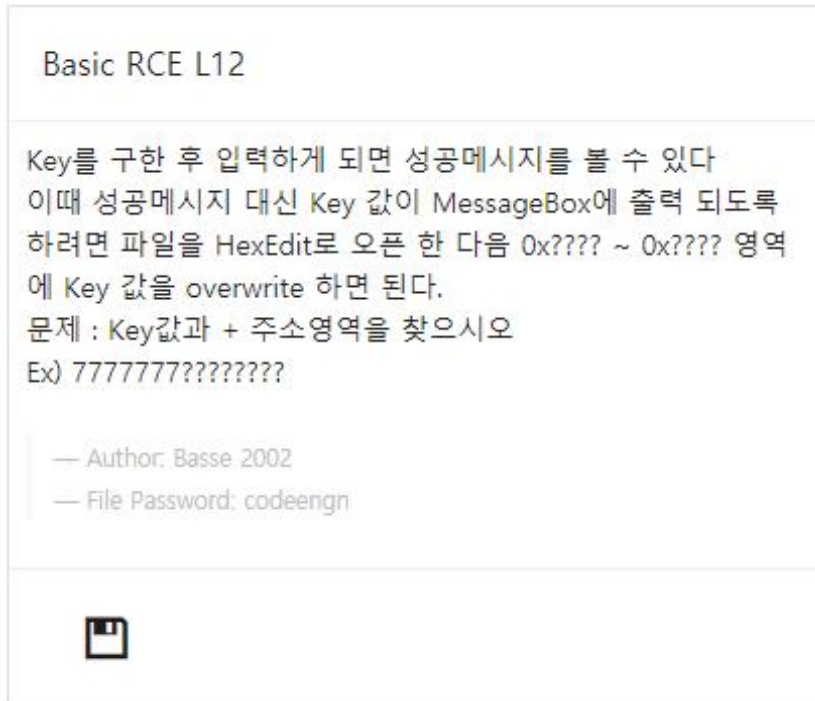


2019.02.16. CodeEngn Basic RCE L12

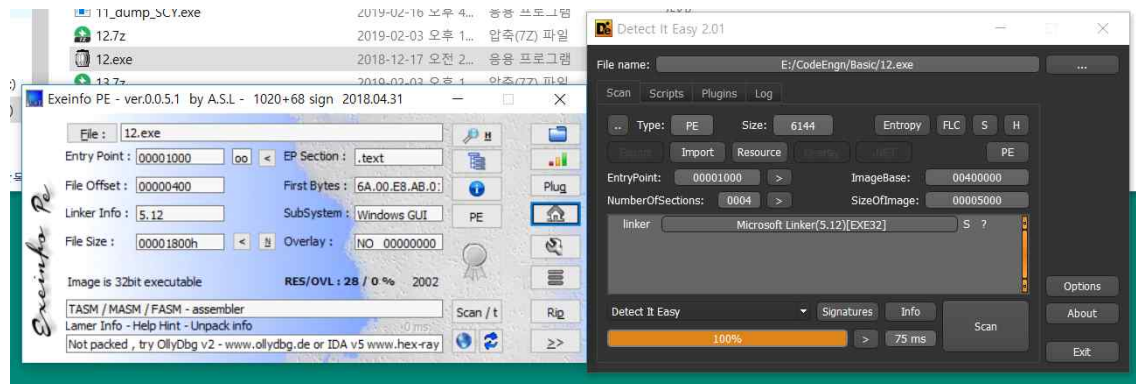
Tree to Tree



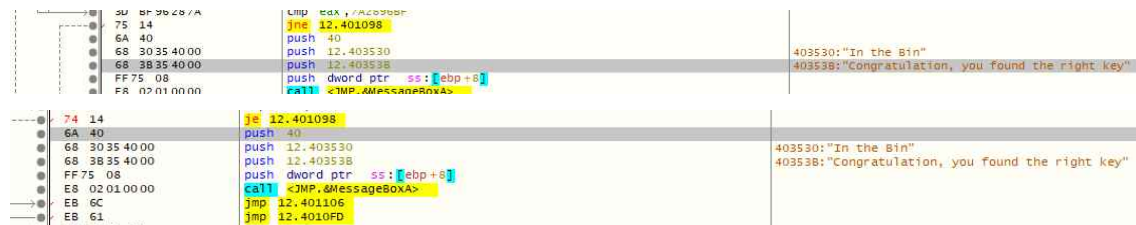
Key값을 찾고 Hexedit으로 띄워서 주소를 찾으라고 한다. 일반적인 실행시 check무반응



