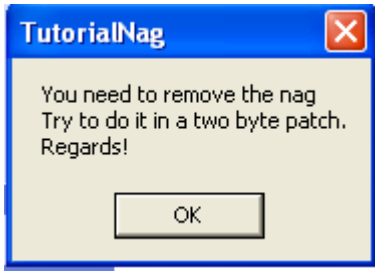
악성코드 분석 보고서

(sand-reversingwithlena-tutorials)

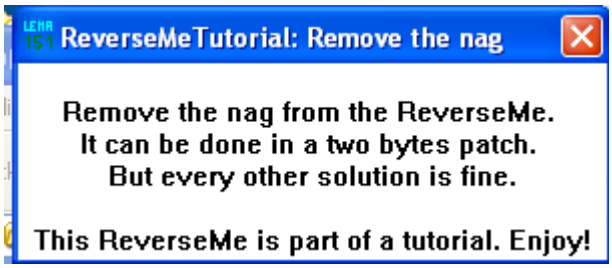
2025.07.29

18

1. 문제



첫 번째 창인 해당 창이 안나오게 삭제



두 번째 창은 삭제하면 안됨.

2. 해결 방법

```
00401288  $  6A 00          PUSH 0
0040128A     E8             DB E8
0040128B     EF             DB EF
0040128C     FF             DB FF
0040128D     FF             DB FF
0040128E     FF             DB FF
0040128F     A3             DB A3
00401290     30314000       DD ReverseM.00403130
00401294  .  BF 11104000    MOV EDI,ReverseM.00401011
00401299  .  E8 71000000    CALL ReverseM.0040130F
0040129E  >  E8 6EFDFFFF    CALL ReverseM.00401011
004012A3  .  33C0           XOR EAX,EAX
004012A5  .  50             PUSH EAX
004012A6  .  57             PUSH EDI
004012A7  .^ 7C F5          JL SHORT ReverseM.0040129E
004012A9  .^ 0F84 51FDFFFF  JE ReverseM.00401000
004012AF  .  6A 00          PUSH 0
004012B1  .  68 7D314000    PUSH ReverseM.0040317D
004012B6  .  68 34314000    PUSH ReverseM.00403134
004012BB  .  6A 00          PUSH 0
004012BD  .  E8 92FFFFFF    CALL <JMP.&USER32.MessageBoxA>
004012C2  .  50             PUSH EAX
004012C3  .  E8 B0FFFFFF    CALL <JMP.&KERNEL32.ExitProcess
004012C8  .  6A 00          PUSH 0
004012CA  .  6A 00          PUSH 0

Style = MB_OK|MB_APPLMODAL
Title = "TutorialNag"
Text = "You need to remove the nag▯▯Try to d
hOwner = NULL
MessageBoxA
ExitCode
ExitProcess
MsgFilterMax = 0
MsgFilterMin = 0
```

처음에 실행을 해보면 첫 번째 창에 나오는 구문이 있는 걸 볼 수 있는데 이게 나오는지 'F8'이용해서 실행해본다.

```
00401290     30314000       DD ReverseM.00403130
00401294  .  BF 11104000    MOV EDI,ReverseM.00401011
00401299  .  E8 71000000    CALL ReverseM.0040130F
0040129E  >  E8 6EFDFFFF    CALL ReverseM.00401011
004012A3  .  33C0           XOR EAX,EAX
004012A5  .  50             PUSH EAX
004012A6  .  57             PUSH EDI
004012A7  .^ 7C F5          JL SHORT ReverseM.0040129E
004012A9  .^ 0F84 51FDFFFF  JE ReverseM.00401000
```

0x0040129E에서 창이 발생하는 걸 볼 수 있다. 그럼 밑에 있는 메시지 박스는 속이기 위해 그냥 있는 걸로 볼 수 있다.

