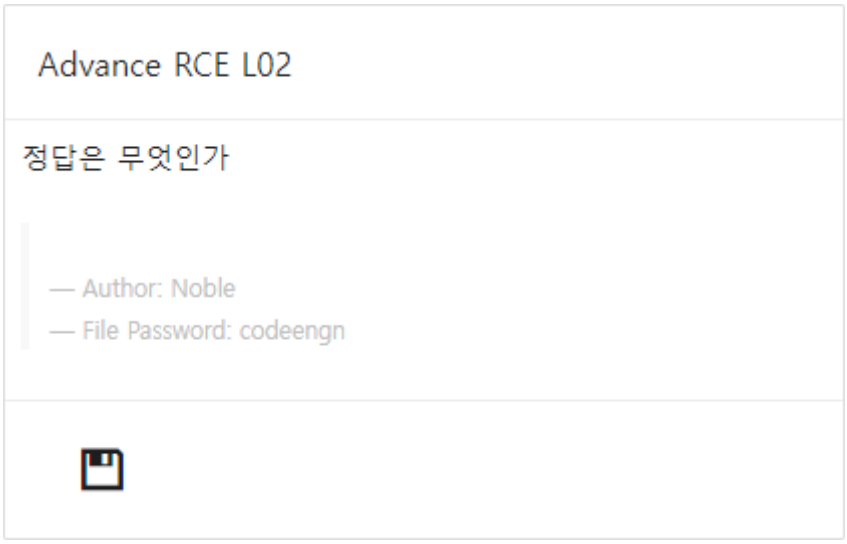
codeengn-advance-L02 풀이

리버싱 문제풀이 / Wonlf / 2022. 5. 3. 23:11



문제는 정답이 무엇인지를 물어보고 있다.

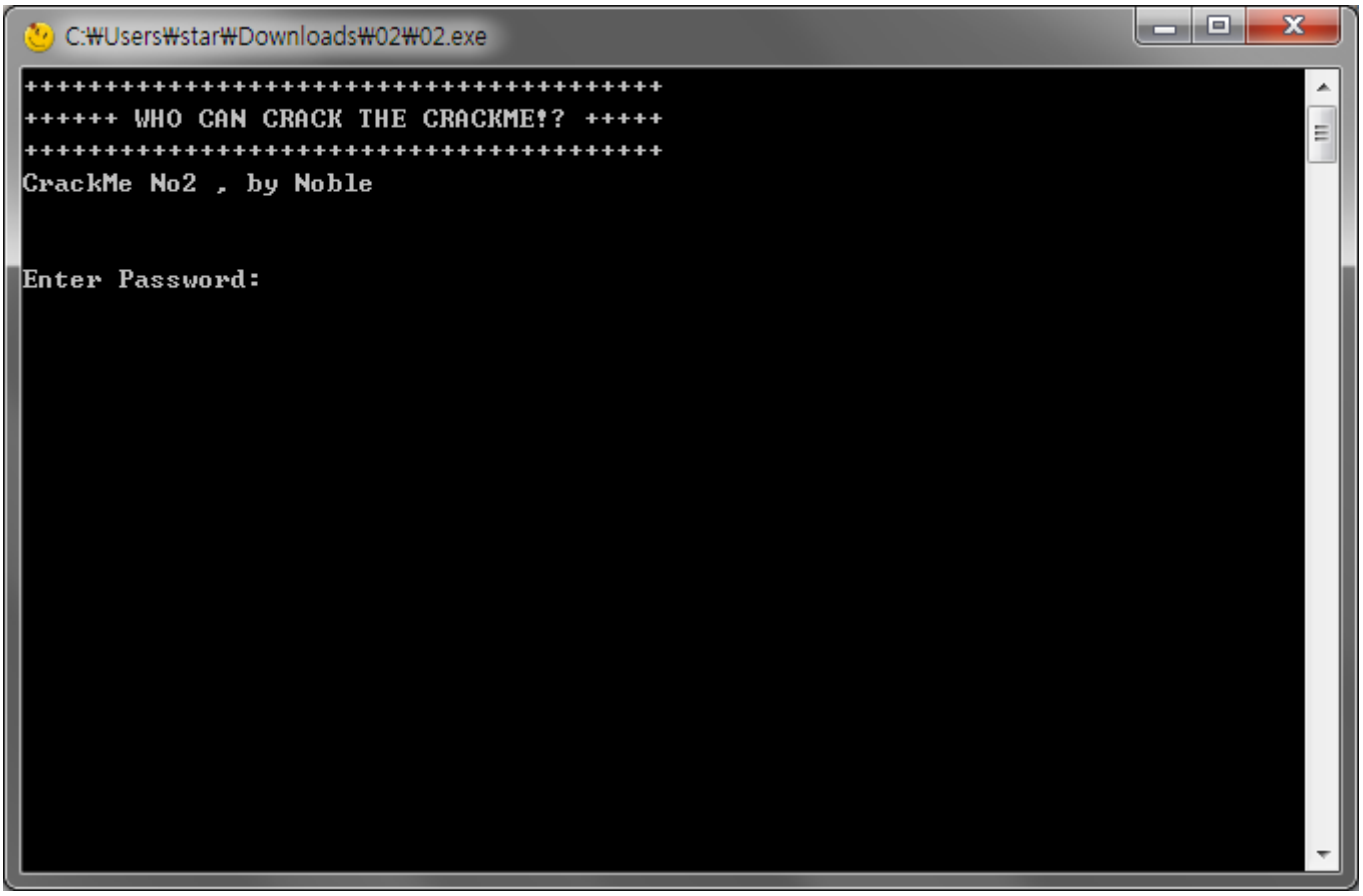
일단 Die로 열어본다.

