Reverse2 L03 Report by vivaman

1. KeyGen 만들기

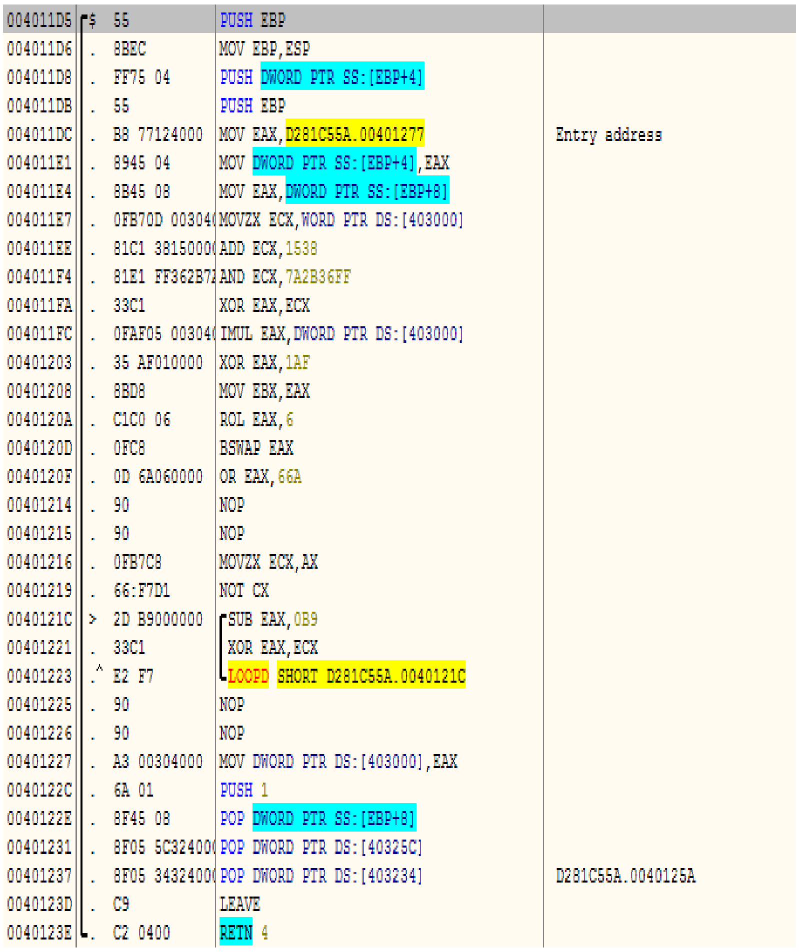
- Name이 CodeEngn 일때 Serial은 무엇인가

- Name 이 입력 될 때마다 코드를 스스로 수정한 후, Serial을 완성합니다.
- 사용Tool: TMG Ripper Studio 0.03, MASM32, OllyDbg
- 3가지 정도가 중요합니다.

➢ 첫째: 코드가 수정된 후 Serial을 완성하는 부분 (KeyGen 소스 일부)

```
LOC_004011D5:
  PUSH EBP
  MOV EBP,ESP
  PUSH DWORD PTR SS:[EBP+4h]
  PUSH EBP
  MOV EAX,LOC_004013DE
  MOV DWORD PTR SS:[EBP+4h],EAX
  MOV EAX,DWORD PTR SS:[EBP+8h]
  LOC_004011E7:ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  ADD BYTE PTR DS:[EAX],AL
  MOV DWORD PTR DS:[403000h],EAX
  PUSH 1
  POP DWORD PTR SS:[EBP+8h]
  POP DWORD PTR DS:[40325Ch]
  POP DWORD PTR DS:[403234h]
  LEAVE
  RETN 4
```

