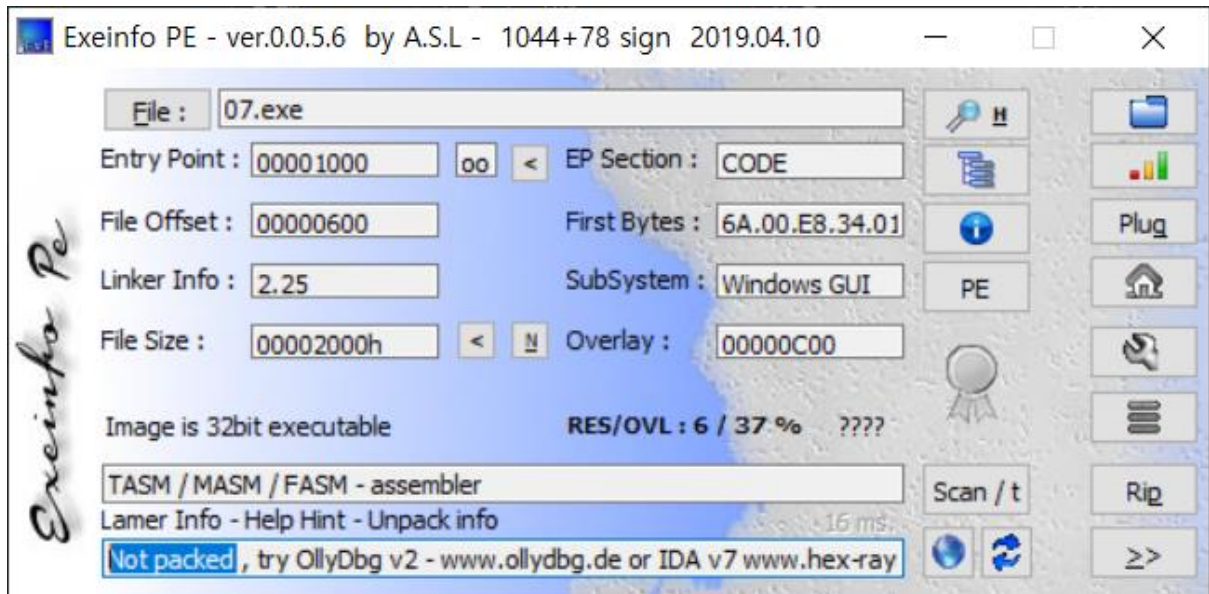
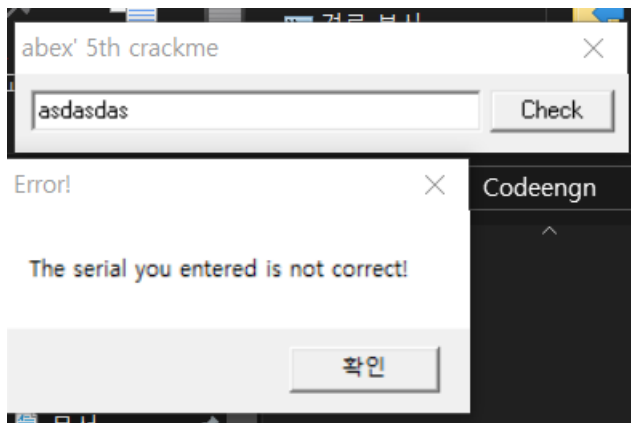


## 06.exe - 컴퓨터 C 드라이브의 이름이 CodeEngn 일경우 시리얼이 생성될때 CodeEngn은 '8어떤것'으로 변경되는가

일단 프로그램을 디버깅 하기 전 PE분석을 해본다.