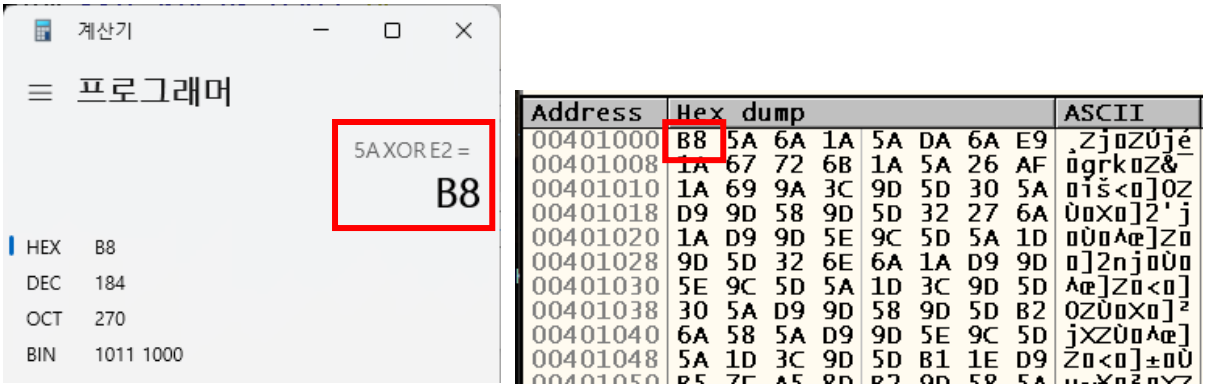
0x00401299, 0x0040129E에 브레이크를 걸고 0x00401299를 우선 'F7'을 이용하여 함수에 들어가 본다.

```
0040130F  $  B8 00104000    MOV EAX,ReverseM.00401000
00401314  >  8030 5A        XOR BYTE PTR DS:[EAX],5A
00401317  .  40             INC EAX
00401318  .  3D 18124000    CMP EAX,<JMP.&USER32.BeginPain
0040131D  .^ 7C F5          JL SHORT ReverseM.00401314
0040131F  .  C3             RETN
00401320  .  B8 00304000    MOV EAX,ReverseM.00403000
```

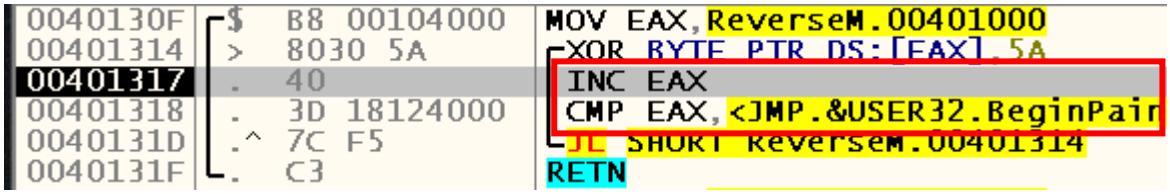
처음 0x00401000의 값이 EAX로 옮겨지는 걸 볼 수 있다.

Address	Hex dump	ASCII
00401000	E2 5A 6A 1A 5A DA 6A E9	âZj▯ZÚjé
00401008	1A 67 72 6B 1A 5A 26 AF	▯grk▯Z&
00401010	1A 69 9A 3C 9D 5D 30 5A	▯iš<▯]0Z
00401018	D9 9D 58 9D 5D 32 27 6A	Ù▯X▯]2'j
00401020	1A D9 9D 5E 9C 5D 5A 1D	▯Ù▯^œ]Z▯
00401028	9D 5D 32 6E 6A 1A D9 9D	▯]2njÙ▯

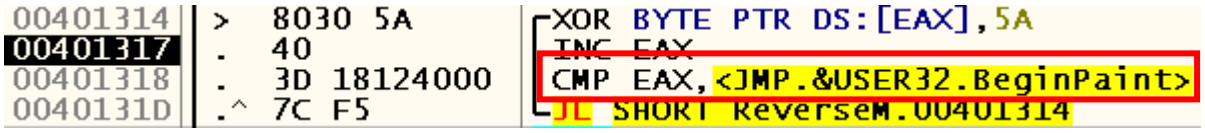
```
Registers (FP
EAX 00401000
ECX 0012FFB0
EDX 7C90E514
EBX 7FFDA000
ESP 0012FFC0
EBP 0012FFF0
ESI FFFFFFFF
EDI 00401011
```



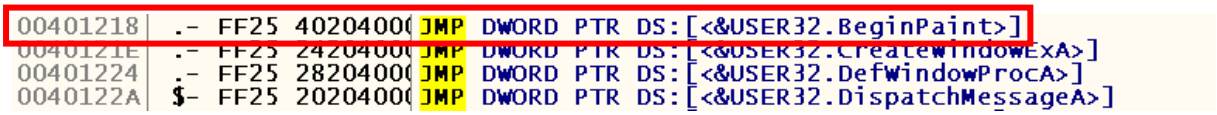
그리고 EAX 주소에 있는 값이 5A로 XOR되는 걸 볼 수 있는데 현재 0x00401000에는 E2가 있고 계산기로 XOR 해보면 B8이 나오는 걸 볼 수 있다. 덤프 창에서도 바뀌는 지 확인해보면 바뀌는 걸 볼 수 있다.