➢ CodeEngn 입력시 코드가 바뀐 화면



```
004011D5  $ 55           PUSH EBP
004011D6  . 8BEC         MOV EBP,ESP
004011D8  . FF75 04      PUSH DWORD PTR SS:[EBP+4]
004011DB  . 55           PUSH EBP
004011DC  . B8 77124000  MOV EAX,D281C55A.00401277                  Entry address
004011E1  . 8945 04      MOV DWORD PTR SS:[EBP+4],EAX
004011E4  . 8B45 08      MOV EAX,DWORD PTR SS:[EBP+8]
004011E7  . 0FB70D 00304 MOVZX ECX,WORD PTR DS:[403000]
004011EE  . 81C1 3815000 ADD ECX,1538
004011F4  . 81E1 FF362B7 AND ECX,7A2B36FF
004011FA  . 33C1         XOR EAX,ECX
004011FC  . 0FAF05 00304 IMUL EAX,DWORD PTR DS:[403000]
00401203  . 35 AF010000  XOR EAX,1AF
00401208  . 8BD8         MOV EBX,EAX
0040120A  . C1C0 06      ROL EAX,6
0040120D  . 0FC8         BSWAP EAX
0040120F  . 0D 6A060000  OR EAX,66A
00401214  . 90           NOP
00401215  . 90           NOP
00401216  . 0FB7C8       MOVZX ECX,AX
00401219  . 66:F7D1      NOT CX
0040121C  > 2D B9000000  SUB EAX,0B9
00401221  . 33C1         XOR EAX,ECX
00401223  .^ E2 F7       LOOPD SHORT D281C55A.0040121C
00401225  . 90           NOP
00401226  . 90           NOP
00401227  . A3 00304000  MOV DWORD PTR DS:[403000],EAX
0040122C  . 6A 01        PUSH 1
0040122E  . 8F45 08      POP DWORD PTR SS:[EBP+8]
00401231  . 8F05 5C32400 POP DWORD PTR DS:[40325C]
00401237  . 8F05 3432400 POP DWORD PTR DS:[403234]                 D281C55A.0040125A
0040123D  . C9           LEAVE
0040123E  . C2 0400      RETN 4
```

➢ 둘째: 실질적으로 코드를 수정하게 하는 부분 (KeyGen 소스 일부)

```
LOC_004013DE:
  PUSH EBP
  MOV EBP,ESP
  MOV EAX,0AAAh
  XOR EAX,DWORD PTR DS:[403000h]
  MOV EAX,7F9h
  SUB DWORD PTR DS:[403000h],EAX
  MOV EAX,LOC_004011D5
  ADD EAX,12h
  PUSH EAX
  XOR EAX,EAX
  BT DWORD PTR DS:[403000h],1
  ADC EAX,0
  OR EAX,EAX
```

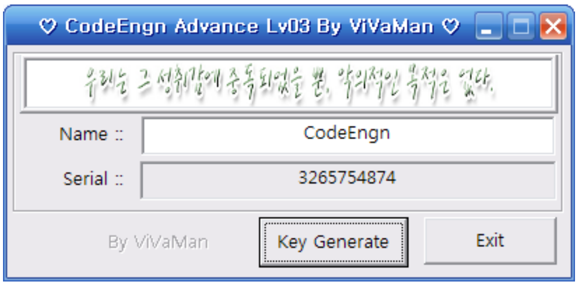
```
  JNZ LOC_004012B1
  MOV EBX,00403004h
  JMP LOC_004012B6
LOC_004012B1:
  MOV EBX,00403044h
LOC_004012B6:
  POP EAX
  XOR ECX,ECX
LOC_004012B9:
  MOV DL,BYTE PTR DS:[EBX+ECX]
  XOR DL,38h
  MOV BYTE PTR DS:[EAX],DL
  INC EAX
  INC ECX
  CMP ECX,1Ch
  JB LOC_004012B9
  PUSH EAX
  XOR EAX,EAX
  BT DWORD PTR DS:[403000h],2h
  ADC EAX,0h
  OR EAX,EAX
  JNZ LOC_004012E1
  MOV EBX,00403020h
  JMP LOC_004012E6
LOC_004012E1:
  MOV EBX,00403060h
LOC_004012E6:
  POP EAX
  XOR ECX,ECX
LOC_004012E9:
  MOV DL,BYTE PTR DS:[EBX+ECX]
  XOR DL,0D4h
  MOV BYTE PTR DS:[EAX],DL
  INC EAX
  INC ECX
  CMP ECX,13h
  JB LOC_004012E9
  PUSH EAX
  XOR EAX,EAX
  BT DWORD PTR DS:[403000h],3h
  ADC EAX,0
  OR EAX,EAX
  JNZ LOC_00401311
  MOV EBX,00403033h
  JMP LOC_00401316
LOC_00401311:
  MOV EBX,00403073h
LOC_00401316:
  POP EAX
  XOR ECX,ECX
LOC_00401319:
  MOV DL,BYTE PTR DS:[EBX+ECX]
  XOR DL,0AFh
  MOV BYTE PTR DS:[EAX],DL
  INC ECX
  INC EAX
  CMP ECX,11h
  JB LOC_00401319
  CMP BYTE PTR SS:[EBP+8],0
  JNZ LOC_00401380
  MOV EAX,DWORD PTR DS:[403258h]
  MOVZX EBX,BYTE PTR DS:[EAX]
  MOVZX ECX,BYTE PTR DS:[EAX+1h]
  SHL EBX,5h
  ADD EBX,2328h
  ADD EBX,ECX
  XOR ECX,0BC614Eh
  IMUL EBX,EBX,0Dh
  XOR EBX,15587h
  PUSH ECX
  PUSH EBX
  CALL LOC_00401241
  XOR EAX,DWORD PTR DS:[403000h]
  MOV DWORD PTR DS:[403000h],EAX
  INC DWORD PTR DS:[403258h]
  MOV EAX,DWORD PTR DS:[403258h]
  INC EAX
  CMP BYTE PTR DS:[EAX],0
  JE LOC_0040138E
```

```
  PUSH 0
  CALL LOC_004013DE
  JMP LOC_0040138E
LOC_00401380:
  MOV EBP,DWORD PTR DS:[40325Ch]
  MOV EAX,DWORD PTR DS:[403234h]
  MOV DWORD PTR SS:[EBP+4h],EAX
LOC_0040138E:
  LEAVE
  RETN 4
```