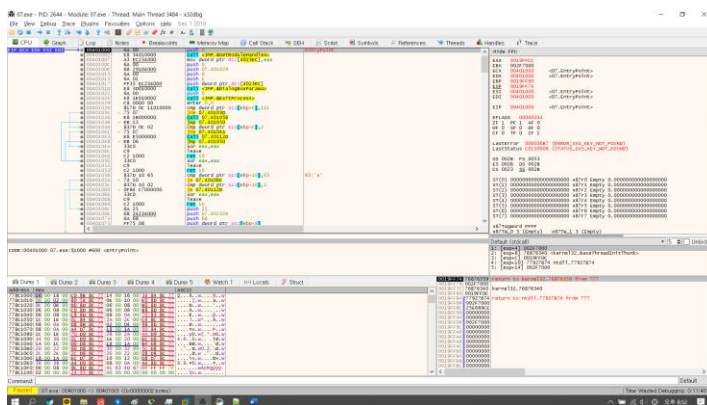


ExeinfoPE를 통해 Not packed가 되었다는 것을 확인함.



프로그램을 실행시 시리얼 키를 입력 받고 확인을 하는 프로그램으로 추정됨.

X32dbg로 실행하여 분석을 시작한다.



일단 Check 버튼을 눌렀을 때 하드디스크의 이름 정보를 가져오는 함수가 작동될 것으로 예상된다.

이때 keyword(volume, disk, information, get 등)을 가지고 인터럽트를 검색해본다.

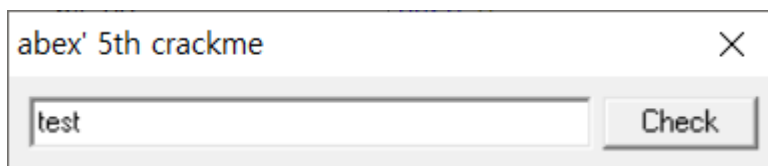
Address	Disassembly	Destination
00401002	call <kernel32.GetModuleHandleA>	<kernel32.GetModuleHandleA>
0040101D	call <kernel32.GetVolumeInformationA>	<kernel32.GetVolumeInformationA>
00401024	call <kernel32.ExitProcess>	<kernel32.ExitProcess>
00401078	call <kernel32.GetDlgItemTextA>	<kernel32.GetDlgItemTextA>
00401099	call <kernel32.GetVolumeInformationA>	<kernel32.GetVolumeInformationA>
004010A6	call <kernel32.lstrcatA>	<kernel32.lstrcatA>
004010D9	call <kernel32.lstrcatA>	<kernel32.lstrcatA>
004010E8	call <kernel32.lstrcatA>	<kernel32.lstrcatA>
004010F7	call <kernel32.lstrcatA>	<kernel32.lstrcatA>
00401110	call <kernel32.lstrcatA>	<kernel32.lstrcatA>
00401126	call <kernel32.lstrcatA>	<kernel32.lstrcatA>
00401132	call <kernel32.lstrcatA>	<kernel32.lstrcatA>

이때 "GetVolumeInformationA" 함수를 발견하게 되는데 이 함수를 통해 볼륨의 이름을 가져오는 것으로 추정된다.

Address	Disassembly	Destination
0040105A	je 07.40106C	
0040105C	cmp dword ptr ss:[ebp+10],2	
00401060	je 07.401120	
00401066	xor eax,eax	
00401068	leave	
00401069	ret 10	
0040106C	push 25	
0040106E	push 07.402324	
00401073	push 68	
00401075	push dword ptr ss:[ebp+8]	
00401078	call <kernel32.GetDlgItemTextA>	
0040107D	push 0	
0040107F	push 0	
00401081	push 07.4020C8	
00401086	push 07.402190	
00401088	push 07.402194	
00401090	push 32	
00401092	push 07.40225C	
00401097	push 0	
00401099	call <kernel32.GetVolumeInformationA>	4023F3: "4562-ABEX"
0040109E	push 07.4023F3	
004010A3	push 07.40225C	
004010A8	call <kernel32.lstrcatA>	
004010AD	mov dl,2	
004010AF	add dword ptr ds:[40225C],1	
004010B6	add dword ptr ds:[40225D],1	
004010BD	add dword ptr ds:[40225E],1	
004010C4	add dword ptr ds:[40225F],1	
004010CB	dec dl	
004010CD	jne 07.4010AF	4023FD: "L2C-5781"
004010CF	push 07.4023FD	
004010D4	push 07.402000	
004010D9	call <kernel32.lstrcatA>	
004010DE	push 07.40225C	
004010E3	push 07.402000	
004010E8	call <kernel32.lstrcatA>	
004010ED	push 07.402324	

