling convention, shellcode, sta");
23    }
24    system("pause");
25    return 0;
26 }
```

2. 找到 seed 跟 flag 兩個涵式

Flag:

```
1 int __cdecl sub_1331270(const char *a1)
2 {
3     void *v2; // eax
4
5     if ( strlen(a1) != 32 )
6         return 0;
7     v2 = VirtualAlloc(0, 0xC8u, 0x1000u, 0x40u);
8     memcpy(v2, &unk_1334058, 0xC8u);
9     return ((int (__cdecl *)(const char *, void *))v2)(a1, &unk_1334018);
10 }
```

可以發現 flag 的涵式將 &4058 位置載入記憶體並做為涵式執行
&4058 記憶體位置內容為一段 shellcode
輸入的 flag 和 &4018 位置的內容做為參數傳進涵式

Seed:

```
4  int result; // eax
5  int v3; // [esp+4h] [ebp-28h]
6  char *v4; // [esp+8h] [ebp-24h]
7  int m; // [esp+14h] [ebp-18h]
8  int i; // [esp+18h] [ebp-14h]
9  signed int k; // [esp+1Ch] [ebp-10h]
10 int v8; // [esp+20h] [ebp-Ch]
11 int j; // [esp+24h] [ebp-8h]
12 int l; // [esp+24h] [ebp-8h]
13
14 v1 = (char *)dword_1334488(0);
15 dword_133448C = (int)v1;
16 result = *v1;
17 if ( result == 77 )
18 {
19     result = *(char *)(dword_133448C + 1);
20     if ( result == 90 )
21     {
22         v4 = (char *)((_DWORD *)dword_133448C + 60) + dword_133448C;
23         result = *v4;
24         if ( result == 80 )
25         {
26             result = v4[1];
27             if ( result == 69 )
28             {
29                 for ( i = (int)(v4 + 248); ; i += 40 )
30                 {
31                     v3 = strcmp((const char *)i, ".data");
32                     if ( v3 )
33                         v3 = -(v3 < 0) | 1;
34                     if ( !v3 )
35                         break;
36                 }
37             }
38         }
39     }
40 }
```

可以發現 seed 函式只有在最下面的地方使用到輸入的 seed 變數
觀察可以發現他在更改某一段記憶體
正好是 flag 函式中 shellcode 的位置
因此可以理解為 seed 改寫 &4058 的記憶體內容讓他成為一段 shellcode

3. 根據 x86 calling convention 合理猜測 shellcode 開頭為 push 結尾為 ret 來反推 seed，可以得到正確的 seed 為 16
4. 取的 seed 之後就得到了 shellcode 然後反組譯之後
可以發現這段組語將 &4018 的記憶體內容 add 23 然後 xor 66 就會是正確的 flag

00140000	55	push ebp
00140001	8BEC	mov ebp,esp
00140003	51	push ecx
00140004	C745 FC 00000000	mov dword ptr ss:[ebp-4],0
00140008	✓ EB 09	jmp 140016
0014000D	8B45 FC	mov eax,dword ptr ss:[ebp-4]
00140010	83C0 01	add eax,1
00140013	8945 FC	mov dword ptr ss:[ebp-4],eax
00140016	8B4D 0C	mov ecx,dword ptr ss:[ebp+C]
00140019	034D FC	add ecx,dword ptr ss:[ebp-4]
0014001C	0FBF11	movsx edx,byte ptr ds:[ecx]
0014001F	85D2	test edx,edx
00140021	✓ 74 25	je 140048
00140023	8B45 08	mov eax,dword ptr ss:[ebp+8]
00140026	0345 FC	add eax,dword ptr ss:[ebp-4]
00140029	0FBF08	movsx ecx,byte ptr ds:[eax]
0014002C	83C1 23	add ecx,23
0014002F	83F1 66	xor ecx,66
00140032	0FBF01	movsx edx,cl
00140035	8B45 0C	mov eax,dword ptr ss:[ebp+C]
00140038	0345 FC	add eax,dword ptr ss:[ebp-4]
0014003B	0FBF08	movsx ecx,byte ptr ds:[eax]
0014003E	3BD1	cmp edx,ecx
00140040	✓ 74 04	je 140046
00140042	33C0	xor eax,eax
00140044	✓ EB 07	jmp 14004D
00140046	^ EB C5	jmp 14000D
00140048	B8 01000000	mov eax,1
0014004D	8BE5	mov esp,ebp
0014004F	5D	pop ebp
00140050	C3	ret

5. 最終得到 flag : FLAG{y3s!!y3s!!y3s!!0h_my_g0d!!}