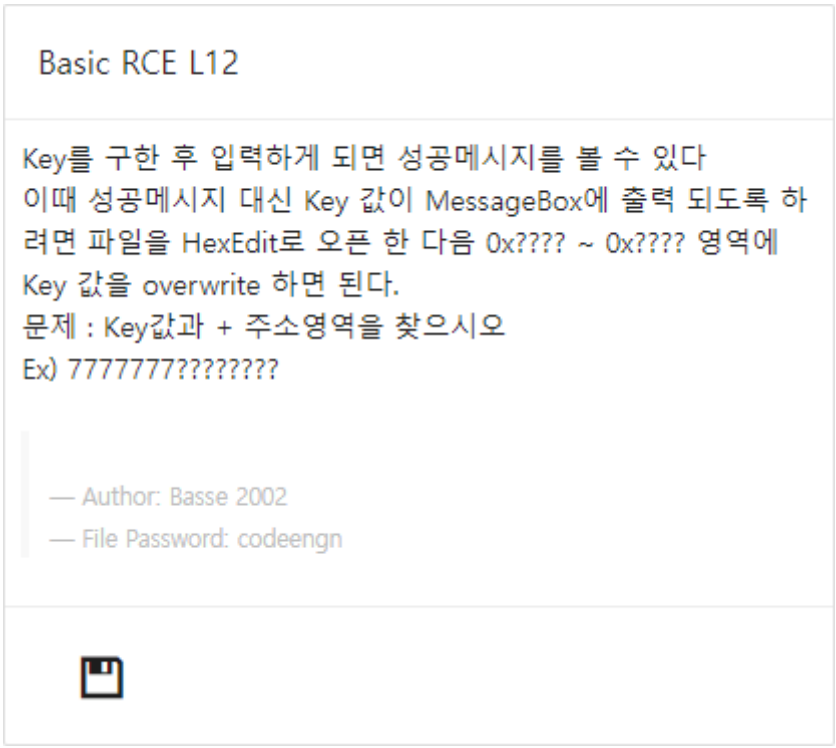
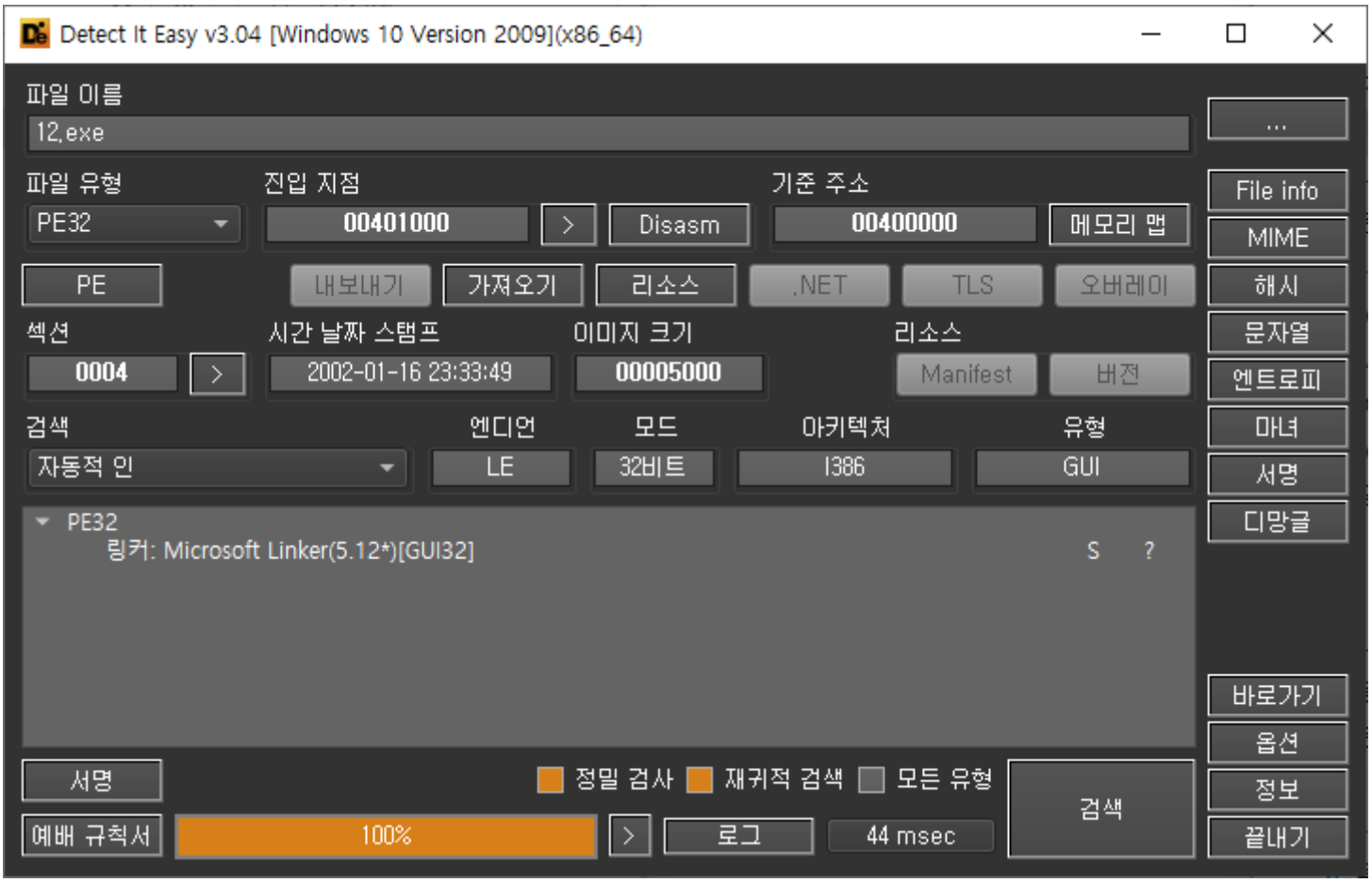
codeengn-basic-L12 풀이