➢ 셋째: "VirtualProtect" (KeyGen 소스 일부) - 중요-

```
;메모리 보호 해제 시작 - 004011E7
    PUSH        00403260h       ; /pOldProtect = D281C55A.00403260
    PUSH        040h            ; |NewProtect = PAGE_EXECUTE_READWRITE
    PUSH        040h            ; |Size = 40 (64.)
    PUSH        LOC_004011D5    ; |Address = D281C55A.004011E7
    CALL        VirtualProtect  ; ₩VirtualProtect
;해제끝
```

- KeyGen 실행화면



♡ CodeEngn Advance Lv03 By ViVaMan ♡

우리는 그 성취감에 중독되었을 뿐, 악의적인 목적은 없다.

Name :: CodeEngn

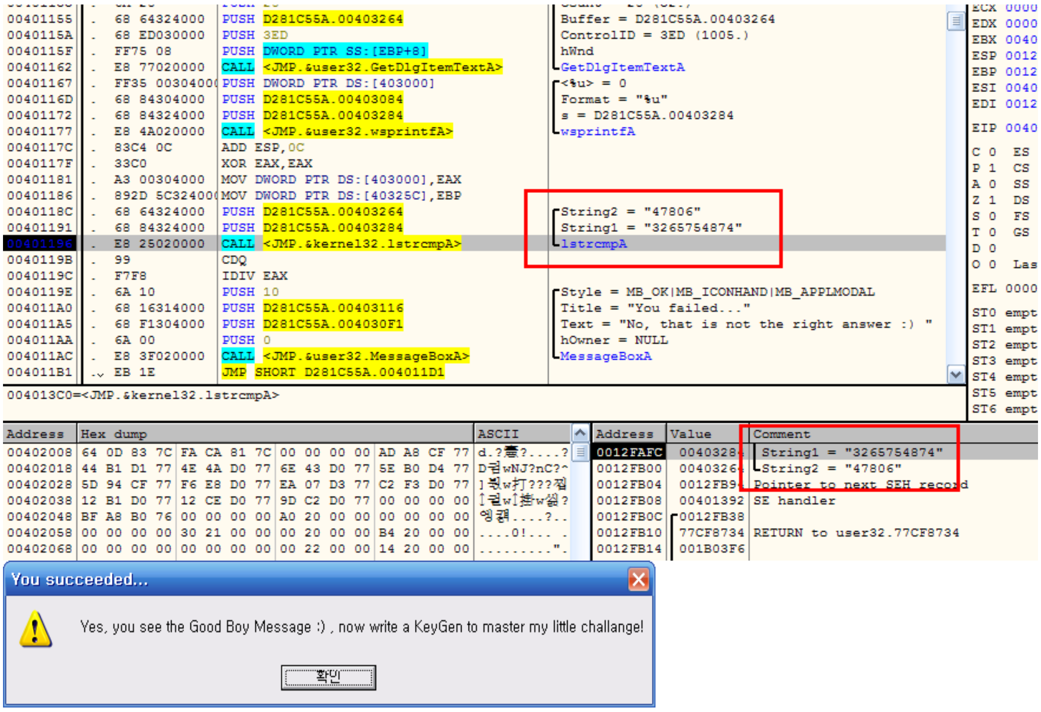
Serial :: 3265754874

후기:

