



여태까지의 문제처럼 이름과 시리얼을 입력하고 이를 인증하는 프로그램입니다.

```

CPU - main thread, module 04
00401006 > 6A 00 PUSH 0
00401008 ? E8 C1010000 CALL <JMP.&kernel32.GetModuleHandleA>
0040100D ? A3 BC304000 MOV DWORD PTR DS:[4030BC],EAX
00401012 ? 6A 00 PUSH 0
00401014 ? 68 36104000 PUSH 04.00401036
00401019 ? 6A 00 PUSH 0
0040101B . 68 E9030000 PUSH 3E9
00401020 ? FF35 BC304000 PUSH DWORD PTR DS:[4030BC]
00401026 ? E8 B5010000 CALL <JMP.&user32.DialogBoxParamA>
0040102B ? A3 24314000 MOV DWORD PTR DS:[403124],EAX
00401030 ? 50 PUSH EAX
00401031 ? E8 92010000 CALL <JMP.&kernel32.ExitProcess>
00401036 ? 55 PUSH EBP
00401037 ? 8BEC MOV EBP,ESP
00401039 . 817D 0C 10010 CMP DWORD PTR SS:[EBP+C],110
00401040 75 DB 75 CHAR 'u'
00401041 38 DB 38 CHAR '8'
00401042 68 DB 68 CHAR 'h'
00401043 D0 DB D0
00401044 07 DB 07
00401045 00 DB 00
00401046 00 DB 00
00401047 FF DB FF
00401048 35 DB 35 CHAR '5'
00401049 BC DB BC
0040104A 30 DB 30 CHAR '0'
0040104B 40 DB 40 CHAR '@'

```

처음 프로그램을 실행하고 진행하다 보면 이렇게 나오는데 이는 Ctrl + A를 통해 해결할 수 있습니다.

```

CPU - main thread, module 04
00401006 > 6A 00 PUSH 0
00401008 . E8 C1010000 CALL <JMP.&kernel32.GetModuleHandleA>
0040100D . A3 BC304000 MOV DWORD PTR DS:[4030BC],EAX
00401012 . 6A 00 PUSH 0
00401014 . 68 36104000 PUSH 04.00401036
00401019 . 6A 00 PUSH 0
0040101B . 68 E9030000 PUSH 3E9
00401020 . FF35 BC304000 PUSH DWORD PTR DS:[4030BC]
00401026 . E8 B5010000 CALL <JMP.&user32.DialogBoxParamA>
0040102B . A3 24314000 MOV DWORD PTR DS:[403124],EAX
00401030 . 50 PUSH EAX
00401031 . E8 92010000 CALL <JMP.&kernel32.ExitProcess>
00401036 . 55 PUSH EBP
00401037 . 8BEC MOV EBP,ESP
00401039 . 817D 0C 10010 CMP DWORD PTR SS:[EBP+C],110
00401040 75 38 JNZ SHORT 04.0040107A
00401042 . 68 D0070000 PUSH 7D0
00401047 . FF35 BC304000 PUSH DWORD PTR DS:[4030BC]
0040104D . E8 A6010000 CALL <JMP.&user32.LoadIconA>
00401052 . 50 PUSH EAX
00401053 . 6A 01 PUSH 1
00401055 . 68 90000000 PUSH 80
0040105A . FF75 08 PUSH DWORD PTR SS:[EBP+8]
0040105D . E8 A2010000 CALL <JMP.&user32.SendMessageA>
00401062 . 68 EA030000 PUSH 3EA
00401067 . FF75 08 PUSH DWORD PTR SS:[EBP+8]
0040106A . E8 7D010000 CALL <JMP.&user32.GetDlgItem>

[GetModuleHandleA
  lParam = NULL
 DlgProc = 04.00401036
  hOwner = NULL
  pTemplate = 3E9
  hInst = NULL
DialogBoxParamA

ExitCode
ExitProcess

RsrcName = 2000.
hInst = NULL
LoadIconA
  lParam
  wParam = 1
  message = WM_SETICON
  hWnd
SendMessageA
  ControlID = 3EA (1002.)
  hWnd
GetDlgItem

```

이제 분석을 시작하도록 하겠습니다.