특이사항은 보이지 않는다.

프로그램을 실행시켜본다.

윈도우 10에서는 작동하지 않는다.

윈도우 7가상머신을 구축해서 실행시켜본다.



이제야 잘 실행 된다.

문제가 원하는 정답은 이 비밀번호를 원하는 것 같다.

비밀번호를 입력하고 엔터를 누르면 프로그램이 종료 된다.

디버거로 열어보자.

```
004012B5  .  68 30 42 41 00      push 02.414230          414230:"Enter Password: "
004012BA  .  68 40 84 41 00      push 02.418440
004012BF  .  AA                  stosb
004012C0  .  E8 BB 0A 00 00      call 02.401D80
004012C5  .  8D 8C 24 F8 03 00 00 lea ecx,dword ptr ss:[esp+3F8]
```

문자열 찾기를 통해, 입력하는 부분을 찾았고 여기부터 실행시켜 보았다.

password를 입력 했을 때 프로그램이 종료되거나 성공하거나 둘 중 하나의 이벤트를 가지니,

종료되는 이벤트를 실행시키는 구문을 포커스해서 찾아보도록 하자.

```
004012B5  .  68 30 42 41 00      push 02.414230          414230:"Enter Password: "
004012BA  .  68 40 84 41 00      push 02.418440
004012BF  .  AA                  stosb
004012C0  .  E8 BB 0A 00 00      call 02.401D80
004012C5  .  8D 8C 24 F8 03 00 00 lea ecx,dword ptr ss:[esp+3F8]
004012CC  .  51                  push ecx
004012CD  .  68 D0 84 41 00      push 02.4184D0
004012D2  .  E8 39 0D 00 00      call 02.402010          여기가 실행되고 나면 입력창이 뜨는 함수
```

402010 함수가 실행되게 되면, 입력받는구문이 실행되게 된다.

함수 아래 구문에 브레이크포인트를 걸어주고,

```
004012D2  .  E8 39 0D 00 00      call 02.402010                          여기가 실행되고 나면 입력창이 뜨는 함수
004012D7  .  83 C4 10            add esp,10
004012DA  .  8D 94 24 88 07 00 00 lea edx,dword ptr ss:[esp+788]
004012E1  .  68 D0 14 40 00      push 02.4014D0
004012E6  .  68 40 14 40 00      push <02.sub_401440>
004012EB  .  6A 64               push 64
004012ED  .  6A 10               push 10
004012EF  .  52                  push edx
004012F0  .  E8 B0 46 00 00      call <02.sub_4059A5>
004012F5  .  C7 84 24 D0 0D 00 00 mov dword ptr ss:[esp+DD0],0
00401300  .  8D 9C 24 8C 07 00 00 lea ebx,dword ptr ss:[esp+78C]
00401307  .  C7 44 24 10 64 00 00 mov dword ptr ss:[esp+10],64            64:'d'
```