이제 함수를 호출하는 주소로 들어가 보면 위와 같이 inputbox에서 입력 값을 받아오고 어떠한 문자열과 "lstrcatA" 함수를 호출하는 것을 볼 수가 있다.

이때 어떠한 작동을 하는지에 대해 분석하기 위해 함수를 호출하는 위치에 Break Point를 걸고 관찰을 한다.



0040105A	74 10	jmp 07.40106C	
0040105C	837D 10 02	cmp dword ptr ss:[ebp+10],2	
00401060	0F84 C7000000	je 07.40112D	
00401066	33C0	xor eax,eax	
00401068	C9	leave	
00401069	C2 1000	ret 10	
0040106C	6A 25	push 25	
0040106E	68 24234000	push 07.402324	
00401073	6A 68	push 68	
00401075	FF75 08	push dword ptr ss:[ebp+8]	[ebp+8]:L"scapi.dll\ext-ms-win-fs-vssapi-11-1-0"
00401078	E8 F4000000	call <JMP.&GetDlgItemTextA>	
0040107D	6A 00	push 0	
0040107F	6A 00	push 0	
00401081	68 C8204000	push 07.4020C8	
00401086	68 90214000	push 07.402190	
00401088	68 94214000	push 07.402194	
00401090	6A 32	push 32	
00401092	68 5C224000	push 07.40225C	

일단 첫번째 Break Point에서 이제 여기서 F8를 눌러 Call를 한다.

0040106E	68 24234000	push 07.402324	402324:"test"
00401073	6A 68	push 68	
00401075	FF75 08	push dword ptr ss:[ebp+8]	[ebp+8]:L"scapi.dll\ext-ms-win-fs-vssapi-11-1-0"
00401078	E8 F4000000	call <JMP.&GetDlgItemTextA>	
0040107D	6A 00	push 0	
0040107F	6A 00	push 0	

실행 결과 402324에 "test" 라는 문자열이 저장된 것을 볼 수 있다.

00401097	6A 00	push 0	
00401099	E8 B5000000	call <JMP.&GetVolumeInformationA>	
0040109E	68 F3234000	push 07.4023F3	4023F3:"4562-ABEX"
004010A3	68 5C224000	push 07.40225C	
004010A8	E8 94000000	call <JMP.&lstrcatA>	
004010AD	B2 02	mov dl,2	
004010AF	8305 5C224000 01	add dword ptr ds:[40225C],1	
004010B6	8305 5D224000 01	add dword ptr ds:[40225D],1	

두번째 Break Point에서 F8를 눌러 Call를 하면 어떠한 변화가 생기는지 관찰을 한다.

00401097	6A 00	push 0	
00401099	E8 B5000000	call <JMP.&GetVolumeInformationA>	
0040109E	68 F3234000	push 07.4023F3	4023F3:"4562-ABEX"
004010A3	68 5C224000	push 07.40225C	40225C:"SSD"
004010A8	E8 94000000	call <JMP.&lstrcatA>	
004010AD	B2 02	mov dl,2	
004010AF	8305 5C224000 01	add dword ptr ds:[40225C],1	0040225C:"SSD"
004010B6	8305 5D224000 01	add dword ptr ds:[40225D],1	0040225D:"SD"
004010BD	8305 5E224000 01	add dword ptr ds:[40225E],1	
004010C4	8305 5F224000 01	add dword ptr ds:[40225F],1	

그 결과 40225C에 저의 하드 볼륨 이름인 SSD가 들어가게 되는 것을 볼 수가 있다.

