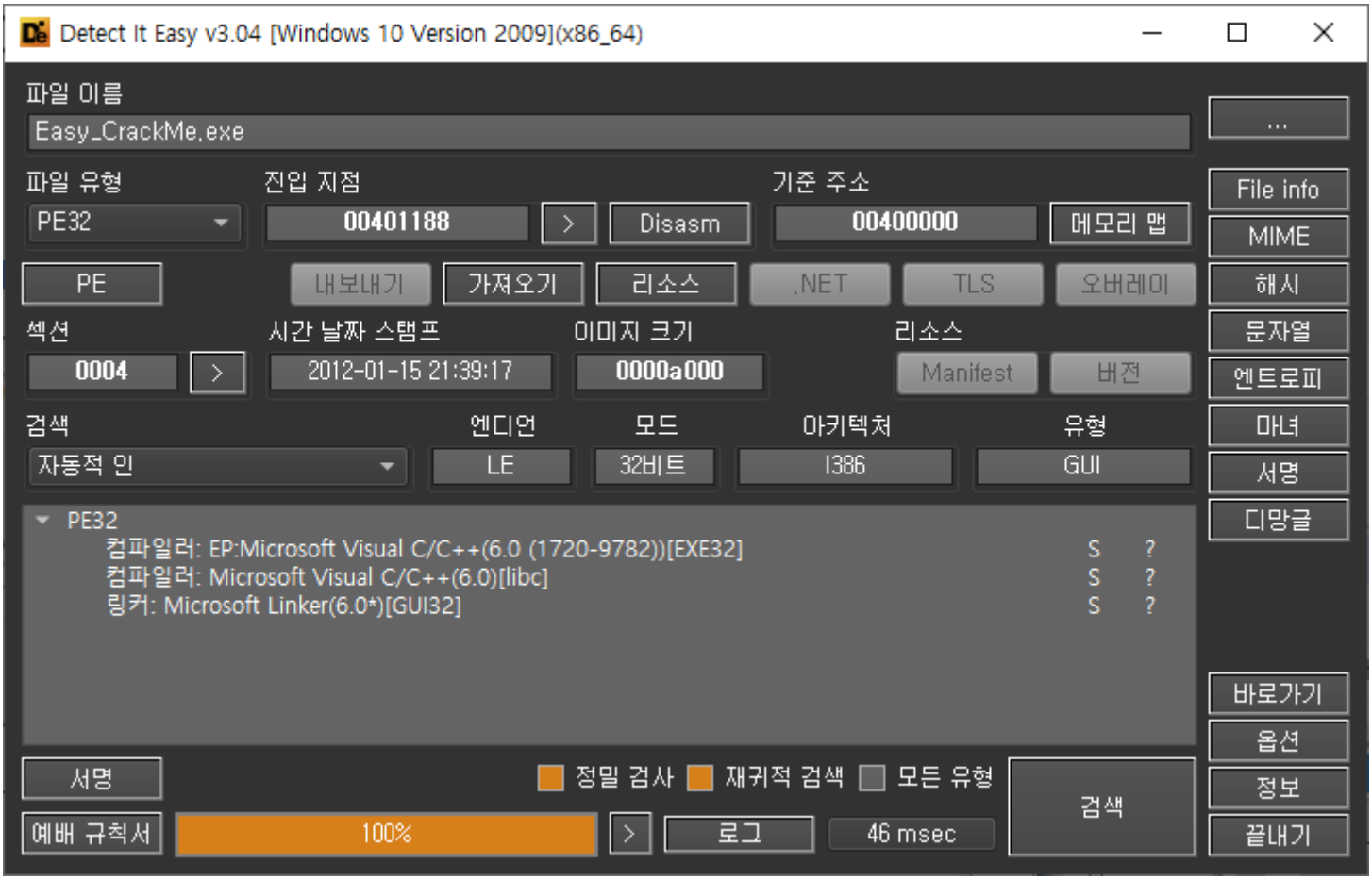
Reversing.kr Easy Crack 1 풀이

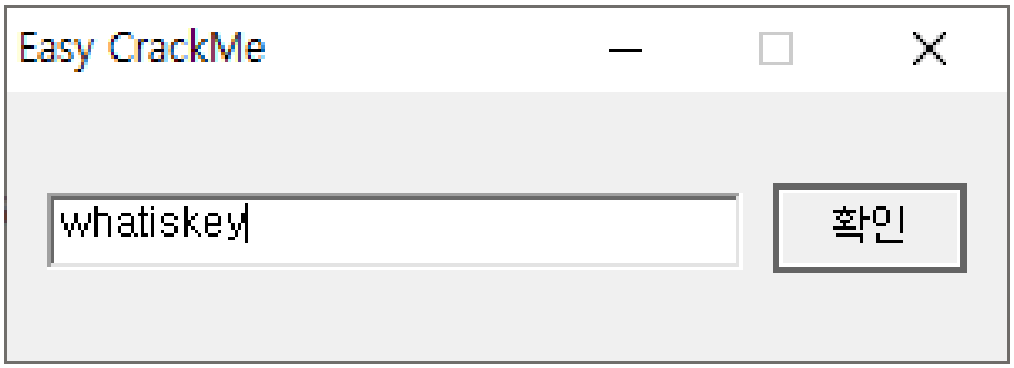
리버싱 문제풀이 / Wonlf / 2022. 3. 29. 23:54

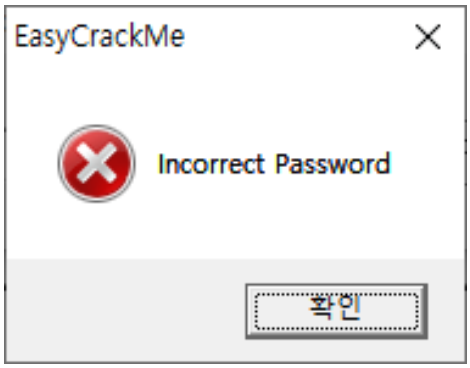
프로그램을 받아서 Die에 넣어보았다.



32비트에 따로 패킹은 되어있지 않은듯 하다.

실행 해보면....





키젠 문제 같다 키를 입력하니 틀렸다고 나온다 디버거로 열어보자.

```
>  33C0              xor eax,eax
.  EB 05             jmp easy_crackme.401107
>  1BC0              sbb eax,eax
.  83D8 FF           sbb eax,FFFFFFFF
>  5E                pop esi
.  5B                pop ebx
.  85C0              test eax,eax
.  75 28             jne easy_crackme.401135
.  807C24 04 45      cmp byte ptr ss:[esp+4],45          45:'E'
.  75 21             jne easy_crackme.401135
.  6A 40             push 40
.  68 58604000       push easy_crackme.406058            406058:"EasyCrackMe"
.  68 44604000       push easy_crackme.406044            406044:"Congratulation !!"
.  57                push edi
.  FF15 A0504000     call dword ptr ds:[<&MessageBoxA>]
.  6A 00             push 0
.  57                push edi
.  FF15 A4504000     call dword ptr ds:[<&EndDialog>]
.  5F                pop edi
.  83C4 64           add esp,64
.  C3                ret
>  6A 10             push 10
.  68 58604000       push easy_crackme.406058            406058:"EasyCrackMe"
.  68 30604000       push easy_crackme.406030            406030:"Incorrect Password"
.  57                push edi
.  FF15 A0504000     call dword ptr ds:[<&MessageBoxA>]
.  5F                pop edi
```