password를 입력한 뒤 한줄씩 실행시켜본다.

```
00401307  .  C7 44 24 10 64 00 00 mov dword ptr ss:[esp+10],64           64:'d'
0040130F  >  8D BC 24 F0 03 00 00 lea edi,dword ptr ss:[esp+3F0]
00401316  .  83 C9 FF             or ecx,FFFFFFFF
00401319  .  33 C0                xor eax,eax
0040131B  .  6A 01                push 1
0040131D  .  F2 AE                repne scasb
0040131F  .  F7 D1                not ecx
00401321  .  49                   dec ecx
00401322  .  8B E9                mov ebp,ecx
00401324  .  8D 4B FC             lea ecx,dword ptr ds:[ebx-4]
00401327  .  55                   push ebp
00401328  .  E8 C3 06 00 00       call <02.sub_4019F0>
0040132D  .  84 C0                test al,al
0040132F  .v 74 24                je 02.401355
00401331  .  8B 3B                mov edi,dword ptr ds:[ebx]              [ebx]:sub_40FFE1
00401333  .  8B CD                mov ecx,ebp
00401335  .  8B C1                mov eax,ecx
00401337  .  8D B4 24 F0 03 00 00 lea esi,dword ptr ss:[esp+3F0]
0040133E  .  C1 E9 02             shr ecx,2
00401341  .  F3 A5                rep movsd
00401343  .  8B C8                mov ecx,eax
00401345  .  83 E1 03             and ecx,3
00401348  .  F3 A4                rep movsb
0040134A  .  8B 03                mov eax,dword ptr ds:[ebx]              [ebx]:sub_40FFE1
0040134C  .  89 6B 04             mov dword ptr ds:[ebx+4],ebp
0040134F  .  03 E8                add ebp,eax
00401351  .  C6 45 00 00          mov byte ptr ss:[ebp],0
00401355  >  8B 44 24 10          mov eax,dword ptr ss:[esp+10]
00401359  .  83 C3 10             add ebx,10
0040135C  .  48                   dec eax
0040135D  .  89 44 24 10          mov dword ptr ss:[esp+10],eax
00401361  .^ 75 AC                jne 02.40130F
```

아래로 내려가다보면 for문이 있는데 특별한 구문은 아닌 듯 해 무시하고 더 내려가보면,

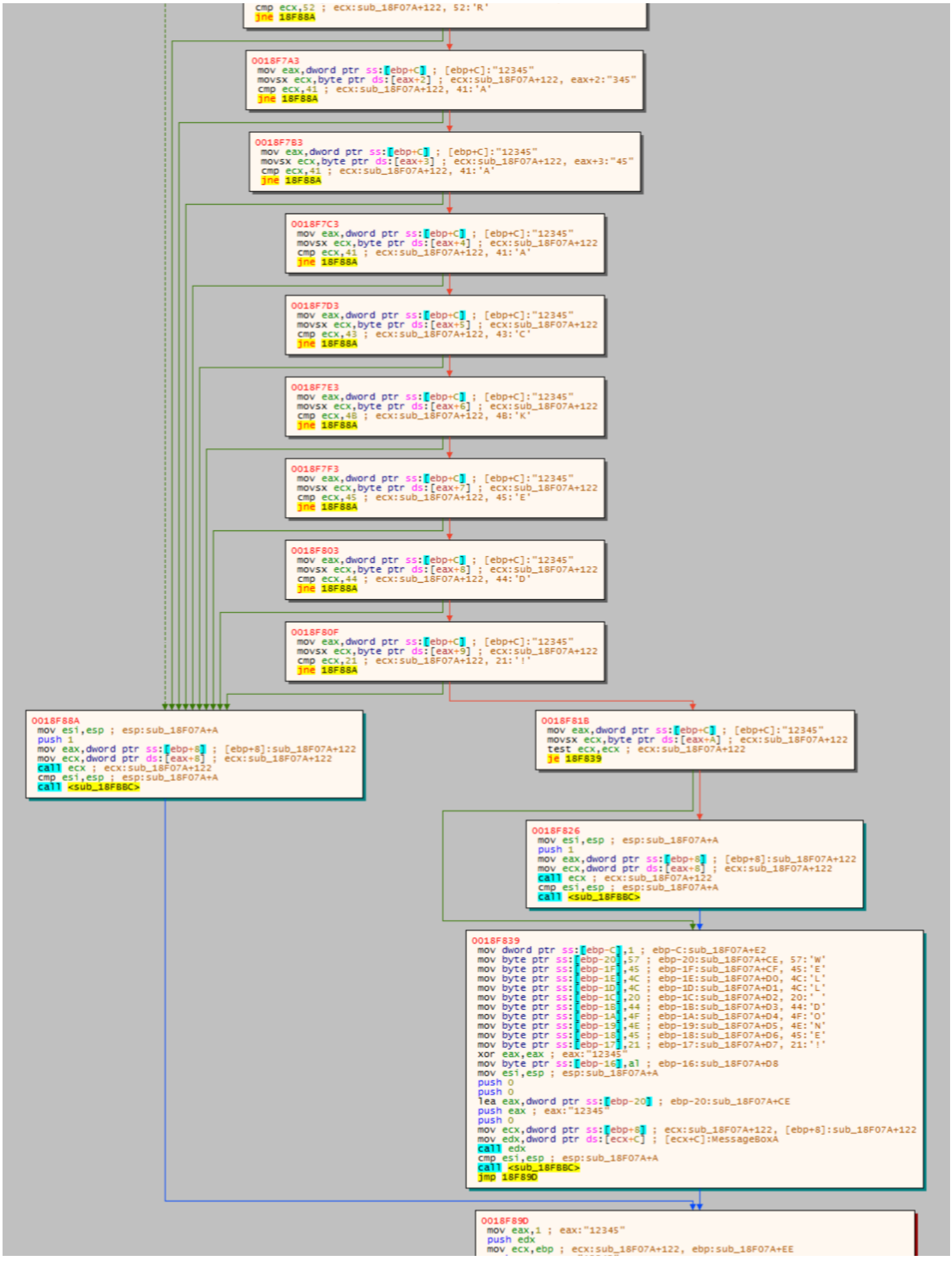
```
004013C5  .  FF D2                call edx
```