INC EAX는 EAX가 1을 증가하게 만들어줄거고 CMP EAX, <JMP.&USER32...는 <JMP.&USER32... 이 값을 EAX와 비교하는 걸 볼 수 있다.



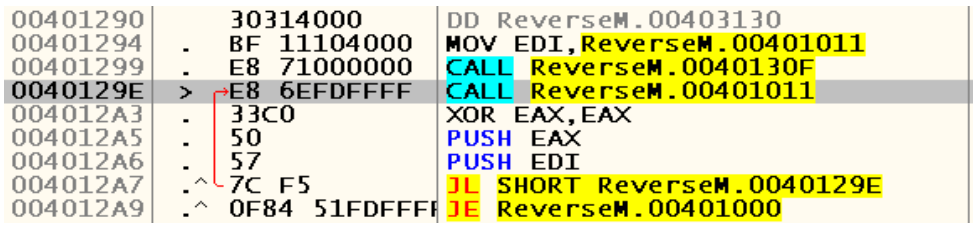
BeginPaint가 어디 있는지 찾아본다.



위쪽에 있는 걸 볼 수 있고 그럼 이 주소 값인 0x00401218과 비교하는 걸 볼 수 있다.

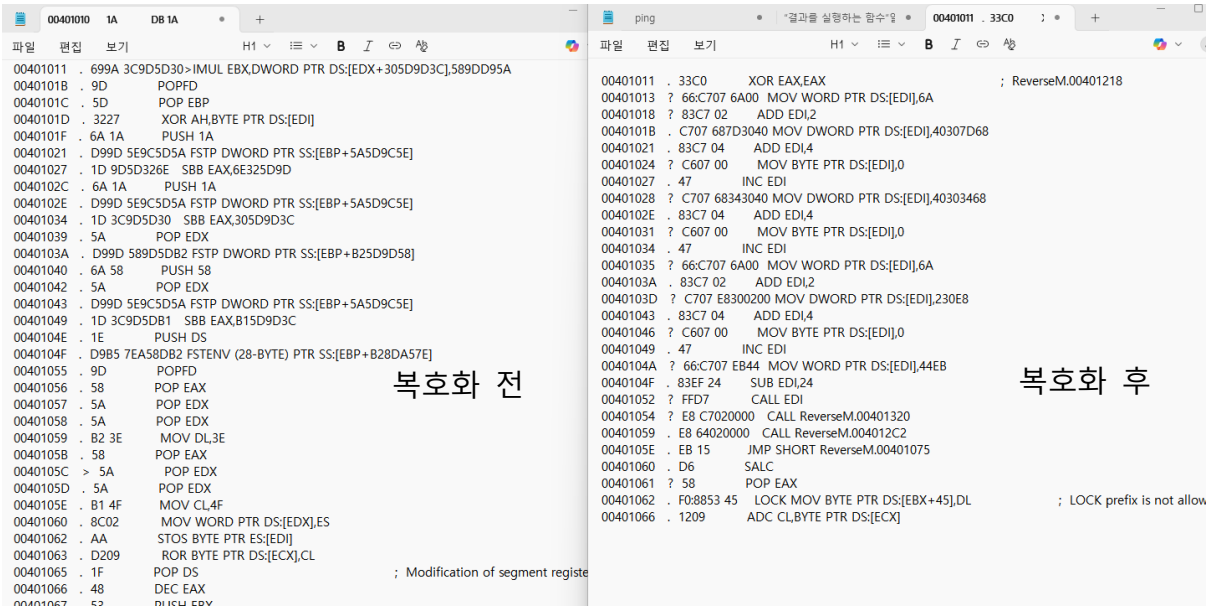
그럼 EAX가 더 커질 때까지 비교하는 루프문인 걸 볼 수 있고 즉, 401000 – 401218까지의 값을 5A와 XOR해주는 것을 볼 수 있다.

