

CodeEngn Basic RCE L01-L20 풀이 보고서

blbk(blbk.bluebook@gmail.com)

Reverse L01 Start

HDD를 CD-Rom으로 인식시키기 위해서는 GetDriveTypeA의 리턴값이 무엇이 되어야 하는가

http://codeengn.com/menu/challenges/reverse/01/Reverse_L01.rar

00401000	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
00401002	68 00204000	PUSH Reverse_.00402000	Title = "abew' 1st crackme"
00401007	68 12204000	PUSH Reverse_.00402012	Text = "Make me think your HD is a CD-Rom."
0040100C	6A 00	PUSH 0	hOwner = NULL
0040100E	E8 4E000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
00401013	68 94204000	PUSH Reverse_.00402094	RootPathName = "c:\\"
00401018	E8 38000000	CALL <JMP.&KERNEL32.GetDriveTypeA>	GetDriveTypeA
0040101D	46	INC ESI	
0040101E	48	DEC EAX	
0040101F	EB 00	JMP SHORT Reverse_.00401021	
00401021	46	INC ESI	
00401022	46	INC ESI	
00401023	48	DEC EAX	
00401024	3BC6	CMP EAX,ESI	
00401026	74 15	JE SHORT Reverse_.0040103D	
00401028	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
0040102A	68 35204000	PUSH Reverse_.00402035	Title = "Error"
0040102F	68 3B204000	PUSH Reverse_.0040203B	Text = "Nah... This is not a CD-ROM Drive!"
00401034	6A 00	PUSH 0	hOwner = NULL
00401036	E8 26000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
0040103B	EB 13	JMP SHORT Reverse_.00401050	
0040103D	6A 00	PUSH 0	Style = MB_OK!MB_APPLMODAL
0040103F	68 5E204000	PUSH Reverse_.0040205E	Title = "YEAH!"
00401044	68 64204000	PUSH Reverse_.00402064	Text = "Ok, I really think that your HD is a CD-ROM! :p"
00401049	6A 00	PUSH 0	hOwner = NULL
0040104B	E8 11000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
00401050	E8 06000000	CALL <JMP.&KERNEL32.ExitProcess>	ExitProcess
00401055	FF25 50304000	JMP DWORD PTR DS:[<&KERNEL32.GetDriveTypeA	kernel32.GetDriveTypeA
0040105B	FF25 54304000	JMP DWORD PTR DS:[<&KERNEL32.ExitProcess	kernel32.ExitProcess
00401061	FF25 5C304000	JMP DWORD PTR DS:[<&USER32.MessageBoxA	USER32.MessageBoxA

BP를 0x401018, 0x401024에 걸어 실행된 이후의 리턴값과 비교구문을 확인 해 보았다. 실행이후 리턴값인 EAX는 3이고 ESI는 0, CMP구문에서의 EAX는 1이고 ESI는 3이다. 두번의 DEC 명령어로 EAX의 값이 2 감소 했고 두 값이 같아지려면(CD-Rom으로 인식하는 영역으로 점프되므로) ESI값의 +2값인 5가 리턴되어야 한다.

정답 : 5

Reverse L02 Start

패스워드로 인증하는 실행파일이 손상되어 실행이 안되는 문제가 생겼다. 패스워드가 무엇인지 분석하시오

<http://codeengn.com/menu/challenges/reverse/02/023C27FA.exe>

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000740	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000750	41	44	44	69	61	6C	6F	67	00	41	72	74	75	72	44	65	ADDIALOG.ArturDe
00000760	6E	74	73	20	43	72	61	63	6B	4D	65	23	31	00	00	00	nts CrackMe#1...
00000770	00	00	00	00	00	4E	6F	70	65	2C	20	74	72	79	20	61Nope, try a
00000780	67	61	69	6E	21	00	59	65	61	68	2C	20	79	6F	75	20	gain!.Yeah, you
00000790	64	69	64	20	69	74	21	00	43	72	61	63	6B	6D	65	20	did it!.Crackme
000007A0	23	31	00	4A	4B	33	46	4A	5A	68	00	00	00	00	00	00	#1.JK3FJZh.....
000007B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

헥스 에디터인 HxD로 바이너리를 열어보았더니 답이 나왔다.

정답 : JK3FJZh

Reverse L03 Start

비주얼베이직에서 스트링 비교함수 이름은?

<http://codeengn.com/menu/challenges/reverse/03/A2DC1DEA.exe>

Regcode에 아무 값이나 넣고 체크해보았더니 틀렸다는 메시지가 나왔고 그 주변코드를 살펴보니 비교구문이 있었다.

00402A07	FF91 A0000000	CALL DWORD PTR DS:[ECX+A0]	
00402A0D	85C0	TEST EAX,EAX	
00402A0F	7D 16	JGE SHORT A2DC1DEA.00402A27	
00402A11	68 A0000000	PUSH 0A0	
00402A16	68 F41D4000	PUSH A2DC1DEA.00401DF4	
00402A1B	FFB5 50FFFFFF	PUSH DWORD PTR SS:[EBP-B0]	
00402A21	50	PUSH EAX	
00402A22	E8 17E7FFFF	CALL <JMP.&MSUBUM50.__vbaHresultCheckOb	
00402A27	FF75 A8	PUSH DWORD PTR SS:[EBP-58]	
00402A29	68 DC1D4000	PUSH A2DC1DEA.00401DDC	
00402A2F	E8 16E7FFFF	CALL <JMP.&MSUBUM50.__vbaStrCmp>	UNICODE "2683G35Hs2"
00402A34	F7D8	NEG EAX	
00402A36	1BC0	SBB EAX,EAX	
00402A38	8D4D A8	LEA ECX,DWORD PTR SS:[EBP-58]	
00402A3B	F7D8	NEG EAX	
00402A3D	F7D8	NEG EAX	
00402A3F	8985 48FFFFFF	MOV DWORD PTR SS:[EBP-B8],EAX	

입력한 문자열을 스택에 PUSH하고 정답을 PUSH하여 _vbsStrCmp함수를 Call하여 비교한다.