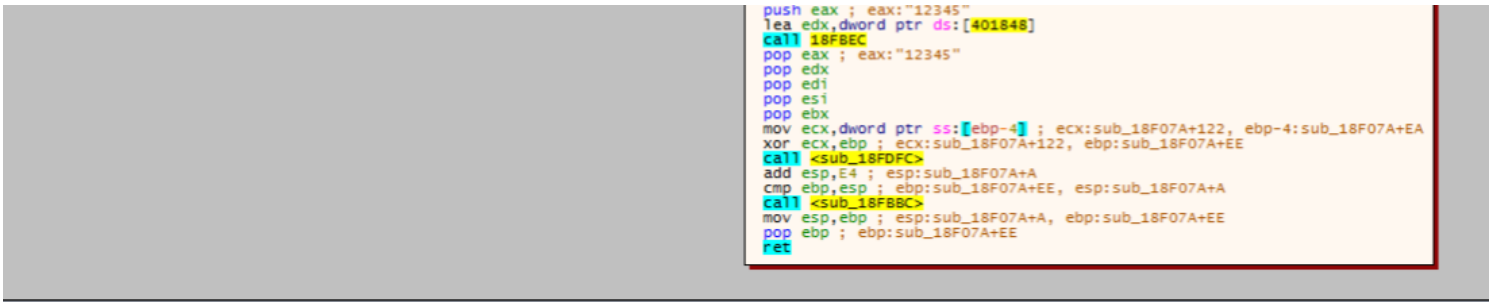
edx=0018F75C

edx안에 있는 주소의 함수를 실행시키고 종료된다. 이 함수 안을 보면

맵 | 호출 스택 | SEH | 스크립트 | 기호 | 소스 | 참조 | 스레드 | Snowman 디컴파일러 | 핸들 | Trace

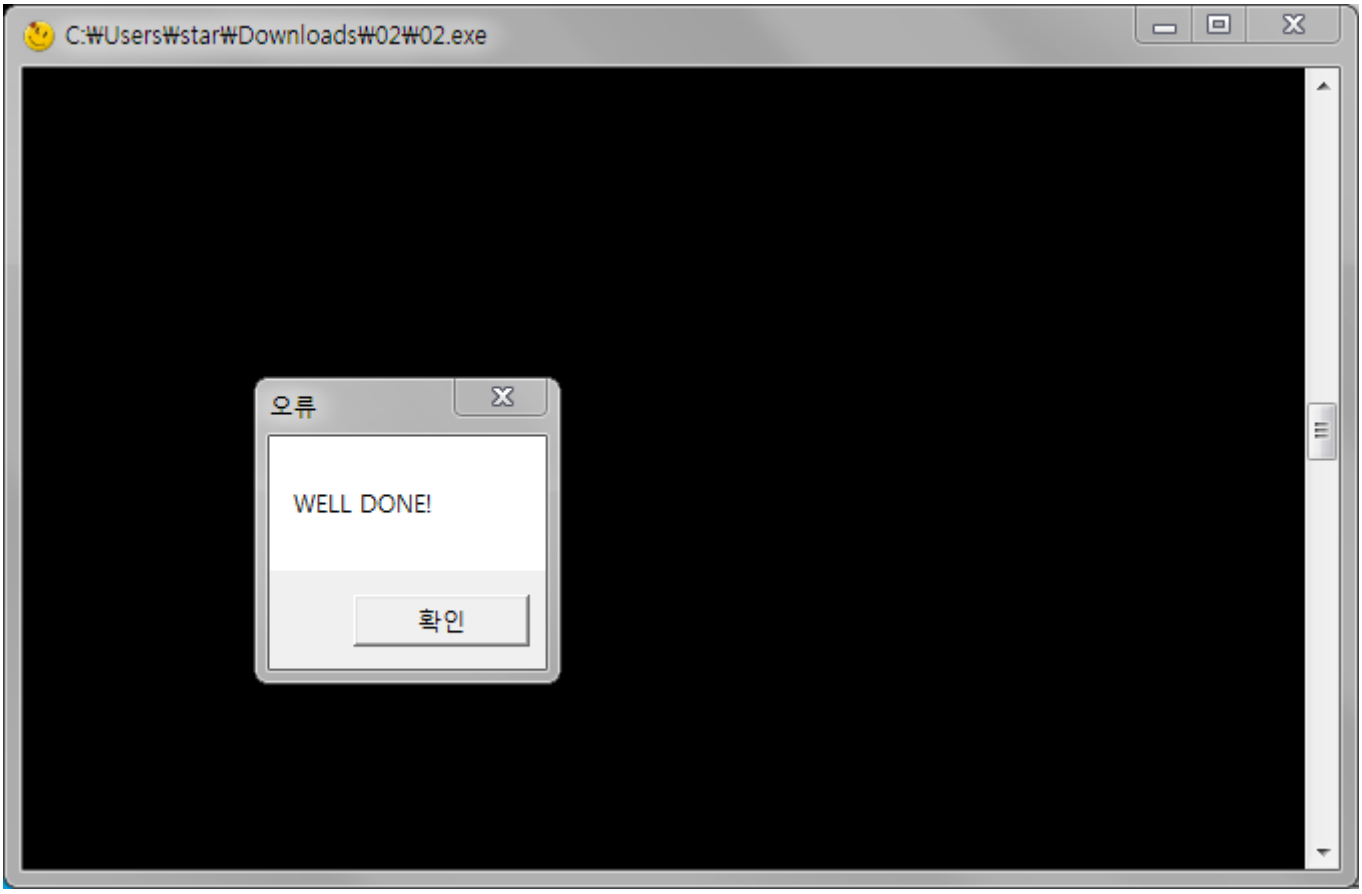
```
0018F75C
push ebp ; ebp:sub_18F07A+EE
mov ebp,esp ; ebp:sub_18F07A+EE, esp:sub_18F07A+A
sub esp,E4 ; esp:sub_18F07A+A
push ebx ; sub_18F765
push esi
push edi
lea edi,dword ptr ss:[ebp-E4] ; ebp-E4:sub_18F07A+A
mov ecx,39 ; ecx:sub_18F07A+122, 39:'9'
mov eax,CCCCCCCC ; eax:"12345"
rep stosd
mov eax,dword ptr ds:[406008] ; eax:"12345"
xor eax,ebp ; ebp:sub_18F07A+EE
mov dword ptr ss:[ebp-4],eax ; ebp-4:sub_18F07A+EA
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax] ; ecx:sub_18F07A+122, eax:"12345"
cmp ecx,43 ; ecx:sub_18F07A+122, 43:'C'
jne 18F88A

0018F793
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+1] ; ecx:sub_18F07A+122, eax+1:"2345"
```