이거는 복호화하는 과정이라고 생각할 수 있다.

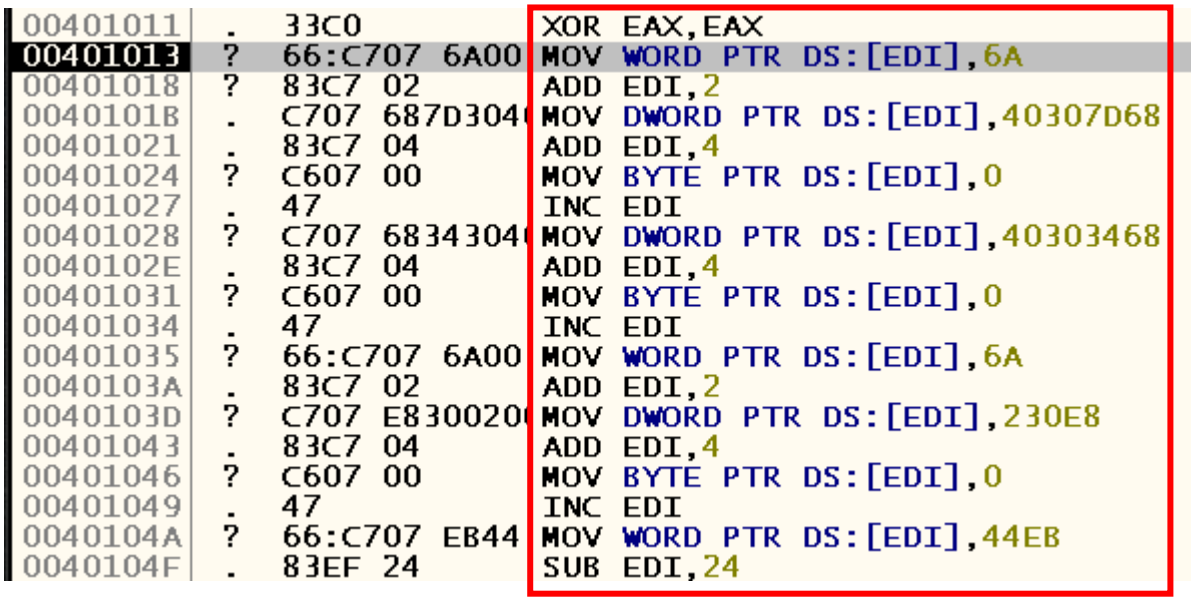


0x0040129E로 가서 호출하는 함수로 들어가보면 0x00401011을 호출하는 걸 볼 수 있는

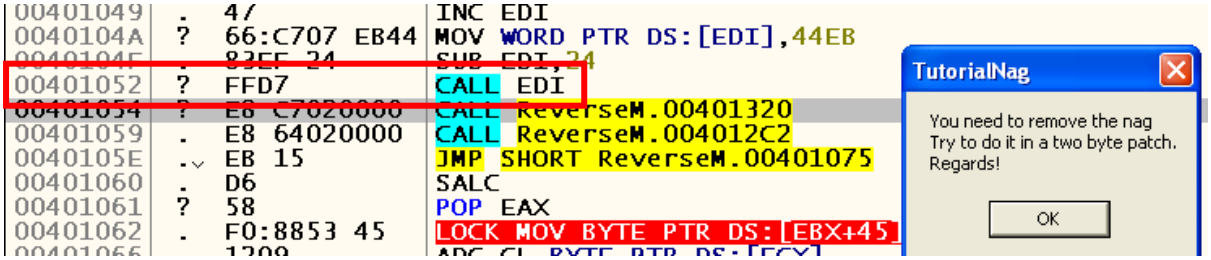
데 아까 복호화(401000 – 401218)한 부분이라고 볼 수 있다.