정답 : vbaStrCmp

Reverse L04 Start

이 프로그램은 디버거 프로그램을 탐지하는 기능을 갖고 있다. 디버거를 탐지하는 함수의 이름은 무엇인가

<http://codeengn.com/menu/challenges/reverse/04/AFA7AD21.exe>

00401048	> 8BF4	MOV ESI,ESP	
0040104A	. 68 E8030000	PUSH 3E8	
0040104F	. FF15 68B14300	CALL DWORD PTR DS:[<&KERNEL32.Sleep>]	[Timeout = 1000. ms Sleep
00401055	. 3BF4	CMP ESI,ESP	
00401057	. E8 B4710000	CALL AFA7AD21.00408210	
0040105C	. 8BF4	MOV ESI,ESP	
0040105E	. FF15 64B14300	CALL DWORD PTR DS:[<&KERNEL32.IsDebuggerPresent>]	[IsDebuggerPresent
00401064	. 3BF4	CMP ESI,ESP	
00401066	. E8 A5710000	CALL AFA7AD21.00408210	
0040106B	. 85C0	TEST EAX,EAX	
0040106D	. 74 0F	JE SHORT AFA7AD21.0040107E	
0040106F	. 68 24104300	PUSH AFA7AD21.00431024	[Arg1 = 00431024
00401074	. E8 17710000	CALL AFA7AD21.00408190	[AFA7AD21.00408190
00401079	. 83C4 04	ADD ESP,4	
0040107C	. EB 0D	JMP SHORT AFA7AD21.0040108B	
0040107E	> 68 1C104300	PUSH AFA7AD21.0043101C	[Arg1 = 0043101C
00401083	. E8 08710000	CALL AFA7AD21.00408190	[AFA7AD21.00408190
00401088	. 83C4 04	ADD ESP,4	
0040108B	> EB BB	JMP SHORT AFA7AD21.00401048	

프로그램 내부에 IsDebuggerPresent 함수를 계속 호출하는 루프가 있다. 디버깅을 당하고 있는 중이면 1을 리턴하여 "디버깅 당함"을 출력하고 그렇지 않으면 "정상"을 출력하는 쪽으로 점프한다.

정답 : IsDebuggerPresent

Reverse L05 Start

이 프로그램의 등록키는 무엇인가

<http://codeengn.com/menu/challenges/reverse/05/FFCD7DC6.exe>

PEiD로 확인한 결과 UPX로 패킹되어 있었고 패킹을 푼 뒤 코드를 확인해 보았다.

00440F2C	. 8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00440F2F	. BA 14104400	MOV EDX,FFCD7DC6.00441014	
00440F34	. E8 F32BFCFF	CALL FFCD7DC6.00403B2C	ASCII "Registered User"
00440F39	. 75 51	JNZ SHORT FFCD7DC6.00440F8C	
00440F3B	. 8D55 FC	LEA EDX,DWORD PTR SS:[EBP-4]	
00440F3E	. 8B83 C8020000	MOV EAX,DWORD PTR DS:[EBX+2C8]	
00440F44	. E8 D7FEFDFF	CALL FFCD7DC6.00420E20	
00440F49	. 8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]	
00440F4C	. BA 2C104400	MOV EDX,FFCD7DC6.0044102C	ASCII "GFX-754-IER-954"
00440F51	. E8 D62BFCFF	CALL FFCD7DC6.00403B2C	
00440F56	. 75 1A	JNZ SHORT FFCD7DC6.00440F72	
00440F58	. 6A 00	PUSH 0	
00440F5A	. B9 3C104400	MOV ECX,FFCD7DC6.0044103C	ASCII "CrackMe cracked successfully"
00440F5F	. BA 5C104400	MOV EDX,FFCD7DC6.0044105C	ASCII "Congrats! You cracked this CrackMe!"
00440F64	. A1 442C4400	MOV EAX,DWORD PTR DS:[442C44]	
00440F69	. 8B00	MOV EAX,DWORD PTR DS:[EAX]	
00440F6B	. E8 F8C0FFFF	CALL FFCD7DC6.0043D068	
00440F70	. EB 32	JMP SHORT FFCD7DC6.00440FA4	
00440F72	. 6A 00	PUSH 0	
00440F74	. B9 80104400	MOV ECX,FFCD7DC6.00441080	ASCII "Beggars off!"
00440F79	. BA 8C104400	MOV EDX,FFCD7DC6.0044108C	ASCII "Wrong Serial, try again!"
00440F7E	. A1 442C4400	MOV EAX,DWORD PTR DS:[442C44]	
00440F83	. 8B00	MOV EAX,DWORD PTR DS:[EAX]	

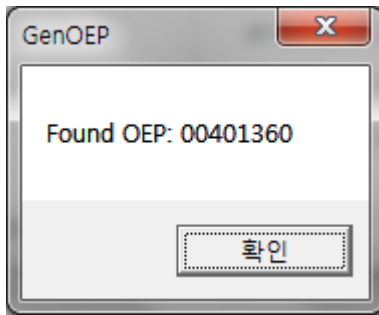
BP를 걸어둔 각 비교함수를 통하여 이름이 "Registered User"와 동일한지와 등록키가 "GFX-754-IER-954"와 동일한지를 확인하고 둘다 맞으면 통과한다.

정답 : GFX-754-IER-954

Reverse L06 Start

Unpack을 한 후 Serial을 찾으시오. 정답인증은 OEP + Serial

<http://codeengn.com/menu/challenges/reverse/06/386D13B0.exe>



PEiD로 알아낸 OEP는 0x00401360, UPX로 패킹되어있어 패킹을 풀고 코드를 확인해 보았다.

00401069	68 04354200	PUSH 386D13B0.004235D4	
0040106E	68 302A4200	PUSH 386D13B0.00422A30	ASCII "AD46DFS547"
00401073	E8 18020000	CALL 386D13B0.00401290	
00401078	83C4 08	ADD ESP,8	
0040107B	85C0	TEST EAX,EAX	
0040107D	75 24	JNZ SHORT 386D13B0.004010A3	
0040107F	8BF4	MOV ESI,ESP	
00401081	6A 40	PUSH 40	
00401083	68 40004200	PUSH 386D13B0.00420048	ASCII "Good Job!"
00401088	68 30004200	PUSH 386D13B0.00420038	ASCII "You got it ;)"
0040108D	8B0D 38364200	MOV ECX, DWORD PTR DS:[423638]	
00401093	51	PUSH ECX	
00401094	FF15 B4524200	CALL DWORD PTR DS:[<&USER32.MessageBoxA	USER32.MessageBoxA
0040109A	3BF4	CMP ESI,ESP	
0040109C	E8 7F020000	CALL 386D13B0.00401320	
004010A1	EB 22	JMP SHORT 386D13B0.004010C5	
004010A3	8BF4	MOV ESI,ESP	
004010A5	6A 10	PUSH 10	
004010A7	68 30004200	PUSH 386D13B0.00420030	ASCII "ERROR"
004010AC	68 1C004200	PUSH 386D13B0.0042001C	ASCII "Wrong serial!!!"

입력값과 "AD46DFS547"을 PUSH하여 비교하는것을 알 수 있다.