리버싱 문제풀이 / Wonlf / 2022. 4. 5. 22:11



문제는 Key를 구하고, Key를 성공 문자열위치를 overwrite하라고 한다.



패킹이 되어 있지는 않다. 디버깅을 해서 Key를 구해보자.

```
00401029  [.  55              push ebp                                 sub_401029
0040102A  |.  8BEC            mov ebp,esp
0040102C  |.  8B45 0C         mov eax,dword ptr ss:[ebp+C]
0040102F  |.  3D 11010000     cmp eax,111
00401034  |.v 0F85 97000000   jne 12.4010D1
0040103A  |   8B55 10         mov edx,dword ptr ss:[ebp+10]
0040103D  |   C1EA 10         shr edx,10
00401040  |   66:0BD2         or dx,dx
00401043  |.v 0F85 B4000000   jne 12.4010FD
00401049  |   8B45 10         mov eax,dword ptr ss:[ebp+10]
0040104C  |   66:83F8 01      cmp ax,1
00401050  |.v 75 4A           jne 12.40109C
00401052  |   6A 00           push 0
00401054  |   6A 00           push 0
00401056  |   68 B90B0000     push BB9
0040105B  |   FF75 08         push dword ptr ss:[ebp+8]
0040105E  |   E8 31010000     call <JMP.&GetDlgItemInt>
00401063  |   BE 00304000     mov esi,12.403000                        403000:"Oqiqb4EhM/4jISMjlzQf6kpGQwLr
00401068  |>  833E 00         cmp dword ptr ds:[esi],0
0040106B  |.v 75 04           jne 12.401071
0040106D  |.v EB 0E           jmp 12.40107D
0040106F  |.v EB 0C           jmp 12.40107D
00401071  |>  8B1E            mov ebx,dword ptr ds:[esi]
00401073  |   E8 97000000     call <12.sub_40110F>
```

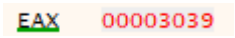
들어와서 문자열을 비교하는 함수까지는 들어 왔는데, 이상한 구문에서 자꾸 걸리길래 역방향으로 key를 찾아 보았다.

여기서 삽질 좀 했다. Key를 구하는데 아무 쓸모 없는 코드였다.

문제를 풀 때 너무 위에서 찾는 경향이 있음 너무 산으로 감 요점만 정확히.

flag를 뿜는 예제면 flag뿜는 구문부터 위로 올라가자.

여러가지 구문들이 있지만, 성공 문자열 위에 있는 CMP구문이 의심스럽다. eax에 뭐가 들어가는지 브레이크 포인트를 걸고 확인해본다.

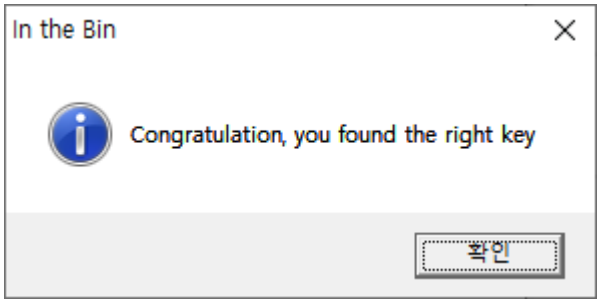


3039가 들어있다.