LibertyorDeath's ASM KeygenMe #1

Name: CodeEnqn  
Serial: 1234  
Test About Exit

|          |               |                                    |  |
|----------|---------------|------------------------------------|--|
| 0040108D | . 6A 20       | PUSH 20                            | Count = 20 (32.)<br>Buffer = 04.004030C0   |
| 0040108F | . 68 C0304000 | PUSH 04.004030C0                   |  |
| 00401094 | . 68 EA030000 | PUSH 3EA                           | ControlID = 3EA (1002.)<br>hWnd  |
| 00401099 | . FF75 08     | PUSH DWORD PTR SS:[EBP+8]          |  |
| 0040109C | . E8 51010000 | CALL <JMP.&user32.GetDlgItemTextA> | GetDlgItemTextA  |
| 004010A1 | . A3 00314000 | MOV DWORD PTR DS:[403100],EAX      | Count = 20 (32.)<br>Buffer = 04.004030E0   |
| 004010A6 | . 33C0        | XOR EAX,EAX                        |  |
| 004010A8 | . 6A 20       | PUSH 20                            | ControlID = 3EB (1003.)<br>hWnd  |
| 004010AA | . 68 E0304000 | PUSH 04.004030E0                   |  |
| 004010AF | . 68 EB030000 | PUSH 3EB                           | GetDlgItemTextA  |
| 004010B4 | . FF75 08     | PUSH DWORD PTR SS:[EBP+8]          |  |
| 004010B7 | . E8 36010000 | CALL <JMP.&user32.GetDlgItemTextA> |  |
| 004010BC | . A3 28314000 | MOV DWORD PTR DS:[403128],EAX      | 비교   |
| 004010C1 | . 33C0        | XOR EAX,EAX                        |  |
| 004010C3 | . E8 4C000000 | CALL 04.00401114                   |  |
| 004010C8 | . 75 16       | JNZ SHORT 04.004010E7              | Style = MB_OKIMB_APPLMO<br>Title = "NFO"<br>Text = "This is my first"<br>hOwner<br>MessageBoxA |
| 004010CA | . 3D F1030000 | CMP EAX,3F1                        |  |
| 004010CF | . 75 16       | JNZ SHORT 04.004010E7              |  |
| 004010D1 | . 6A 00       | PUSH 0                             |  |
| 004010D3 | . 68 80304000 | PUSH 04.00403080                   |  |
| 004010D8 | . 68 00304000 | PUSH 04.00403000                   |  |
| 004010DD | . FF75 08     | PUSH DWORD PTR SS:[EBP+8]          |  |
| 004010E0 | . E8 19010000 | CALL <JMP.&user32.MessageBoxA>     |  |
| 004010E5 | . 75 16       | JNZ SHORT 04.004010E7              |  |

00401114=04.00401114

| Address  | Hex dump                | ASCII    |
|----------|-------------------------|----------|
| 004030C0 | 43 6F 64 65 45 6E 67 6E | CodeEnqn |
| 004030C8 | 00 00 00 00 00 00 00 00 | .....    |
| 004030D0 | 00 00 00 00 00 00 00 00 | .....    |
| 004030D8 | 00 00 00 00 00 00 00 00 | .....    |
| 004030E0 | 31 32 33 34 00 00 00 00 | 1234.... |

GetDlgItemText함수를 통해 이름과 시리얼을 버퍼에 저장했습니다.

그 후 주석에 비교라고 되어있는 함수콜이 보입니다.