정답 : 00401360AD46DFS547

Reverse L07 Start

컴퓨터 C 드라이브의 이름이 CodeEngn 일경우 시리얼이 생성될때 CodeEngn은 "어떤것"으로 변경되는가

<http://codeengn.com/menu/challenges/reverse/07/BABB04F7.exe>

0040106C	>	6A 25	PUSH 25	Count = 25 (37.)
0040106E	.	68 24234000	PUSH BABB04F7.00402324	Buffer = BABB04F7.00402324
00401073	.	6A 68	PUSH 68	ControlID = 68 (104.)
00401075	.	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
00401078	.	E8 F4000000	CALL <JMP.&USER32.GetDlgItemTextA>	GetDlgItemTextA
0040107D	.	6A 00	PUSH 0	pFileSystemNameSize = NULL
0040107F	.	6A 00	PUSH 0	pFileSystemNameBuffer = NULL
00401081	.	68 C3204000	PUSH BABB04F7.004020C8	pFileSystemFlags = BABB04F7.004020C8
00401086	.	68 90214000	PUSH BABB04F7.00402190	pMaxFilenameLength = BABB04F7.00402190
00401088	.	68 94214000	PUSH BABB04F7.00402194	pVolumeSerialNumber = BABB04F7.00402194
00401090	.	6A 32	PUSH 32	MaxVolumeNameSize = 32 (50.)
00401092	.	68 5C224000	PUSH BABB04F7.0040225C	VolumeNameBuffer = BABB04F7.0040225C
00401097	.	6A 00	PUSH 0	RootPathName = NULL
00401099	.	E8 B5000000	CALL <JMP.&KERNEL32.GetVolumeInformationA>	GetVolumeInformationA
0040109E	.	68 F3234000	PUSH BABB04F7.004023F3	StringToAdd = "4562-ABEX"
004010A3	.	68 5C224000	PUSH BABB04F7.0040225C	ConcatString = ""
004010A8	.	E8 94000000	CALL <JMP.&KERNEL32.lstrcatA>	lstrcatA
004010AD	.	B2 02	MOV DL,2	
004010AF	>	8305 5C224000	ADD DWORD PTR DS:[40225C],1	
004010B6	.	8305 5D224000	ADD DWORD PTR DS:[40225D],1	
004010BD	.	8305 5E224000	ADD DWORD PTR DS:[40225E],1	
004010C4	.	8305 5F224000	ADD DWORD PTR DS:[40225F],1	
004010C8	.	FECA	DEC DL	
004010CD	^	75 E0	JNZ SHORT BABB04F7.004010AF	
004010CF	.	68 FD234000	PUSH BABB04F7.004023FD	StringToAdd = "L2C-5781"
004010D4	.	68 00204000	PUSH BABB04F7.00402000	ConcatString = ""
004010D9	.	E8 63000000	CALL <JMP.&KERNEL32.lstrcatA>	lstrcatA
004010DE	.	68 5C224000	PUSH BABB04F7.0040225C	StringToAdd = ""
004010E3	.	68 00204000	PUSH BABB04F7.00402000	ConcatString = ""
004010E8	.	E8 54000000	CALL <JMP.&KERNEL32.lstrcatA>	lstrcatA
004010ED	.	68 24234000	PUSH BABB04F7.00402324	String2 = ""
004010F2	.	68 00204000	PUSH BABB04F7.00402000	String1 = ""
004010F7	.	E8 51000000	CALL <JMP.&KERNEL32.lstrcmpiA>	lstrcmpiA
004010FC	.	83F8 00	CMP EAX,0	
004010FF	√	74 16	JE SHORT BABB04F7.00401117	
00401101	.	6A 00	PUSH 0	Style = MB_OK MB_APPLMODAL
00401103	.	68 34244000	PUSH BABB04F7.00402434	Title = "Error!"
00401108	.	68 3B244000	PUSH BABB04F7.0040243B	Text = "The serial you entered is not correct!"
0040110D	.	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hOwner
00401110	.	E8 56000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA
00401115	√	EB 16	JMP SHORT BABB04F7.0040112D	
00401117	>	6A 00	PUSH 0	Style = MB_OK MB_APPLMODAL
00401119	.	68 06244000	PUSH BABB04F7.00402406	Title = "Well Done!"
0040111E	.	68 11244000	PUSH BABB04F7.00402411	Text = "Yep, you entered a correct serial!"
00401123	.	FF75 08	PUSH DWORD PTR SS:[EBP+8]	hOwner
00401126	.	E8 40000000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA

코드를 보면 VolumeNameBuffer로 사용되는 주소값이 0x0040225c이고 밑에 lstrcatA에서도 인자로 쓰이는 것을 알 수 있다.

Address	Hex dump	ASCII
0040225C	43 6F 64 65 45 6E 67 6E 00 00 00 00 00 00 00 00	CodeEngn.....
0040226C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040227C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

GetVolumeInformationA 함수 호출후 0x0040225c값을 "CodeEngn"으로 수정하고 lstrcmpiA 전까지 실행 해 보았다.

004010AF	>	8305 5C224000	ADD DWORD PTR DS:[40225C],1	
004010B6	.	8305 5D224000	ADD DWORD PTR DS:[40225D],1	
004010BD	.	8305 5E224000	ADD DWORD PTR DS:[40225E],1	
004010C4	.	8305 5F224000	ADD DWORD PTR DS:[40225F],1	
004010C8	.	FECA	DEC DL	
004010CD	^	75 E0	JNZ SHORT BABB04F7.004010AF	
004010CF	.	68 FD234000	PUSH BABB04F7.004023FD	StringToAdd = "L2C-5781"
004010D4	.	68 00204000	PUSH BABB04F7.00402000	ConcatString = "L2C-5781EqfgEngn4562-ABEX"
004010D9	.	E8 63000000	CALL <JMP.&KERNEL32.lstrcatA>	lstrcatA
004010DE	.	68 5C224000	PUSH BABB04F7.0040225C	StringToAdd = "EqfgEngn4562-ABEX"
004010E3	.	68 00204000	PUSH BABB04F7.00402000	ConcatString = "L2C-5781EqfgEngn4562-ABEX"
004010E8	.	E8 54000000	CALL <JMP.&KERNEL32.lstrcatA>	lstrcatA
004010ED	.	68 24234000	PUSH BABB04F7.00402324	String2 = "1234567890"
004010F2	.	68 00204000	PUSH BABB04F7.00402000	String1 = "L2C-5781EqfgEngn4562-ABEX"
004010F7	.	E8 51000000	CALL <JMP.&KERNEL32.lstrcmpiA>	lstrcmpiA

앞 4자리를 각각 1씩 더하여 "CodeEngn"이 "EqfgEngn"으로 변경되었다.

정답 : EqfgEngn

Reverse L08 Start

OEP를 구하시오 Ex) 00400000

<http://codeengn.com/menu/challenges/reverse/08/1BF0364F.exe>

010200B0	. 61	POPAD
010200BE	. 8D4424 80	LEA EAX,DWORD PTR SS:[ESP-80]
010200C2	> 6A 00	PUSH 0
010200C4	. 39C4	CMP ESP,EAX
010200C6	. ^ 75 FA	JNZ SHORT 1BF0364F.010200C2
010200C8	. 83EC 80	SUB ESP,-80
010200CB	.- E9 A516FFFF	JMP 1BF0364F.01012475
010200D0	00	DB 00
010200D1	00	DB 00
010200D2	00	DB 00
010200D3	00	DB 00
010200D4	00	DB 00
010200D5	00	DB 00
010200D6	00	DB 00

OEP부분으로 가는 JMP문이 보인다.

