Anti_Baek

주어진 프로그램을 실행하면



다음과 같은 버튼이 나온다 'NO'를 누르면 프로그램이 종료되고 YES를 누르면



합격이라는 사진이 등장한다.

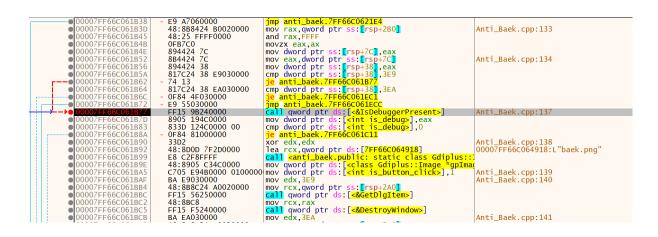
이제 디버거로 실행을 해서 YES를 누르면



영역을 전개한 백종원이 나온다. 우리는 영역에 당한것이다...

디버거로 실행하면 다른 사진이 등장하므로 안티 디버깅 기법이 있다는 것을 알 수 있다.

힌트로 주어진 IsDebuggerPresent 함수를 찾아보면



해당 위치에 존재하고 이 결과값(eax)을 1 → 0으로 바꿔주면 안티디버깅이 우회가 된다.

이후 그대로 실행을 해보면

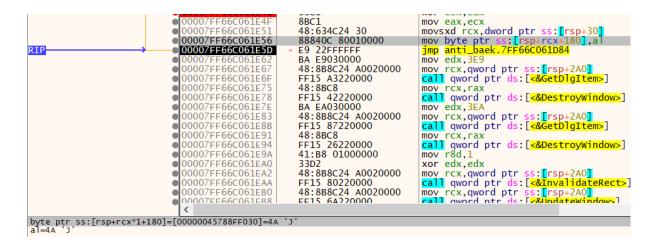
```
[ISP+AU]. SECKET
        007FF66C061DFA
007FF66C061E03
         007FF66C061E08
                                                                                                                                                                                                             [rsp+A0]: "SECRET"
             7FF66C061E19
7FF66C061E1B
7FF66C061E20
7FF66C061E28
7FF66C061E2A
                                                                                                  mov rcx,qword ptr ss:[rsp+A8]
div rcx
                                                 48:88C24 A800000
48:F7F1
48:8BC2
48:8DOD 72420000
0FBE0401
8B8C24 80000000
               FF66C061E35
FF66C061E38
FF66C061E3B
FF66C061E42
                                                                                                  mov rax,rdx
lea rcx,qword ptr ds:[<a href="mailto:kchar">ckhar *key>
movsx eax,byte ptr ds:[rcx+rax]
mov ecx,dword ptr ss:[rsp+80]
                                                                                                                                                                                                             00007FF66C0660B4: "SECRET"
                                                 33C8
                                                                                                  xor ecx, eax
                                                                                                  xor ecx, eax
mov eax, ecx
movsxd rcx, dword ptr ss:[rsp+30]
mov byte ptr ss:[rsp+rcx+180], al
jmp anti_baek./FF66C061D84
mov edx, 3E9
mov edx, 3E9
                                                 33C8
8BC1
48:634C24 30
88840C 80010000
E9 22FFFFF
BA E9030000
                                                                                                                                                                                                            Anti_Baek.cpp:153
Anti_Baek.cpp:154
                                                 48:8B8C24 A0020000
FF15 A3220000
48:8BC8
FF15 42220000
                                                                                                  mov rcx,qword ptr ss:[rsp+2A0]

call qword ptr ds:[x&GetDlgItem>]

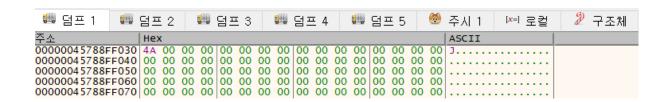
mov rcx,rax

call qword ptr ds:[x&DestroyWindow>]
00007FF66C061E7500007FF66C061E78
```

SECRET이라는 문자열과 xor을 하는 부분이 보이고



그 값을 특정 메모리에 저장을 한다.



이 메모리 위치를 보면 xor 결과인 J가 들어있으므로 반복 루틴이 끝나는 지점에 break point를 걸고

```
| 00007FF66C061E4D | 8B8C24 80000000 | mov ecx,dword ptr ss:[rsp+80] | 33C8 | xor ecx,eax | mov eax,ecx | mov eax,ecx | 48:634C24 30 | mov sxd rcx,dword ptr ss:[rsp+30] | mov byte ptr ss:[rsp+rcx+180],al | mov edx,3E9 | mov edx,4word ptr ss:[rsp+80] | mov ecx,dword ptr ss:[rsp+80] | mov ecx,dword ptr ss:[rsp+80] | mov edx,6Word ptr ss:[rsp+30] | mov edx,6
```

실행 결과인 메모리를 확인해보면 FLAG를 얻을 수 있다.

주소	Hex														ASCII	
00000045788FF030	4A 46	53	7B	44	30	6D	61	69	6E	5F	65	78	70	61	6E	JFS{D0main_expan
																5ion?_Just_@void
00000045788FF050	5F 69	74	21	7D	00	00	00	00	00	00	00	00	00	00	00	_it!}
00000045788FF060																

FLAG : JFS{D0main_expan5ion?_Just_@void_it!}