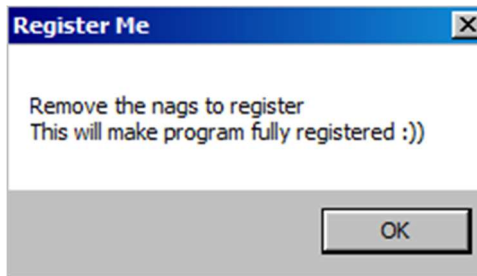


악성코드 분석 보고서

(sand-reversingwithlana-tutorials)

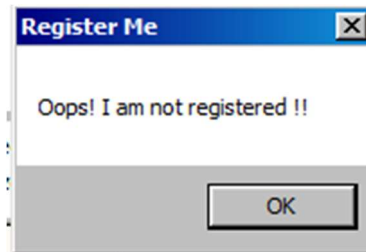
2025.05.27



해석: 등록 메시지를 제거하세요
그러면 프로그램이 완전히 등록됩니다 :))



해석: 지금 저를 등록해야 합니다!
등록은 꽤 쉬운 일입니다.
프로그램을 패치해서 nag 메시지를 제거하세요.



해석: 이런! 저는 등록되지 않았어요!!
즉, 프로그램을 패치해서 nag 메시지를 제거해야함.

```

[ pModule = NULL
  GetModuleHandleA

[ Style = MB_OK|MB_APPLMODAL
  Title = "Register Me"
  Text = "Remove the nags to registerThis
  hOwner = NULL
  MessageBoxA
[ Arg4 = 0000000A
  Arg3 = 00000000
  Arg2 = 00000000
  Arg1 = 00400000
  Register.00401052
[ Style = MB_OK|MB_APPLMODAL
  Title = "Register Me"
  Text = "Oops! I am not registered !!"
  hOwner = NULL
  MessageBoxA
[ ExitCode
  ExitProcess

```

GetModuleHandleA: 지정된 모듈에 대한 모듈 핸들을 검색하고 호출 프로세스에서 모듈을 로드하는 함수

MessageBoxA : 시스템 아이콘, 단추 집합 및 상태 또는 오류 정보와 같은 간단한 애플리케이션 관련 메시지가 포함된 모달 대화 상자를 표시하는 함수

Register.00401052: 실행 파일 안에 있는 자체 함수의 주소

우선 첫 번째 등록메시지를 없애기 위해서는 저 함수를 뛰어 넘어야함.

00401000	\$ 6A 00	PUSH 0
00401002	. E8 0D020000	CALL <JMP.&KERNEL32.GetModuleHandleA>
00401007	. A3 1C314000	MOV DWORD PTR DS:[40311C],EAX
0040100C	. 83F8 00	CMP EAX,0
0040100F	. 74 13	JE SHORT Register.00401024
00401011	. 6A 00	PUSH 0
00401013	. 68 7D304000	PUSH Register.0040307D
00401018	. 68 34304000	PUSH Register.00403034
0040101D	. 6A 00	PUSH 0
0040101F	. E8 C6010000	CALL <JMP.&USER32.MessageBoxA>
00401024	> 6A 0A	PUSH 0A
00401026	. FF35 20314000	PUSH DWORD PTR DS:[403120]

0x0040100F를 뛰어 넘기 위해서는 EAX=0이 되어야함. 하지만 EAX를 보면 0x00400000의 값이 있고 "MZ"가 저장되어 있는 걸 볼 수 있음. 즉, EAX는 절대 0이 될 수 없음.

Registers (FPU)	
EAX 00400000	ASCII "MZ"

00400000	00001000	Register	PE header	Imag	R	RWE
D Dump - Register 00400000..00400FFF						
00400000	4D 5A	ASCII "MZ"	DOS EXE Signature			

"M"(Memory Map)을 누르고 저 "PE header"을 누르면 "Dump"로 들어감.

그럼 EAX에 저장되는 주소가 0x00400000인 걸 알 수 있음.