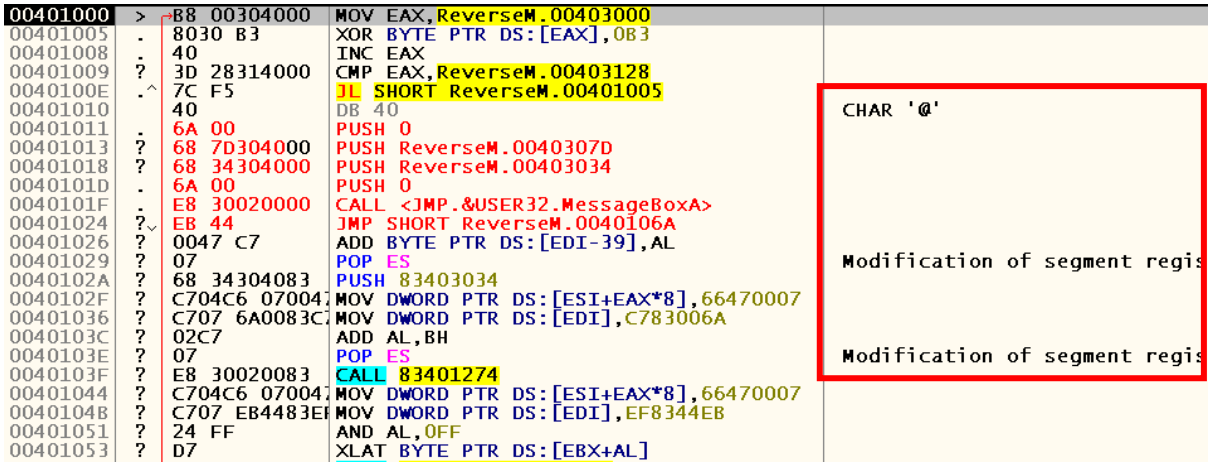
복호화 하기 전 부분하고 복호화 하고 그 코드 부분을 비교해 보면 달라진 걸 볼 수 있다.

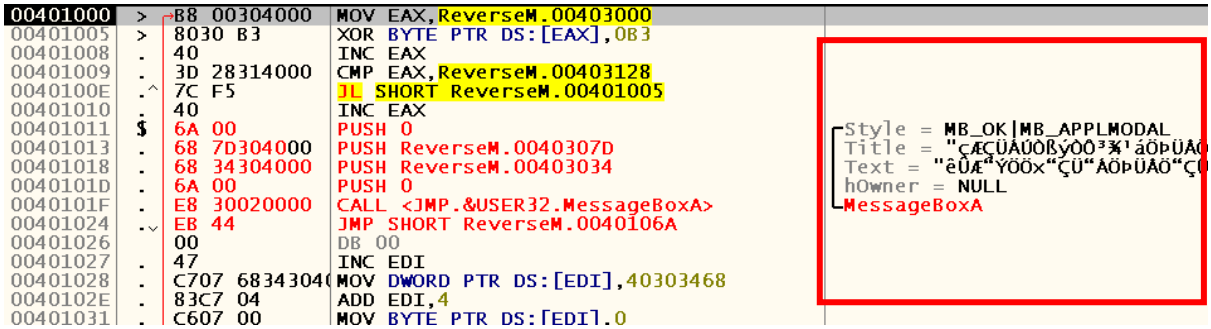


이 부분은 EDI의 값을 바꿔주는 부분으로 보이고 실행해보면

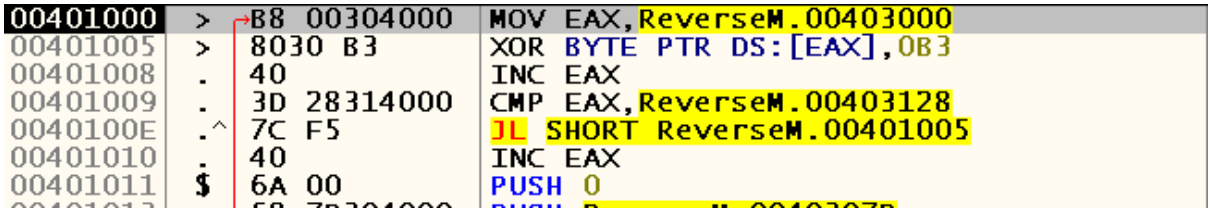


0x00401052에서 두 개의 창을 출력하는 걸 볼 수 있다.