정답 : 01012475

Reverse L09 Start

StolenByte를 구하시오 Ex) 75156A0068352040

<http://codeengn.com/menu/challenges/reverse/09/7EA5E6FB.exe>

0040736E	. 6A 00	PUSH 0	ASCII "abex' 3rd crackme" ASCII "Click OK to check for the keyfile."
00407370	. 68 00204000	PUSH 7EA5E6FB.00402000	
00407375	. 68 12204000	PUSH 7EA5E6FB.00402012	
0040737A	. 8D4424 80	LEA EAX,DWORD PTR SS:[ESP-80]	
0040737E	> 6A 00	PUSH 0	
00407380	. 39C4	CMP ESP,EAX	
00407382	. ^ 75 FA	JNZ SHORT 7EA5E6FB.0040737E	
00407384	. 83EC 80	SUB ESP,-80	
00407387	.- E9 809CFFFF	JMP 7EA5E6FB.0040100C	

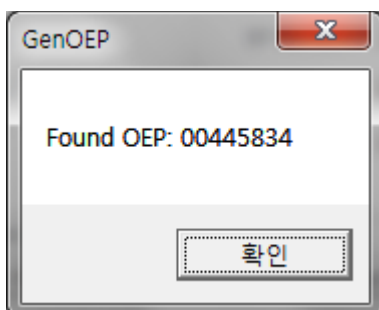
OEP부분으로 가기 전 StolenByte가 보인다.

정답 : 6A0068002040006812204000

Reverse L10 Start

OEP를 구한 후 "등록성공"으로 가는 분기점의 OPCODE를 구하시오. 정답인증은 OEP + OPCODE

<http://codeengn.com/menu/challenges/reverse/10/B1359C15.exe>



004454D4	. 75 55	JNZ SHORT B1359C15.0044552B	
004454D6	. 8D85 F4FDFFF	LEA EAX,DWORD PTR SS:[EBP-20C]	
004454DC	. 8D95 17FEFFF	LEA EDX,DWORD PTR SS:[EBP-1E9]	
004454E2	. E8 1DE6FBFF	CALL B1359C15.00403B04	
004454E7	. 8B95 F4FDFFF	MOV EDX,DWORD PTR SS:[EBP-20C]	
004454ED	. 8B87 D402000	MOV EAX,DWORD PTR DS:[EDI+2D4]	
004454F3	. E8 B4F5FDFF	CALL B1359C15.00424AAC	
004454F8	. 8B87 D802000	MOV EAX,DWORD PTR DS:[EDI+2D8]	
004454FE	. 8B55 FC	MOV EDX,DWORD PTR SS:[EBP-4]	
00445501	. E8 A6F5FDFF	CALL B1359C15.00424AAC	
00445506	. 8B87 E802000	MOV EAX,DWORD PTR DS:[EDI+2E8]	
0044550C	. BA 60564400	MOV EDX,B1359C15.00445660	
00445511	. E8 96F5FDFF	CALL B1359C15.00424AAC	
00445516	. 8B87 E802000	MOV EAX,DWORD PTR DS:[EDI+2E8]	
0044551C	. 8B40 58	MOV EAX,DWORD PTR DS:[EAX+58]	
0044551F	. BA 00800000	MOV EDX,8000	
00445524	. E8 BFF2FCFF	CALL B1359C15.004147E8	

ASCII "Registered ... well done!"

PEiD로 알아낸 OEP는 0x00445834, 분기점의 OP코드인 "7555"인것을 알 수 있다.

정답 : 004458347555

Reverse L11 Start

OEP를 찾으시오. Ex) 00401000 / StolenByte 를 찾으시오. Ex) FF35CA204000E84D000000

정답인증은 OEP+ StolenByte Ex) 00401000FF35CA204000E84D000000

<http://codeengn.com/menu/challenges/reverse/11/F5B19BBB.exe>

0040736E	. 6A 00	PUSH 0	
00407370	. 68 00204000	PUSH F5B19BBB.00402000	
00407375	. 68 12204000	PUSH F5B19BBB.00402012	
0040737A	. 8D4424 80	LEA EAX,DWORD PTR SS:[ESP-80]	
0040737E	> 6A 00	PUSH 0	
00407380	. 39C4	CMPL ESP,EAX	
00407382	. ^ 75 FA	JNZ SHORT F5B19BBB.0040737E	
00407384	. 83EC 80	SUB ESP,-80	
00407387	. E9 809CFFFF	JMP F5B19BBB.0040100C	
0040738C	. 00	DB 00	
0040738D	. 00	DB 00	
0040738E	. 00	DB 00	
0040738F	. 00	DB 00	
00407390	. 00	DB 00	
00407391	. 00	DB 00	
00407392	. 00	DB 00	
00407393	. 00	DB 00	

ASCII "abex' 3rd crackme"
ASCII "Click OK to check for the keyfile."

Stolenbyte는 "6A0068002040006812204000"이다.

00401000	90	NOP	
00401001	90	NOP	
00401002	90	NOP	
00401003	90	NOP	
00401004	90	NOP	
00401005	90	NOP	
00401006	90	NOP	
00401007	90	NOP	
00401008	90	NOP	
00401009	90	NOP	
0040100A	90	NOP	
0040100B	90	NOP	
0040100C	6A 00	PUSH 0	
0040100E	E8 8C000000	CALL F5B19BBB.0040109F	
00401013	6A 00	PUSH 0	
00401015	68 80000000	PUSH 80	
0040101A	6A 03	PUSH 3	
0040101C	6A 00	PUSH 0	
0040101E	6A 00	PUSH 0	

JMP to USER32.MessageBoxA

OEP는 0x00401000이다.

정답 : 004010006A0068002040006812204000

Reverse L12 Start

Key를 구한 후 입력하게 되면 성공메시지를 볼 수 있다. 이때 성공메시지 대신 Key 값이 MessageBox에 출력 되도록 하려면 파일을 HexEdit로 오픈 한 다음 0x???? ~ 0x???? 영역에 Key 값을 overwrite 하면 된다.

Key값과 + 주소영역을 찾으시오 Ex) 77777777???????