004000C0	50 45 00 00	ASCII "PE"	PE signature (PE)
004000C4	4C01	DW 014C	Machine = IMAGE_FILE_MACHINE_I386
004000C6	0400	DW 0004	NumberOfSections = 4
004000C8	1E29D138	DD 38D1291E	TimeDateStamp = 38D1291E
004000CC	00000000	DD 00000000	PointerToSymbolTable = 0
004000D0	00000000	DD 00000000	NumberOfSymbols = 0
004000D4	E000	DW 00E0	SizeOfOptionalHeader = E0 (224.)
004000D6	0F01	DW 010F	Characteristics = EXECUTABLE_IMAGE
004000D8	0B01	DW 010B	MagicNumber = PE32
004000DA	05	DB 05	MajorLinkerVersion = 5
004000DB	0C	DB 0C	MinorLinkerVersion = C (12.)
004000DC	00040000	DD 00000400	SizeOfCode = 400 (1024.)
004000E0	000A0000	DD 00000A00	SizeOfInitializedData = A00 (2560.)
004000E4	00000000	DD 00000000	SizeOfUninitializedData = 0
004000E8	00100000	DD 00001000	AddressOfEntryPoint = 1000
004000EC	00100000	DD 00001000	BaseOfCode = 1000
004000F0	00200000	DD 00002000	BaseOfData = 2000
004000F4	00004000	DD 00400000	ImageBase = 400000

Address	Hex dump	ASCII
004000C8	1E 29 D1 38 00 00 00 00 00 00 00 00 E0 00 0F 01)N8.....ä.æ
004000D8	0B 01 05 0C 00 04 00 00 00 0A 00 00 00 00 00 00	z ...J.....
004000E8	00 10 00 00 00 10 00 00 00 20 00 00 00 00 40 00	+...+... ..@.
004000F8	00 10 00 00 00 02 00 00 04 00 00 00 00 00 00 00	+...+...J.....

저장되어 있는 걸 볼 수 있음.

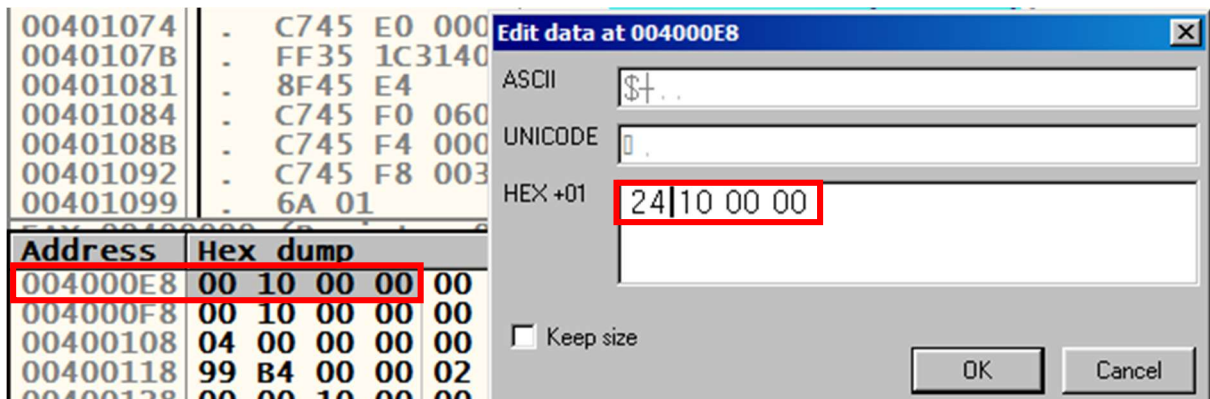
Image_NT_Header > Image_Optional_Header > AddressOfEntryPoint

PE가 로드된 후 실행을 시작할 주소에 대한 RVA를 가지고 있는 중요한 필드로 코드섹션 내에서 특정번지가 됨. EXE PE의 경우 해당 PE가 로드된 후 **이 프로세스의 메인 쓰레드가 실행하는 최초의 코드임**. 일반적으로 이 번지가 가리키는 값은 다음 런타임 시작 루틴의 RVA가 설정됨.