cmp ecx,52 ; ecx:sub_18F07A+122, 52:'R'
jne 18F88A
0018F7A3
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+2] ; ecx:sub_18F07A+122, eax+2:"345"
cmp ecx,41 ; ecx:sub_18F07A+122, 41:'A'
jne 18F88A
0018F7B3
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+3] ; ecx:sub_18F07A+122, eax+3:"45"
cmp ecx,41 ; ecx:sub_18F07A+122, 41:'A'
jne 18F88A
0018F7C3
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+4] ; ecx:sub_18F07A+122
cmp ecx,41 ; ecx:sub_18F07A+122, 41:'A'
jne 18F88A
0018F7D3
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+5] ; ecx:sub_18F07A+122
cmp ecx,43 ; ecx:sub_18F07A+122, 43:'C'
jne 18F88A
0018F7E3
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+6] ; ecx:sub_18F07A+122
cmp ecx,4B ; ecx:sub_18F07A+122, 4B:'K'
jne 18F88A
0018F7F3
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+7] ; ecx:sub_18F07A+122
cmp ecx,45 ; ecx:sub_18F07A+122, 45:'E'
jne 18F88A
0018F803
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+8] ; ecx:sub_18F07A+122
cmp ecx,44 ; ecx:sub_18F07A+122, 44:'D'
jne 18F88A
0018F80F
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+9] ; ecx:sub_18F07A+122
cmp ecx,21 ; ecx:sub_18F07A+122, 21:'!'
jne 18F88A
0018F88A
mov esi,esp ; esp:sub_18F07A+A
push 1
mov eax,dword ptr ss:[ebp+8] ; [ebp+8]:sub_18F07A+122
mov ecx,dword ptr ds:[eax+8] ; ecx:sub_18F07A+122
call ecx ; ecx:sub_18F07A+122
cmp esi,esp ; esp:sub_18F07A+A
call <sub_18FBBC>
0018F81B
mov eax,dword ptr ss:[ebp+C] ; [ebp+C]:"12345"
movsx ecx,byte ptr ds:[eax+A] ; ecx:sub_18F07A+122
test ecx,ecx ; ecx:sub_18F07A+122
je 18F839
0018F826
mov esi,esp ; esp:sub_18F07A+A
push 1
mov eax,dword ptr ss:[ebp+8] ; [ebp+8]:sub_18F07A+122
mov ecx,dword ptr ds:[eax+8] ; ecx:sub_18F07A+122
call ecx ; ecx:sub_18F07A+122
cmp esi,esp ; esp:sub_18F07A+A
call <sub_18FBBC>
0018F839
mov dword ptr ss:[ebp-C],1 ; ebp-C:sub_18F07A+E2
mov byte ptr ss:[ebp-20],57 ; ebp-20:sub_18F07A+CE, 57:'W'
mov byte ptr ss:[ebp-1F],45 ; ebp-1F:sub_18F07A+CF, 45:'E'
mov byte ptr ss:[ebp-1E],4C ; ebp-1E:sub_18F07A+D0, 4C:'L'
mov byte ptr ss:[ebp-1D],4C ; ebp-1D:sub_18F07A+D1, 4C:'L'
mov byte ptr ss:[ebp-1C],20 ; ebp-1C:sub_18F07A+D2, 20:' '
mov byte ptr ss:[ebp-1B],44 ; ebp-1B:sub_18F07A+D3, 44:'D'
mov byte ptr ss:[ebp-1A],4F ; ebp-1A:sub_18F07A+D4, 4F:'O'
mov byte ptr ss:[ebp-19],4E ; ebp-19:sub_18F07A+D5, 4E:'N'
mov byte ptr ss:[ebp-18],45 ; ebp-18:sub_18F07A+D6, 45:'E'
mov byte ptr ss:[ebp-17],21 ; ebp-17:sub_18F07A+D7, 21:'!'
xor eax,eax ; eax:"12345"
mov byte ptr ss:[ebp-16],al ; ebp-16:sub_18F07A+D8
mov esi,esp ; esp:sub_18F07A+A
push 0
push 0
lea eax,dword ptr ss:[ebp-20] ; ebp-20:sub_18F07A+CE
push eax ; eax:"12345"
push 0
mov ecx,dword ptr ss:[ebp+8] ; ecx:sub_18F07A+122, [ebp+8]:sub_18F07A+122
mov edx,dword ptr ds:[ecx+C] ; [ecx+C]:MessageBoxA
call edx
cmp esi,esp ; esp:sub_18F07A+A
call <sub_18FBBC>
jmp 18F89D
0018F89D
mov eax,1 ; eax:"12345"
push edx
mov ecx,ebp ; ecx:sub_18F07A+122, ebp:sub_18F07A+EE



이미지가 커서 잘 안보이지만, 입력한 값의 한자리씩 특정 값과 비교하여 맞다면 다음 인자로 틀리다면 다른 분기로 가게 된다.

특정 문자를 조합해주면 "CRAAACKED!" 라는 문자열이 나오고 이것을 입력해주면 통과 된다.



성공!