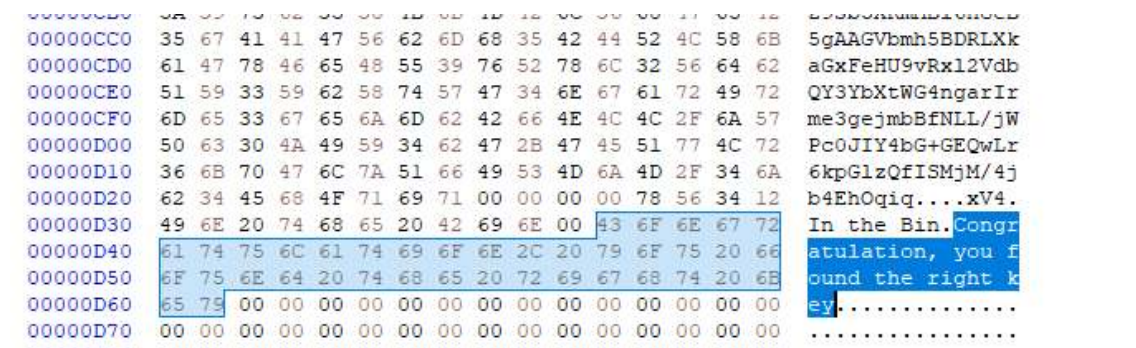
패킹이 되어있지 않다.



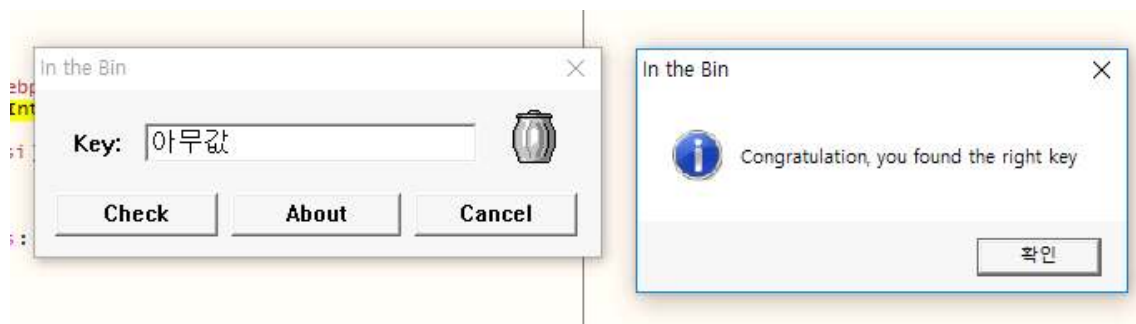
성공했을 때 나오는 문구를 캐치하기 위하여 코드를 따라가다 발견



간단한 우회로 성공할 수 있지만 문제는 코드값을 찾아야한다. 저부분이 출력된다.



0D3B~0D62 까지다.



유형	주소	Module/Label/Exception	상태	디스어셈블리	Hits	Summary
소프트웨어	0040105E	12.exe	정상화일	call <JMP.&GetDlgItemInt>	0	
	0040107D	12.exe	정상화일	cmp eax, 7A2896BF	0	
	746337EB	user32.dll	정상화일	call <user32.GetDlgItemTextW>	0	
	74633854	user32.dll	정상화일	call dword ptr ds:[<&FoldStringW>]	0	

트레이싱하면서 중점으로 봤던 함수들

GetDlgItemTextW 등등

들어가보니 문자를 하나하나 다시 재저장하는 듯하다.