<http://codeengn.com/menu/challenges/reverse/12/B643D2BD.exe>

00401052	• 6A 00	PUSH 0	IsSigned = FALSE
00401054	• 6A 00	PUSH 0	oSuccess = NULL
00401056	• 68 B9050000	PUSH 0BB9	ControlID = BB9 (3001.)
00401058	• FF75 08	PUSH DWORD PTR SS:[EBP+8]	hWnd
0040105E	• E8 31010000	CALL <JMP.&USER32.GetDlgItemInt>	GetDlgItemInt
00401063	• BE 00304000	MOV ESI,B643D2BD.00403000	ASCII "0q1qb4EhM/4jISMj1zQf6kpG0wLrG+6EIV4bPc0JL/jWBfNLeJnbme3garIrG4ngbXtWQV3V2UdbvRx1eHU9a
00401068	> 83E 00	CMPI DWORD PTR DS:[ESI],0	
0040106B	✓ 75 04	JNZ SHORT B643D2BD.00401071	
0040106D	✓ EB 0E	JMP SHORT B643D2BD.0040107D	
0040106F	✓ EB 0C	JMP SHORT B643D2BD.0040107D	
00401071	> 8B1E	MOV EBX, DWORD PTR DS:[ESI]	
00401073	• E8 97000000	CALL B643D2BD.0040110F	
00401078	• 83C6 04	ADD ESI,4	
0040107B	✓ EB EB	JMP SHORT B643D2BD.00401068	
0040107D	> 3D BF96287A	CMPI EAX,7A2896BF	
00401082	✓ 75 14	JNZ SHORT B643D2BD.00401098	
00401084	• 6A 40	PUSH 40	Style = MB_OK+MB_ICONASTERISK+MB_APPLMODAL
00401086	• 68 30354000	PUSH B643D2BD.00403530	Title = "In the Bin"
0040108B	• 68 30354000	PUSH B643D2BD.00403530	Text = "Congratulation, you found the right key"
00401090	• FF75 08	PUSH DWORD PTR SS:[EBP+8]	hOwner
00401093	• E8 02010000	CALL <JMP.&USER32.MessageBoxA>	MessageBoxA

GetDlgItemInt함수로 입력키를 Int형으로 받고 밑에서 0x7A2896BF(2049480383)와 비교한다.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000D10 36 6B 70 47 6C 7A 51 66 49 53 4D 6A 4D 2F 34 6A 6kpG1zQfISMjM/4j
00000D20 62 34 45 68 4F 71 69 71 00 00 00 00 78 56 34 12 b4EhOqiQ...xV4.
00000D30 49 6E 20 74 68 65 20 42 69 6E 00 43 6F 6E 67 72 In the Bin. Congr
00000D40 61 74 75 6C 61 74 69 6F 6E 2C 20 79 6F 75 20 66 atulation, you f
00000D50 6F 75 6E 64 20 74 68 65 20 72 69 67 68 74 20 6B ound the right k
```

시작 오프셋은 0D3B, 키 "2049480383"의 길이가 10이므로 끝은 0x0D45(0x0D3B+10)

정답 : 20494803830D3B0D45

Reverse L13 Start

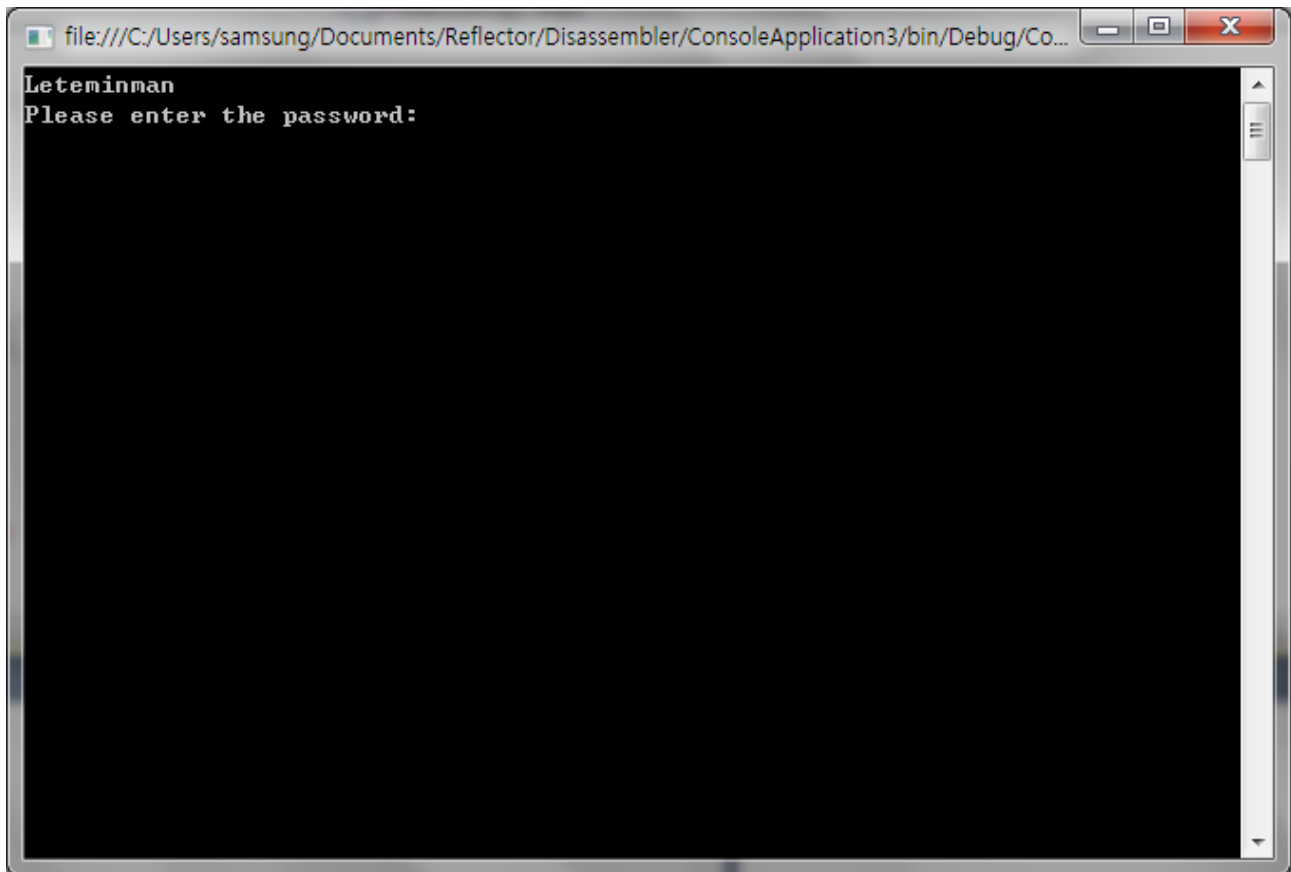
정답은 무엇인가

<http://codeengn.com/menu/challenges/reverse/13/2DBBC62F.exe>

먼저 .NET Reflector로 디컴파일하여 프로젝트를 추출했다.