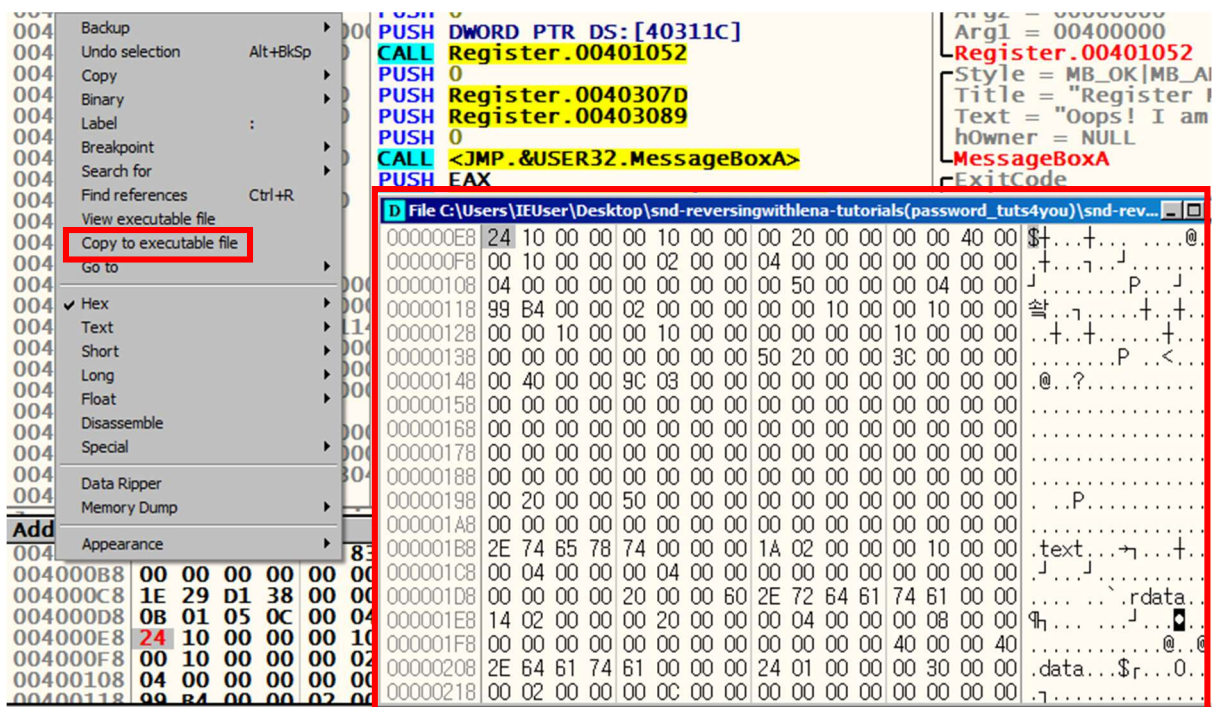
※ RVA (Relative Virtual Address): 특정 기준점(ImageBase)로 부터의 계산된 상대 주소를 뜻함.

VA == RVA + ImageBase

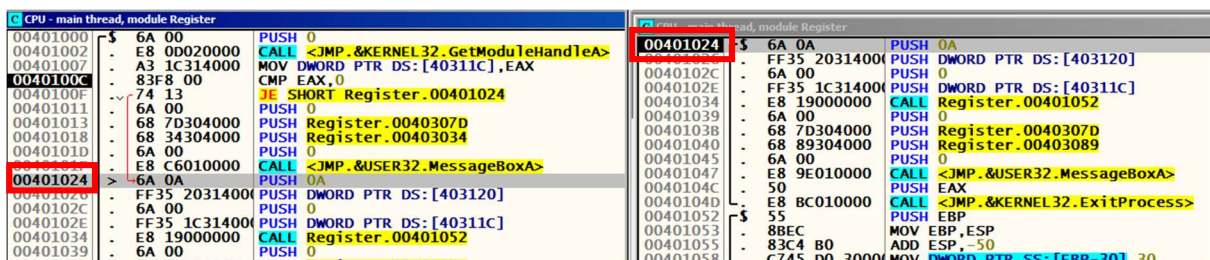
※ VA (Virtual Address): 변하지 않는 프로세스의 가상 메모리의 절대 주소를 뜻함.