그리고 Call "lstrcatA"라는 함수를 호출하는데 어떠한 변화가 생기는지 관찰을 한다.

004010A3	68 5C224000	push 07.40225C	40225C:"SSD4562-ABEX"
004010A8	E8 94000000	call <JMP.&lstrcatA>	
004010AD	B2 02	mov dl,2	
004010AF	8305 5C224000 01	add dword ptr ds:[40225C],1	0040225C:"SSD4562-ABEX"
004010B6	8305 5D224000 01	add dword ptr ds:[40225D],1	0040225D:"SD4562-ABEX"
004010BD	8305 5E224000 01	add dword ptr ds:[40225E],1	0040225E:"D4562-ABEX"
004010C4	8305 5F224000 01	add dword ptr ds:[40225F],1	0040225F:"4562-ABEX"
004010CB	FECA	dec dl	
004010CD	75 E0	jne 07.4010AF	
004010CF	68 FD234000	push 07.4023FD	4023FD:"L2C-5781"
004010D4	68 00204000	push 07.402000	

Call를 한 결과 40225C에 시리얼 키로 추정이 되는 문자열과 결합이 되어 "SSD4562-ABEX"가 된 것을 볼 수 있다.

그리고 DL을 2로 설정하고 문자열에 1을 더하는 작업을 여러 번 한 뒤 DL을 1감소시켜 DL이 0이 될 때 까지 반복을 하고 있는 것을 볼 수 있다.

이제 어떻게 문자열이 변화하는지 관찰을 한다.

004010AD	B2 02	mov dl,2		EAX	0040225C
004010AF	8305 5C224000 01	add dword ptr ds:[40225C],1	0040225C:"SSD4562-ABEX"	EBX	00000001
004010B6	8305 5D224000 01	add dword ptr ds:[40225D],1	0040225D:"SD4562-ABEX"	ECX	13EDACA0
004010BD	8305 5E224000 01	add dword ptr ds:[40225E],1	0040225E:"D4562-ABEX"	EDX	00000002
004010C4	8305 5F224000 01	add dword ptr ds:[40225F],1	0040225F:"4562-ABEX"	EBP	0019F800
004010CB	FECA	dec dl		ESP	0019FBCC
004010CD	75 E0	jne 07.4010AF		ESI	000508D2
004010CF	68 FD234000	push 07.4023FD	4023FD:"L2C-5781"	EDI	80006010
004010D4	68 00204000	push 07.402000			
004010D9	FF75 08	call <JMP.&lstrcatA>			

일단 DL은 2로 설정이 된 것을 볼 수 있다.

004010A6	E8 94000000	call <JMP.&lstrcatA>			
004010A7	B2 02	mov di,2			
004010A8	8305 5C224000	add dword ptr ds:[40225C],1	0040225C:"TTE5562-ABEX"	EAX 0040225C	"TTE5562-ABEX"
004010A9	8305 5D224000	add dword ptr ds:[40225D],1	0040225D:"TE5562-ABEX"	EBX 00000001	
004010AB	8305 5E224000	add dword ptr ds:[40225E],1	0040225E:"TE5562-ABEX"	ECX 13EDACA0	
004010AC	8305 5F224000	add dword ptr ds:[40225F],1	0040225F:"5562-ABEX"	EDX 00000002	
004010B0	FECA	dec di		EBP 0019F800	
004010C0	75 E0	jmp 07.4010AF		ESP 0019F8CC	
004010C1	68 FD234000	push 07.4023FD	4023FD:"L2C-5781"	ESI 00050802	L"scapi.dlltext-i
004010D4	68 00204000	push 07.402000		EDI 80006010	
004010D5	FB	cmp al,rcatA			

그리고 앞문자부터 하나씩 1씩 증가하는 것을 4번 하여 TTE5562-ABEX가 되는 것을 볼 수 있다.