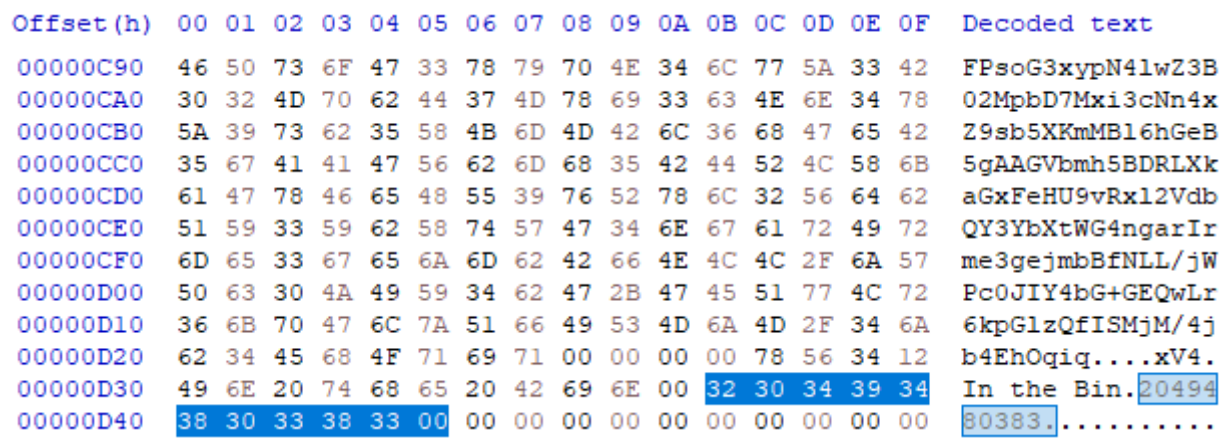
계산기로 확인해보니 내가 입력한 12345가 맞다.

그럼 비교하는 구문의 뒤 인자인 "0x7A2896BF"를 10진수로 바꿔보니

2049480383이 나왔다. 키로 입력해보면,



Key는 이렇게 찾게 되었고 HxD로 문자열 부분을 Overwrite해주겠다.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000C90	46	50	73	6F	47	33	78	79	70	4E	34	6C	77	5A	33	42	FPsoG3xypN4lwZ3B
00000CA0	30	32	4D	70	62	44	37	4D	78	69	33	63	4E	6E	34	78	02MpbD7Mxi3cNn4x
00000CB0	5A	39	73	62	35	58	4B	6D	4D	42	6C	36	68	47	65	42	Z9sb5XKmMBl6hGeB
00000CC0	35	67	41	41	47	56	62	6D	68	35	42	44	52	4C	58	6B	5gAAGVbmh5BDRLXk
00000CD0	61	47	78	46	65	48	55	39	76	52	78	6C	32	56	64	62	aGxFeHU9vRxl2Vdb
00000CE0	51	59	33	59	62	58	74	57	47	34	6E	67	61	72	49	72	QY3YbXtWG4ngarIr
00000CF0	6D	65	33	67	65	6A	6D	62	42	66	4E	4C	4C	2F	6A	57	me3gejmbBfNLL/jW
00000D00	50	63	30	4A	49	59	34	62	47	2B	47	45	51	77	4C	72	Pc0JIY4bG+GEQwLr
00000D10	36	6B	70	47	6C	7A	51	66	49	53	4D	6A	4D	2F	34	6A	6kpGlzQfISMjM/4j
00000D20	62	34	45	68	4F	71	69	71	00	00	00	00	78	56	34	12	b4EhOqiq....xV4.
00000D30	49	6E	20	74	68	65	20	42	69	6E	00	32	30	34	39	34	In the Bin.20494
00000D40	38	30	33	38	33	00	00	00	00	00	00	00	00	00	00	00	80383...........

이렇게 되면 파일 오프셋은 0x0D3B ~ 0x0D45이 되게 되는데, 0x0D44가 아닌 이유는마지막에는 꼭 널문자를 포함하여 계산 해야한다. 문자열의 끝은 NULL로 구분하기 때문에.

KEY : 2049480383

주소 영역 : 0D3B0D45

정답은 "**20494803830D3B0D45**"