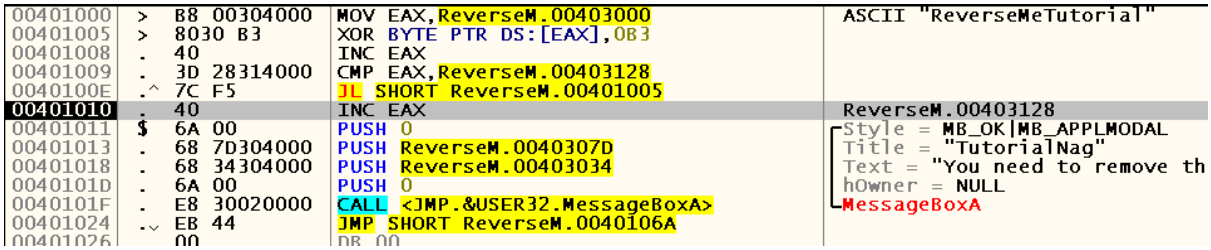




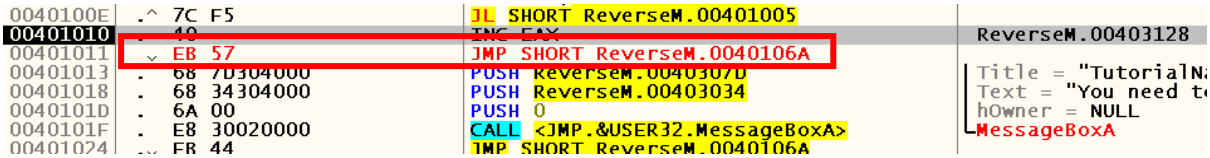
이 함수를 들어가보면 이렇게 보이는데 'ctrl + a'를 누르면 주석 부분이 바뀌는 걸 볼 수 있다.



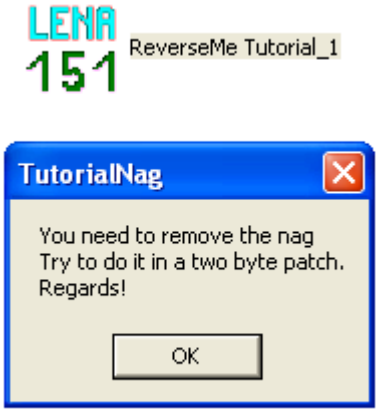
이 부분을 보면 아까랑 똑같이 루프로 보이며 이번에는 403000 – 403128까지 변경하는 걸로 보인다.



루프에서 빠져나오면 아까 주석 부분이 변한 걸 볼 수 있는데 이 메시지박스 함수가 첫 번째 창인 걸 알 수 있다. 근데 이 메시지박스를 실행 안되게 해야된다. 그럼 메시지 박스 실행되고 다음에 뭐가 실행되는지 보면 0x0040106A로 점프하는 걸 볼 수 있는데 0x00401011에서 그냥 0x0040106A 점프 시키면 쉽게 실행 안되게 할 수 있다.

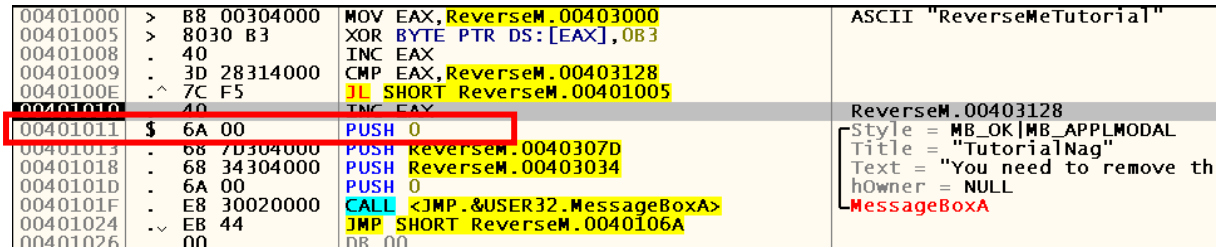


둘 다 2byte이니까 0x00401011을 바꿔주고 저장한다.