```
string passPhrase = "^F79ejk56$Wx00a3";
string saltValue = "DHj47&+)$h";
string hashAlgorithm = "MD5";
int passwordIterations = 0x400;
string initVector = "&!Wx00a3$X^&+()CvHgE!";
int keySize = 0x100;
RijndaelSimple.Encrypt(plainText, passPhrase, saltValue, hashAlgorithm, passwordIterations, initVector, keySize);
plainText = RijndaelSimple.Decrypt(cipherText, passPhrase, saltValue, hashAlgorithm, passwordIterations, initVector, keySize);
Label_0056:
Console.WriteLine(plainText);
Console.WriteLine("Please enter the password: ");
if (Console.ReadLine() == plainText)
{
    Console.WriteLine("Well Done! You cracked it!");
    Console.ReadLine();
}
```

plainText변수에 복호화된 평문이 들어가게 되므로 password 입력전에 출력을 하도록 수정했다.



정답 : Leteminman

Reverse L14 Start

Name이 CodeEngn 일때 Serial을 구하시오

(이 문제는 정답이 여러개 나올 수 있는 문제이며 5개의 숫자로 되어있는 정답을 찾아야함, bruteforce 필요
Ex) 11111

<http://codeengn.com/menu/challenges/reverse/14/B0F24994.exe>

이름을 매개로 변환된 ESI 값과 키값을 매개로 변환된 EAX값을 비교한다.

키값은 정수형으로 변환되어 EAX에 놓여지기 때문에 변환된 ESI값만 구하면 된다.

0040132E	56	PUSH ESI	
0040132F	68 38314000	PUSH B0F24994.00403138	ASCII "1234567890"
00401334	E8 4A000000	CALL B0F24994.00401383	
00401339	5E	POP ESI	
0040133B	3BC6	CMP EAX,ESI	
0040133C	75 15	JNZ SHORT B0F24994.00401353	
0040133E	6A 00	PUSH 0	
00401340	68 62344000	PUSH B0F24994.00403462	ASCII "Key/CrackMe #2 "
00401345	68 B8344000	PUSH B0F24994.004034B8	ASCII " Good Job, I Wish You the Very Best"
0040134A	6A 00	PUSH 0	
0040134C	E8 9D000000	CALL B0F24994.004013EE	JMP to USER32.MessageBoxA
00401351	EB 13	JMP SHORT B0F24994.00401366	
00401353	6A 00	PUSH 0	
00401355	68 62344000	PUSH B0F24994.00403462	ASCII "Key/CrackMe #2 "
0040135A	68 86344000	PUSH B0F24994.00403486	ASCII " You Have Enter A Wrong Serial, Please Try Again "
0040135F	6A 00	PUSH 0	
00401361	E8 88000000	CALL B0F24994.004013EE	JMP to USER32.MessageBoxA

해당 비교구문에 BP를 걸어 ESI값을 확인해 보았다.

```

EAX 499602D2
ECX 00000000
EDX 00401339 ASCII "234567890"
EBX 00000037
ESP 0018FBA8
EBP 0018FBA8
ESI 000129A1
EDI 00000000
EIP 0040133A B0F24994.0040133A

```

ESI값이 0x000129A1(76193)인것을 알 수 있다.

정답 : 76193

Reverse L15 Start

Name이 CodeEngn 일때 Serial을 구하시오

<http://codeengn.com/menu/challenges/reverse/15/311F4179.exe>

00458831	. 3B05 44B84501	CMP EAX,DWORD PTR DS:[45B844]	
00458837	. 75 1B	JNZ SHORT 311F4179.00458854	
00458839	. B8 88884500	MOV EAX,311F4179.00458888	ASCII "You cracked the UBC CrackMe#1
0045883E	. E8 29C1FEFF	CALL 311F4179.0044496C	
00458843	. BA E8884500	MOV EDX,311F4179.004588E8	ASCII "CRACKED"
00458848	. A1 3CB84500	MOV EAX,DWORD PTR DS:[45B83C]	
0045884D	. E8 9ECDFCFF	CALL 311F4179.004255F0	
00458852	> EB 0A	JMP SHORT 311F4179.0045885E	
00458854	. B8 F8884500	MOV EAX,311F4179.004588F8	ASCII "Try Again !"
00458859	. E8 0EC1FEFF	CALL 311F4179.0044496C	

비교구문에 BP를 걸고 시리얼값으로 57005(0xDEAD)를 넣어보았다.

```

DS:[0045B844]=00006160
EAX=0000DEAD

```

비교값이 0x6160(24928)인것을 알 수 있다.

정답 : 24928

Reverse L16 Start

Name이 CodeEngn일때 Serial을 구하시오

<http://codeengn.com/menu/challenges/reverse/16/1F651B57.exe>