이제 DL이 0이 되어 반복문이 끝날 때까지 어떻게 변화하는지 관찰하여 본다.

0040225C "UUF6562-ABEX"

최종적으로 UUF6562-ABEX가 되는 것을 볼 수 있다.

이제 밑에는 있는 부분을 이어서 분석을 해본다.

004010CF	68 FD234000	push 07.4023FD	4023FD:"L2C-5781"
004010D4	68 00204000	push 07.402000	
004010D5	E8 63000000	call <JMP.&lstrcatA>	
004010DE	68 5C224000	push 07.40225C	40225C:"UUF6562-ABEX"
004010E3	68 00204000	push 07.402000	
004010E8	E8 54000000	call <JMP.&lstrcatA>	
004010ED	68 24234000	push 07.402324	402324:"test"
004010F2	68 00204000	push 07.402000	
004010F7	E8 51000000	call <JMP.&lstrcmpiA>	
004010FC	83F8 00	cmp eax,0	eax:"UUF6562-ABEX"
004010FF	74 16	je 07.401117	
00401101	6A 00	push 0	

새로운 키조합에 사용되는 것으로 예상이 되는 문자열 L2C-5781를 stack에 올린다.

004010D5	E8 63000000	call <JMP.&lstrcatA>	
004010DE	68 5C224000	push 07.40225C	40225C:"UUF6562-ABEX"
004010E3	68 00204000	push 07.402000	402000:"L2C-5781"
004010E8	E8 54000000	call <JMP.&lstrcatA>	
004010ED	68 24234000	push 07.402324	402324:"test"
004010F2	68 00204000	push 07.402000	402000:"L2C-5781"
004010F7	E8 51000000	call <JMP.&lstrcmpiA>	
004010FC	83F8 00	cmp eax,0	eax:"L2C-5781"
004010FF	74 16	je 07.401117	
00401101	6A 00	push 0	
00401103	68 34244000	push 07.402434	402434:"Error!"
00401108	68 3B244000	push 07.402438	402438:"The serial you entered is not correct!"
0040110D	FF75 08	push dword ptr ss:[ebp+8]	
00401110	E8 56000000	call <JMP.&MessageBoxA>	
00401115	EB 16	jmp 07.40112D	
00401117	6A 00	push 0	
00401119	68 06244000	push 07.402406	402406:"well Done!"
0040111E	68 11244000	push 07.402411	402411:"Yep, you entered a correct serial!"
00401123	FF75 08	push dword ptr ss:[ebp+8]	
00401128	EB 00	jmp 07.40112D	
0040112D	6A 00	push 0	
0040112F	FF75 08	push dword ptr ss:[ebp+8]	
00401132	FB	cmp al,rcatA	

또한 그 이후 "lstrcatA" 함수를 통해 L2C-5781와UUF6562-ABEX를 합치는 것을 볼 수 있다.

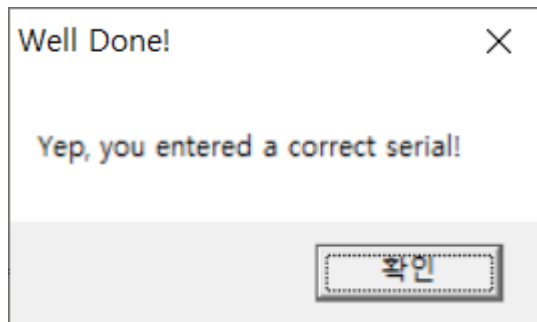
그리고 "lstrcmpiA" 함수를 통해 비교한 결과를 ZF 받고 메시지를 출력하는 것을 볼 수가 있다.