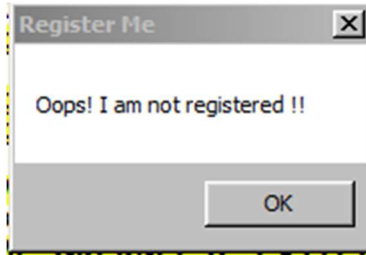
0x004000E8 이 부분을 '24 10 00 00'으로 바꿔줌. 왜냐하면 ImageBase 400000+Entry Point 1000의 값이 시작 점이기 때문에 AddressOfEntryPoint를 바꿔줌. 점프해야 할 주소가 0x00401024이기 때문임.



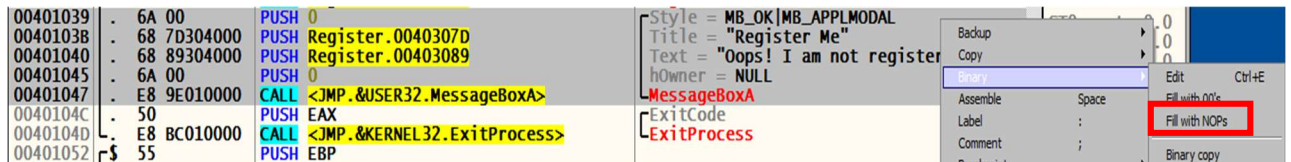
오른쪽 마우스를 누르고 'Copy to executable file'을 누르면 숫자가 바뀐 덤프 창 하나가 생김. 그 창을 다시 저장해주고 OllyDbg로 켜주면



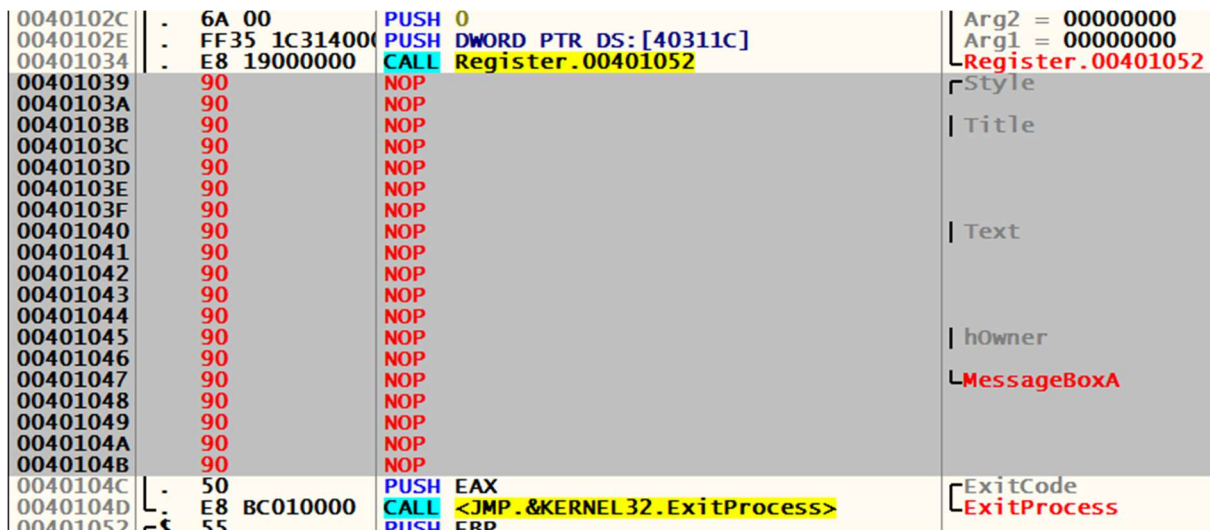
0x00401024부터 시작하는 파일 하나가 생성됨.



실행해보면 이 창이 또 발생하는 걸 볼 수 있음. 이걸 없애기 위해서는



0x00401039 – 0x00401047까지 Nop으로 바꿔줌.



이런 식으로 변경된 것을 확인할 수 있음.

다시 저장 후 클릭해보면 2번째 메시지만 발생하는 걸 볼 수 있음.



해결 완료!