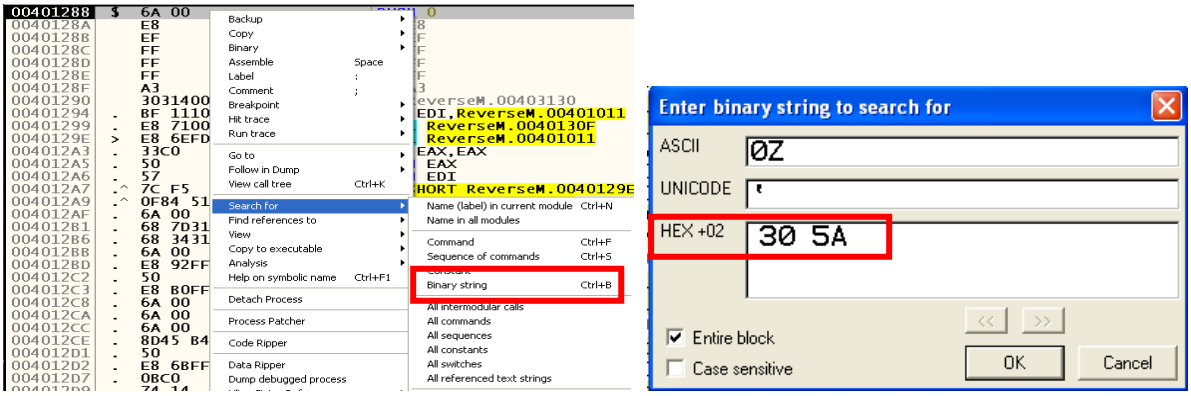


실행해보면 아까와 같은 창이 발생하는 걸 볼 수 있다. 그럼 코드 수정이 잘못된거를 알 수 있다.

안되는 이유는 401000 – 401218에서 복호화를 했는데 거기서 그냥 코드를 수정해버리면 인식을 못하기 때문이다.



6A 00을 5A로 XOR 해주면 30 5A로 바뀌는 걸 볼 수 있다. 그럼 처음 코드에서 30 5A를 찾아서 바꿔줘야한다. -> 이거는 복호화하기 전에 위치를 찾는 거



'ctrl + F2' 누르고 Search for > Binary string 누르고 30 5A 입력해준다.

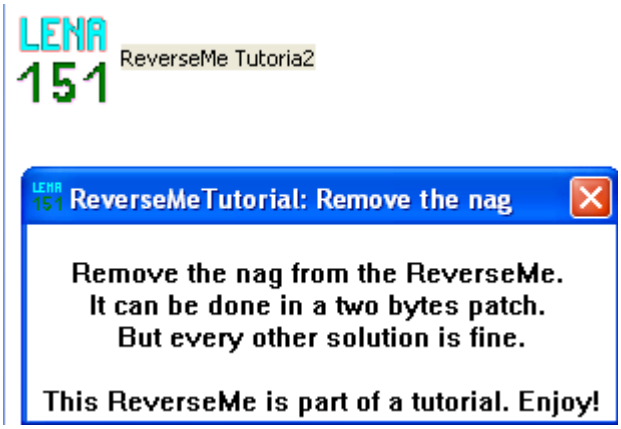
```
00401010    .  1A              DB 1A
00401011    .  699A 3C9D5D30 5AD99D58  IMUL EBX,DWORD PTR DS:[EDX+305D9D3C],5
00401018    .  9D              POPFD
0040101C    .  5D              POP EBP
0040101D    .  3227            XOR AH,BYTE PTR DS:[EDI]
0040101F    .  6A 1A           PUSH 1A
```

그럼 이렇게 찾아주는 걸 볼 수 있다. 중간에 30 5A가 보인다.

아까 0x00401011에서 JMP 코드로 바꿔줬을 때 어셈블리 코드가 EB 57이였는데 이걸 다시 5A를 이용하여 XOR하면 B1 0D가 나온다.

```
0040100E    .  26:AF           SCAS DWORD PTR ES:[EDI]
00401010    .  1A              DB 1A
00401011    .  699A 3C9D5DB1 0DD99D58  IMUL EBX,DWORD PTR DS:[EDX+B15D9D3C],5
00401018    .  9D              POPFD
0040101C    .  5D              POP EBP
```

그럼 30 5A -> B1 0D로 바꿔주고 다시 저장해서 실행해본다.



그럼 첫 번째 창이 안나오는 걸 볼 수 있다.

문제 해결!