```

0040159F . 3B45 C4 CMP EAX,DWORD PTR SS:[EBP-3C]
004015A2 . 0F85 94000001 JNZ 1F651B57.0040163C
Stack SS:[0028FF0C]=E4C60D97
EAX=0000DEAD

```

이름에 CodeEngn을 입력하고 시리얼에 57005(0xDEAD)를 넣어보면 0xE4C60D97()과 비교하는것을 알 수 있다.

정답 : 3838184855

Reverse L17 Start

Key 값이 BEDA-2F56-BC4F4368-8A71-870B 일때 Name은 무엇인가

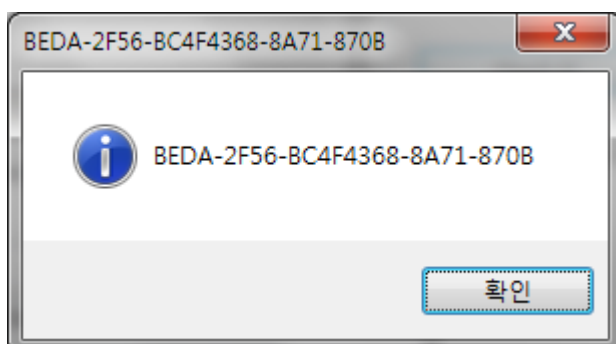
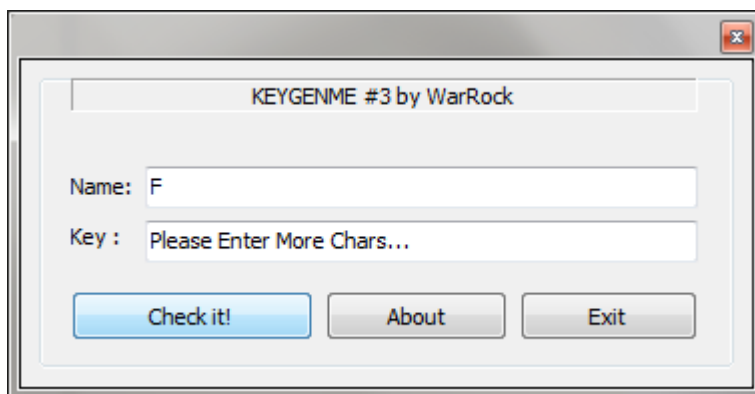
힌트 : Name은 한자리인데.. 알파벳일수도 있고 숫자일수도 있고..

정답인증은 Name의 MD5 해쉬값(대문자)

<http://codeengn.com/menu/challenges/reverse/17/E683EC0B.exe>

0045BB27	EB 15	JMP SHORT E683EC0B.0045BB3E	
0045BB29	BA 18BC4500	MOV EDX,E683EC0B.0045BC18	
0045BB2E	8B83 74030000	MOV EAX,DWORD PTR DS:[EBX+374]	ASCII "Please Enter More Chars..."
0045BB34	E8 6BE5F0FF	CALL E683EC0B.0043A0A4	
0045BB39	E9 91000000	JMP E683EC0B.0045BB3F	
0045BB3E	8D55 F4	LEA EDX,DWORD PTR SS:[EBP-C]	
0045BB41	8B83 68030000	MOV EAX,DWORD PTR DS:[EBX+368]	
0045BB47	E8 28E5F0FF	CALL E683EC0B.0043A074	
0045BB4C	8B45 F4	MOV EAX,DWORD PTR SS:[EBP-C]	
0045BB4F	8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	
0045BB52	8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]	
0045BB55	85C0	TEST EAX,EAX	
0045BB57	74 05	JE SHORT E683EC0B.0045BB5E	
0045BB59	83E8 04	SUB EAX,4	
0045BB5C	8B00	MOV EAX,DWORD PTR DS:[EAX]	
0045BB5E	83F8 1E	CMP EAX,1E	
0045BB61	7E 12	JLE SHORT E683EC0B.0045BB75	
0045BB63	BA 3CBC4500	MOV EDX,E683EC0B.0045BC3C	
0045BB68	8B83 74030000	MOV EAX,DWORD PTR DS:[EBX+374]	ASCII "Please Enter Not More Than 30 Chars..."
0045BB6E	E8 31E5F0FF	CALL E683EC0B.0043A0A4	
0045BB73	EB 5A	JMP SHORT E683EC0B.0045BB7F	
0045BB75	8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
0045BB78	8B83 74030000	MOV EAX,DWORD PTR DS:[EBX+374]	
0045BB7E	E8 F1E4F0FF	CALL E683EC0B.0043A074	
0045BB83	8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]	
0045BB86	50	PUSH EAX	
0045BB87	8D55 E8	LEA EDX,DWORD PTR SS:[EBP-18]	
0045BB8A	8B83 68030000	MOV EAX,DWORD PTR DS:[EBX+368]	
0045BB90	E8 DFE4F0FF	CALL E683EC0B.0043A074	
0045BB95	8B45 E8	MOV EAX,DWORD PTR SS:[EBP-18]	
0045BB98	8D55 EC	LEA EDX,DWORD PTR SS:[EBP-14]	
0045BB9B	E8 80FCFFFF	CALL E683EC0B.0045B850	
0045BBA0	8B55 EC	MOV EDX,DWORD PTR SS:[EBP-14]	
0045BBA3	58	POP EAX	
0045BBA4	E8 9390FAFF	CALL E683EC0B.00404C3C	
0045BBA9	90	NOP	
0045BBAA	90	NOP	
0045BBAB	6A 40	PUSH 40	
0045BBAD	B9 688E6200	MOV ECX,628E68	ASCII "68E2-2336-81C577AC-89C5-222D"
0045BBB2	BA 688E6200	MOV EDX,628E68	ASCII "68E2-2336-81C577AC-89C5-222D"
0045BBB7	A1 C0E94500	MOV EAX,DWORD PTR DS:[45E9C0]	

Name 길이를 한자리로 하기 위해서 계속 진행하고 체크후 메시징박스에 키를 출력하도록 수정하였다.



정답 : 800618943025315F869E4E1F09471012

Reverse L18 Start

Name이 CodeEngn일때 Serial은 무엇인가

<http://codeengn.com/menu/challenges/reverse/18/5AF8B382.exe>

004011E5	. 68 F0804000	PUSH 5AF8B382.004080F0	String2 = "06162370056B6AC0"
004011E9	. 68 F07E4000	PUSH 5AF8B382.00407EF0	String1 = "DEAD"
004011EF	. E8 DA000000	CALL <JMP.&kernel32.lstrcmpiA>	lstrcmpiA
004011F4	. 0BC0	OR EAX,EAX	
004011F6	. 74 16	JE SHORT 5AF8B382.0040120E	
004011F8	. 6A 10	PUSH 10	
004011FA	. 68 04664000	PUSH 5AF8B382.00406604	Style = MB_OK MB_ICONHAND MB_APPLMODAL
004011FF	. 68 E4654000	PUSH 5AF8B382.004065E4	Title = "Bad"
00401204	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	Text = "You serial is Wrong, try again"
00401207	. E8 E6000000	CALL <JMP.&user32.MessageBoxA>	hOwner
0040120C	. EB 5C	JMP SHORT 5AF8B382.0040126A	MessageBoxA
0040120E	. 6A 40	PUSH 40	
00401210	. 68 3C664000	PUSH 5AF8B382.0040663C	Style = MB_OK MB_ICONASTERISK MB_APPLMODAL
00401215	. 68 08664000	PUSH 5AF8B382.00406608	Title = "Good"
0040121A	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	Text = "Your serial is correct! now you know what 2 do :p"
0040121D	. E8 D0000000	CALL <JMP.&user32.MessageBoxA>	hOwner
00401222	. C9	LEAVE	MessageBoxA

이름에 CodeEngn을 입력하였고 lstrcmpiA 함수를 통해 값을 비교하는것을 알 수 있다.

정답 : 06162370056B6AC0

Reverse L19 Start

이 프로그램은 몇 밀리세컨드 후에 종료 되는가

<http://codeengn.com/menu/challenges/reverse/19/B5352594.exe>

00444C3E	. 8B3D 58D74700	MOV EDI,DWORD PTR DS:[&WINMM.timeGetTime]	WINMM.timeGetTime
00444C44	. FFD7	CALL EDI	<&WINMM.timeGetTime>
00444C46	. 803D 03E84800	CMP BYTE PTR DS:[48E8D3],0	
00444C4D	. 8BF0	MOV ESI,EAX	
00444C4F	. 0F84 FF000000	JE B5352594.00444D54	
00444C55	. 8B5C24 14	MOV EBX,DWORD PTR SS:[ESP+14]	
00444C59	. 8B2D 58D14700	MOV EBP,DWORD PTR DS:[&KERNEL32.Sleep]	kernel32.Sleep
00444C5F	. FFD7	CALL EDI	
00444C61	. 3BC6	CMP EAX,ESI	

timeGetTime함수를 호출하여 시간값을 구한다.

00444D38	. 2BC6	SUB EAX,ESI
00444D3A	. 3B43 04	CMP EAX,DWORD PTR DS:[EBX+4]
00444D3D	. 0F83 2EFFFFFF	JNB B5352594.00444C71
00444D43	. 6A 0A	PUSH 0A
00444D45	. FFD5	CALL EBP

현재시간과 이전시간의 차를 0x2B70(11120)과 비교한다.

정답 : 11120

Reverse L20 Start



이 프로그램은 Key파일을 필요로 하는 프로그램이다.

위 문구가 출력되도록 하려면 crackme3.key 파일안의 데이터는 무엇이 되어야 하는가

Ex) 41424344454647

<http://codeengn.com/menu/challenges/reverse/20/42564675.exe>

00401016	6A 00	PUSH 0	hTemplateFile = NULL
00401018	68 80000000	PUSH 80	Attributes = NORMAL
0040101D	6A 03	PUSH 3	Mode = OPEN_EXISTING
0040101F	6A 00	PUSH 0	pSecurity = NULL
00401021	6A 03	PUSH 3	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE
00401023	68 000000C0	PUSH C0000000	Access = GENERIC_READ GENERIC_WRITE
00401028	68 07204000	PUSH 42564675.004020D7	FileName = "CRACKME3.KEY"
0040102D	E8 76040000	CALL <JMP.&KERNEL32.CreateFileA>	CreateFileA
00401032	83F8 FF	CMP EAX,-1	
00401035	75 0C	JNZ SHORT 42564675.00401043	
00401037	68 0E214000	PUSH 42564675.0040210E	ASCII "CrackMe v3.0"
0040103C	E8 B4020000	CALL 42564675.004012F5	
00401041	EB 6B	JMP SHORT 42564675.004010AE	
00401043	A3 F5204000	MOV DWORD PTR DS:[4020F5],EAX	
00401048	B8 12000000	MOV EAX,12	
0040104D	BB 08204000	MOV EBX,42564675.00402008	
00401052	6A 00	PUSH 0	pOverlapped = NULL
00401054	68 A0214000	PUSH 42564675.004021A0	pBytesRead = 42564675.004021A0
00401059	50	PUSH EAX	BytesToRead => 12 (18.)
0040105A	53	PUSH EBX	Buffer => 42564675.00402008
0040105B	FF35 F5204000	PUSH DWORD PTR DS:[4020F5]	hFile = NULL
00401061	E8 30040000	CALL <JMP.&KERNEL32.ReadFile>	ReadFile
00401066	833D A0214000	CMP DWORD PTR DS:[4021A0],12	
0040106D	75 C8	JNZ SHORT 42564675.00401037	

"CRACKME3.KEY" 파일이 있으면 그 내용의 크기가 0x12(18)와 같은지 비교한다.

00401311	33C9	XOR ECX,ECX	kerne132.75813F0C
00401313	33C0	XOR EAX,EAX	
00401315	8B7424 04	MOV ESI,DWORD PTR SS:[ESP+4]	
00401319	B3 41	MOV BL,41	
0040131B	8A06	MOV AL,BYTE PTR DS:[ESI]	
0040131D	32C3	XOR AL,BL	
0040131F	8B06	MOV BYTE PTR DS:[ESI],AL	
00401321	46	INC ESI	
00401322	FEC3	INC BL	
00401324	0105 F9204000	ADD DWORD PTR DS:[4020F9],EAX	
0040132A	3C 00	CMP AL,0	
0040132C	74 07	JE SHORT 42564675.00401335	
0040132E	FEC1	INC CL	
00401330	80FB 4F	CMP BL,4F	
00401333	75 E6	JNZ SHORT 42564675.0040131B	
00401335	890D 49214000	MOV DWORD PTR DS:[402149],ECX	

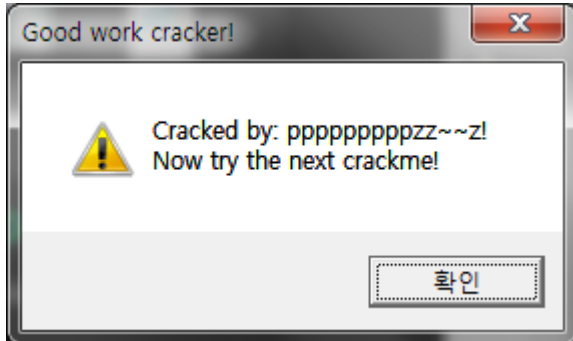
위는 BL에 0x41(65)값을 넣고 문자열의 각 값과 xor을 하고 BL을 증가시키는 루프이다.

0040106F	68 08204000	PUSH 42564675.00402008	ASCII "pppppppppz~z5678"
00401074	E8 98020000	CALL 42564675.00401311	
00401079	8135 F9204000	XOR DWORD PTR DS:[4020F9],12345678	
00401083	83C4 04	ADD ESP,4	
00401086	68 08204000	PUSH 42564675.00402008	ASCII "pppppppppz~z5678"
0040108B	E8 AC020000	CALL 42564675.0040133C	
00401090	83C4 04	ADD ESP,4	
00401093	3B05 F9204000	CMP EAX,DWORD PTR DS:[4020F9]	

키파일에 "123456789012345678"을 넣었더니 "ppppppppppzz~z5678"가 되었고 0x00401093에서 호출된 함수의 리턴값과 한 값을 비교하는것을 알 수 있다.