문자열로 검색하여 출력하는 듯한 어떠한 함수로 들어왔다. CTRL + A로 함수를 구분지어 주고 맨 위로 올라가 본다.

```
00401080  r$  83EC 64          sub esp,64                          sub_401080
00401083  .   57               push edi
00401084  .   B9 18000000      mov ecx,18
00401089  .   33C0             xor eax,eax
0040108B  .   8D7C24 05        lea edi,dword ptr ss:[esp+5]
0040108F  .   C64424 04 00     mov byte ptr ss:[esp+4],0
00401094  .   6A 64            push 64
00401096  .   F3:AB            rep stosd
00401098  .   66:AB            stosw
0040109A  .   AA               stosb
0040109B  .   8B7C24 70        mov edi,dword ptr ss:[esp+70]
0040109F  .   8D4424 08        lea eax,dword ptr ss:[esp+8]
004010A3  .   50               push eax
004010A4  .   68 E8030000      push 3E8
004010A9  .   57               push edi
004010AA  .   FF15 9C504000    call dword ptr ds:[<&GetDlgItemTextA>]
004010B0  .   807C24 05 61     cmp byte ptr ss:[esp+5],61          61:'a'
004010B5  .v  75 7E            jne easy_crackme.401135
004010B7  .   6A 02            push 2
004010B9  .   8D4C24 0A        lea ecx,dword ptr ss:[esp+A]
004010BD  .   68 78604000      push easy_crackme.406078            406078:"5y"
004010C2  .   51               push ecx
004010C3  .   E8 88000000      call <easy_crackme.sub_401150>
004010C8  .   83C4 0C          add esp,C
004010CB  .   85C0             test eax,eax
004010CD  .v  75 66            jne easy_crackme.401135
004010CF  .   53               push ebx
004010D0  .   56               push esi
004010D1  .   BE 6C604000      mov esi,easy_crackme.40606C         40606C:"R3versing"
004010D6  .   8D4424 10        lea eax,dword ptr ss:[esp+10]
004010DA  >   8A10             mov dl,byte ptr ds:[eax]
```

문자열을 계속 비교하는 구문이 보인다. 첫번째로 [esp+5] 와 61(a)을 비교하는데 같지 않으면 401135주소로 점프하는데 저 주소는 incorrect를 출력하는 구문이다. 그러므로 [esp+5] 와 61(a)이 같아야한다.

[esp+5]에는 뭐가 들어있는지 브레이크 포인트를 걸고 확인해보자.

```
ESP   0019F7EC
```

ESP에는 19F7EC가 들어있고 ESP+5이니 19F7F1에 있는 값은

```
0019F7EC   11 01 00 00 77 68 61 74 69 73 6B 65 79 00 00 00   ....whatiskey...
0019F7FC   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0019F80C   00 00 00 00 00 0  [0019F7F1] = 69746168 (사용자 데이터)  ...............
```

내가 입력한 키의 2번째 값이 들어있다.

이런식으로 계속 아래로 가며 확인해보면,

```
004010B5  .v 75 7E          jne easy_crackme.401135
004010B7  .  6A 02          push 2
004010B9  .  8D4C24 0A      lea ecx,dword ptr ss:[esp+A]
004010BD  .  68 78604000    push easy_crackme.406078          406078:"5y"
004010C2  .  51             push ecx
004010C3  .  E8 88000000    call <easy_crackme.sub_401150>
004010C8  .  83C4 0C        add esp,C
004010CB  .  85C0           test eax,eax
004010CD  .v 75 66          jne easy_crackme.401135
004010CF  .  53             push ebx
004010D0  .  56             push esi
004010D1  .  BE 6C604000    mov esi,easy_crackme.40606C       40606C:"R3versing"
004010D6  .  8D4424 10      lea eax,dword ptr ss:[esp+10]
004010DA  >  8A10           mov dl,byte ptr ds:[eax]
```

esp+A의 주소를 갖고 오는데, esp+A는 1개의 바이트만 가질 수 있지만 아래에 주석으로 5y라는 문자열이 뜨는 이유는,
C언어의 이것을 생각하면 된다.

```
char *s = "foobar";

s[3] == 'b';


char *x = &s[3]

x == "bar";
```

esp+A부터 해당하는 주소의 문자열 끝까지 참조하겠다는 뜻이다. 그렇기 때문에 a다음 문자열 중에 5y를 포함하는가? 라고 해석할 수 있다. 그 뒤도 똑같다.

정리해보면,

1. 내가 입력한 값의 2번째 위치에는 a가 들어가야한다.
2. a다음에는 5y가 들어가야한다.
3. 5y다음에는 R3versing이 들어가야한다.

까지 있다가, 마지막 쯤에

```
0040110D  .  807C24 04 45   cmp byte ptr ss:[esp+4],45       45:'E'
00401112  .v 75 21          jne easy_crackme.401135
00401114  .  6A 40          push 40
00401116  .  68 58604000    push easy_crackme.406058          406058:"EasyCrackMe"
0040111B  .  68 44604000    push easy_crackme.406044          406044:"Congratulation !!"
```

esp+5는 2번째 자리였으니 esp+4는 첫번째 자리가 되므로

맨 앞자리의 값과 45(E)를 비교하니 전부 합치면

"Ea5yR3versing"이 키가 되겠다.