답은 쉬운데, 문제가 어려웠다는 " VirtualProtect "

-끝-

- Serial 에 47806(BABE)를 입력하고 "lstrcmpA"에 브포 걸면
- "3265754874"과 비교합니다. 정답이겠죠..^^



```
00401155  .  68 64324000    PUSH D281C55A.00403264            Buffer = D281C55A.00403264
0040115A  .  68 ED030000    PUSH 3ED                          ControlID = 3ED (1005.)
0040115F  .  FF75 08        PUSH DWORD PTR SS:[EBP+8]         hWnd
00401162  .  E8 77020000    CALL <JMP.&user32.GetDlgItemTextA>  GetDlgItemTextA
00401167  .  FF35 00304000  PUSH DWORD PTR DS:[403000]        <%u> = 0
0040116D  .  68 84304000    PUSH D281C55A.00403084            Format = "%u"
00401172  .  68 84324000    PUSH D281C55A.00403284            s = D281C55A.00403284
00401177  .  E8 4A020000    CALL <JMP.&user32.wsprintfA>      wsprintfA
0040117C  .  83C4 0C        ADD ESP,0C
0040117F  .  33C0           XOR EAX,EAX
00401181  .  A3 00304000    MOV DWORD PTR DS:[403000],EAX
00401186  .  892D 5C324000  MOV DWORD PTR DS:[40325C],EBP
0040118C  .  68 64324000    PUSH D281C55A.00403264            String2 = "47806"
00401191  .  68 84324000    PUSH D281C55A.00403284            String1 = "3265754874"
00401196  .  E8 25020000    CALL <JMP.&kernel32.lstrcmpA>     lstrcmpA
0040119B  .  99             CDQ
0040119C  .  F7F8           IDIV EAX
0040119E  .  6A 10          PUSH 10                           Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
004011A0  .  68 16314000    PUSH D281C55A.00403116            Title = "You failed..."
004011A5  .  68 F1304000    PUSH D281C55A.004030F1            Text = "No, that is not the right answer :) "
004011AA  .  6A 00          PUSH 0                            hOwner = NULL
004011AC  .  E8 3F020000    CALL <JMP.&user32.MessageBoxA>    MessageBoxA
004011B1  .v EB 1E          JMP SHORT D281C55A.004011D1
004013C0=<JMP.&kernel32.lstrcmpA>

Address   Hex dump                                         ASCII
00402008  64 0D 83 7C FA CA 81 7C 00 00 00 00 AD A8 CF 77  d.?燾?.....?
00402018  44 B1 D1 77 4E 4A D0 77 6E 43 D0 77 5E B0 D4 77  D궈wNJ?nC?^
00402028  5D 94 CF 77 F6 E8 D0 77 EA 07 D3 77 C2 F3 D0 77  ]뷁w打???집
00402038  12 B1 D0 77 12 CE D0 77 9D C2 D0 77 00 00 00 00  ↕궈w↕掛w씷?
00402048  BF A8 B0 76 00 00 00 00 A0 20 00 00 00 00 00 00  엥궮....?..
00402058  00 00 00 00 30 21 00 00 00 20 00 00 B4 20 00 00  ....0!... .
00402068  00 00 00 00 00 00 00 00 00 22 00 00 14 20 00 00  .........".

Address   Value     Comment
0012FAFC  00403284  String1 = "3265754874"
0012FB00  00403264  String2 = "47806"
0012FB04  0012FB94  Pointer to next SEH record
0012FB08  00401392  SE handler
0012FB0C  0012FB38
0012FB10  77CF8734  RETURN to user32.77CF8734
0012FB14  001B03F6
```

You succeeded...

Yes, you see the Good Boy Message :) , now write a KeyGen to master my little challange!

- Name: CodeEngn
- Serial: 3265754874