이름을 이용해서 시리얼을 만들고 입력한 값과 비교하는 함수인데 분석해보도록 하겠습니다.

```

CPU - main thread, module 04
00401114 $ 57 PUSH EDI
00401115 . 833D 00314000 CMP DWORD PTR DS:[403100],0
0040111C ~ 74 73 JE SHORT 04.00401191
0040111E . 833D 28314000 CMP DWORD PTR DS:[403128],0
00401125 ~ 74 6A JE SHORT 04.00401191
00401127 . 33F6 XOR ESI,ESI
00401129 . 33C9 XOR ECX,ECX
0040112B . 33D2 XOR EDX,EDX
0040112D . B9 00000000 MOV ECX,0
00401132 . BA 0A000000 MOV EDX,0A
00401137 > 0FBEB1 C03040 MOVSB EAX, BYTE PTR DS:[ECX+4030C0]
0040113E . 40 INC EAX
0040113F . 03C2 ADD EAX,EDX
00401141 . 03F0 ADD ESI,EAX
00401143 . 41 INC ECX
00401144 . 0FAFD1 IMUL EDX,ECX
00401147 . 8BFA MOV EDI,EDX
00401149 . 0FAFFE IMUL EDI,ESI
0040114C . 3B0D 00314000 CMP ECX,DWORD PTR DS:[403100]
00401152 ^ 75 E3 JNZ SHORT 04.00401137
00401154 . 57 PUSH EDI
00401155 . 56 PUSH ESI
00401156 . 68 AE304000 PUSH 04.004030AE
00401158 . 68 04314000 PUSH 04.00403104
00401160 . E8 75000000 CALL <JMP.&user32.wsprintfA>

```

위의 네모의 부분은 이름과 시리얼이 빈칸인지 아닌지 체크하는 부분입니다.

빈칸이면 루틴을 실행하지 않고 넘어갑니다.

그 밑으로 XOR을 이용해서 초기화 하는 부분을 볼 수 있고 ECX와 EDX에 값을 집어넣습니다.

ECX는 카운팅을 위해 0을 넣고 EDX는 알고리즘에 사용하기 위해 0x0A의 값을 넣습니다.

밑줄 친 부분은 본격적인 알고리즘의 시작으로 EAX에 입력한 이름의 HEX값을 한 글자씩 넣는 부분입니다.

HEX값에 + 1을 하고 EDX와 더한 뒤 그 결과를 ESI에 저장합니다.

EDX와 ECX는 서로 곱하고 그 값을 EDI에 집어넣습니다. 최종적으로 EDI와 ESI를 곱하게 됩니다.

입력한 문자가 모두 해당 알고리즘을 지난 뒤 wsprintf를 통해 시리얼을 만듭니다.

입력값은 EDI와 ESI 두 개인데 EDI는 HEX값(%x) 그대로, ESI는 10진수(%u)로 변환되어 만들어집니다.

```

00401168 . 68 E0304000 PUSH 04.004030E0
0040116D . 68 04314000 PUSH 04.00403104
00401172 . E8 5D000000 CALL <JMP.&kernel32.lstrcmpA>
00401177 . 83F8 00 CMP EAX,0
0040117A . 5F POP EDI
0040117B ~ 75 14 JNZ SHORT 04.00401191
0040117D . 6A 00 PUSH 0
0040117F . 68 92304000 PUSH 04.00403092
00401184 . 68 84304000 PUSH 04.00403084
00401189 . 6A 00 PUSH 0
0040118B . E8 6E000000 CALL <JMP.&user32.MessageBoxA>
00401190 . C3 RETN
00401191 > 6A 00 PUSH 0
00401193 . 68 A7304000 PUSH 04.004030A7
00401198 . 68 97304000 PUSH 04.00403097
0040119D . 6A 00 PUSH 0
0040119F . E8 5A000000 CALL <JMP.&user32.MessageBoxA>
004011A4 . 68 24314000 PUSH 04.00403124
004011A9 . E8 1A000000 CALL <JMP.&kernel32.ExitProcess>

```

입력한 값과 만들어진 시리얼을 비교하고 분기하게 됩니다.