004010D5	E8 63000000	call <JMP.&lstrcatA>	
004010DE	68 5C224000	push 07.40225C	40225C:"UUF6562-ABEX"
004010E3	68 00204000	push 07.402000	402000:"L2C-5781"
004010E8	E8 54000000	call <JMP.&lstrcatA>	
004010ED	68 24234000	push 07.402324	402324:"test"
004010F2	68 00204000	push 07.402000	402000:"L2C-5781"
004010F7	E8 51000000	call <JMP.&lstrcmpiA>	
004010FC	83F8 00	cmp eax,0	
004010FF	74 16	je 07.401117	
00401101	6A 00	push 0	
00401103	68 34244000	push 07.402434	402434:"Error!"
00401108	68 3B244000	push 07.402438	402438:"The serial you entered is not correct!"
0040110D	FF75 08	push dword ptr ss:[ebp+8]	
00401110	E8 56000000	call <JMP.&MessageBoxA>	
00401115	EB 16	jmp 07.40112D	
00401117	6A 00	push 0	
00401119	68 06244000	push 07.402406	402406:"well Done!"
0040111E	68 11244000	push 07.402411	402411:"Yep, you entered a correct serial!"
00401123	FF75 08	push dword ptr ss:[ebp+8]	
00401128	EB 00	jmp 07.40112D	
0040112D	6A 00	push 0	
0040112F	FF75 08	push dword ptr ss:[ebp+8]	
00401132	FB	cmp al,rcatA	

이때 생성된 시리얼값 생성 알고리즘을 분석한 것이 정확한지 직접 넣어서 확인해본다.

abex' 5th crackme

004010D4	68 00204000	push 07.402000	402000:"L2C-5/81UUF6562-ABEX"
004010D9	E8 63000000	call <JMP.&Istrcata>	
004010DE	68 5C224000	push 07.40225C	40225C:"UUF6562-ABEX"
004010E3	68 00204000	push 07.402000	402000:"L2C-5781UUF6562-ABEX"
004010E8	E8 54000000	call <JMP.&Istrcata>	
004010ED	68 24234000	push 07.402324	402324:"L2C-5781UUF6562-ABEX"
004010F2	68 00204000	push 07.402000	402000:"L2C-5781UUF6562-ABEX"
004010F7	E8 51000000	call <JMP.&Istrcmp1A>	
004010FC	83F8 00	cmp eax,0	
004010FF	74 16	je 07.401117	
00401101	6A 00	push 0	
00401103	68 34244000	push 07.402434	402434:"Error!"
00401108	68 3B244000	push 07.40243B	40243B:"The serial you entered is not correct!"
0040110D	FF75 08	push dword ptr ss:[ebp+8]	
00401110	E8 56000000	call <JMP.&MessageBoxA>	
00401115	EB 16	jmp 07.40112D	
00401117	6A 00	push 0	
00401119	68 06244000	push 07.402406	402406:"well done!"
0040111E	68 11244000	push 07.402411	402411:"Yep, you entered a correct serial!"
00401123	FF75 08	push dword ptr ss:[ebp+8]	
00401126	E8 40000000	call <JMP.&MessageBoxA>	
0040112B	EB 00	jmp 07.40112D	
0040112D	6A 00	push 0	
0040112F	FF75 08	push dword ptr ss:[ebp+8]	



이러한 결과를 토대로 시리얼 키를 생성하는 알고리즘은 드라이브의 볼륨이름을 가져와 4562-ABEX와 결합을 하고 앞 4 글자에 1씩더하는 것을 2번 하여서 L2C-5781와 결합하여 최종 시리얼 키를 생성하는 것을 파악할 수 있다.

이제 문제로 돌아와 C드라이브 이름이 CodeEngn일때의 시리얼 키는 "L2C-5781EqfgEngn4562-ABEX" 라는 것을 알 수가 있다. 그리고 CodeEngn은 EgfgEngn으로 변화하는 것을 알 수 있다.

**정답: EgfgEngn**