```
DS:[004020F9]=12345022
EAX=38373635
```

EAX값이 38373635인것으로 보아 마지막 뒤 4자리의 DWORD값인것을 알 수 있고 마지막 4바이트를 0x12345022("Wx22Wx50Wx34Wx12")로 바꾸어 키파일에 저장해 보았다.



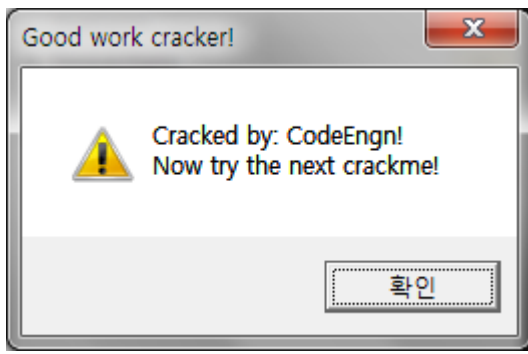
성공적으로 메시지가 떴으나 "Crack by: CodeEngn!" 으로 나타나지 않았으므로 CodeEngn이 되기전 값으로 키파일을 만들었다.

```
>>> text='CodeEngnWx00Wx00Wx00Wx00Wx00Wx00Wx00Wx00Wx00'
>>> ret=""
>>> i=0x41
>>> for ch in text:
...   ret+=chr(ord(ch)^i)
...   i+=1
...
>>> ret
"Wx02-'!Wx00( &IJKLMNOPQR"
>>> f=open('crackme3.key', 'wb')
>>> f.write(ret)
>>> f.close()
```

```
00401090 . 83C4 04      ADD ESP,4
00401093 . 3B05 F9204001 CMP EAX,DWORD PTR DS:[4020F9]
```

```
DS:[004020F9]=1234557B
EAX=5251504F
```

마지막 4바이트를 0x1234557B와 비교하기 때문에 다시 마지막 값을 0x1234557B("Wx7BWx55Wx34Wx12")로 바꾸어 키파일에 저장하고 프로그램을 실행 해 보았다.



정답 : 022D272100282026494A4B4C4D4E7B553412