7463384F	68 80000000	push 80			
74633854	FF 15 2C B4 68 74	call dword ptr ds:[<&FoldStringW>]			
7463385A	33 C0	xor eax, eax			
7463385C	8D 95 3C FF FF FF	lea edx, dword ptr ss:[ebp-C4]			
74633862	88 C8	mov ecx, eax			
74633864	0F B7 85 3C FF FF FF	movzx eax, word ptr ss:[ebp-C4]			
7463386B	89 85 38 FF FF FF	mov dword ptr ss:[ebp-C8], eax			
74633871	85 C0	test eax, eax			
74633873	74 33	jz user32.746338A8			
74633875	83 C2 02	add edx, 2			edx:L"345"
74633878	89 95 2C FF FF FF	mov dword ptr ss:[ebp-D4], edx			[ebp-D4]:L"2345"
7463387E	8D 5D D0	lea edx, dword ptr ds:[eax-30]			edx:L"345"
74633881	85 D2	test edx, edx			edx:L"345"
74633883	78 23	jle user32.746338A8			
74633885	83 FA 09	cmp edx, 9			edx:L"345", 9:'\t'
74633888	7F 1E	jg user32.746338A8			
7463388A	3B CE	cmp ecx, esi			
7463388C	0F 85 3A C4 02 00	jbe user32.7465FCCC			
74633892	68 C9 0A	imul ecx, ecx, 4			
74633895	89 9D 28 FF FF FF	mov dword ptr ss:[ebp-D8], ebx			
74633898	03 CA	add ecx, edx			edx:L"345"
7463389D	88 95 2C FF FF FF	mov edx, dword ptr ss:[ebp-D4]			[ebp-D4]:L"2345"
746338A3	0F B7 02	movzx eax, word ptr ds:[edx]			edx:L"345"
746338A6	EB C3	jmp user32.7463386B			
746338A8	83 BD 34 FF FF FF 00	cmp dword ptr ss:[ebp-CC], 0			
746338AF	75 31	jne user32.746338E2			
746338B1	85 FF	test edi, edi			

그 후 엄청 긴 문자열들을 작업하는데

0040105E	FF 75 08	push dword ptr ss:[ebp+8]			
00401063	58 31 01 00 00	call <JMP.&GetDlgItemInt>			
00401068	BE 00 30 40 00	mov esi, 12.403000			esi:"bhsuvbLRO6BmfZnNjrhakG681WcuJ93rN7bS"
0040106D	83 3E 00	cmp dword ptr ds:[esi], 0			esi:"bhsuvbLRO6BmfZnNjrhakG681WcuJ93rN7bS"
0040106F	75 04	jnz 12.401071			
00401070	EB 0E	jmp 12.40107D			
00401071	8B 1E	mov ebx, dword ptr ds:[esi]			esi:"bhsuvbLRO6BmfZnNjrhakG681WcuJ93rN7bS"
00401073	E8 97 00 00 00	call 12.40110F			esi:"bhsuvbLRO6BmfZnNjrhakG681WcuJ93rN7bS"
00401078	83 C6 04	add esi, 4			
0040107C	EB 08	jmp 12.401068			
0040107E	3D BF 96 28 7A	cmp eax, 7A2896BF			
00401083	75 14	jnz 12.401098			
00401084	6A 40	push 40			
00401086	68 30 35 40 00	push 12.403530			403530:"In the Bin"
00401088	68 38 35 40 00	push 12.403538			403538:"Congratulation, you found the right"
00401093	E8 02 01 00 00	push dword ptr ss:[ebp+8]			
00401098	EB 6C	call <JMP.&SendMessageA>			
0040109A	EB 61	jmp 12.401106			
0040109C	66 83 F8 02	cmp ax, 1			
004010A0	75 10	jnz 12.401082			
004010A2	6A 00	push 0			
004010A4	6A 00	push 0			
004010A6	6A 10	push 10			
004010A8	FF 75 08	push dword ptr ss:[ebp+8]			
004010AB	E8 F0 00 00 00	call <JMP.&SendMessageA>			
004010B0	EB 4B	jmp 12.4010FD			
004010B2	66 3D 88 0B	cmp ax, 883			
004010B6	75 45	jnz 12.4010FD			
004010B8	6A 00	push 0			

cmp eax, 7A2896BF할 때 eax값을 유심히 보니

0040105E	75 6A	jnz 12.40109C			
0040105F	6A 00	push 0			
00401064	6A 00	push 0			
00401068	68 89 08 00 00	push 889			
0040106B	FF 75 08	push dword ptr ss:[ebp+8]			
0040106E	58 31 01 00 00	call <JMP.&GetDlgItemInt>			
00401073	BE 00 30 40 00	mov esi, 12.403000			
00401078	83 C6 04	add esi, 4			
0040107C	EB 08	jmp 12.401068			
0040107E	3D BF 96 28 7A	cmp eax, 7A2896BF			
00401083	75 14	jnz 12.401098			
00401084	6A 40	push 40			
00401086	68 30 35 40 00	push 12.403530			403530:"In the Bin"
00401088	68 38 35 40 00	push 12.403538			403538:"Congratulation, you found the right"

계산기

≡

프로그래머

12,345

HEX 3039

DEC 12,345

OCT 30 071

BIN 0011 0000 0011 1001

QWORD MS M\*

Lsh	Rsh	Or	Xor	Not	And
↑	Mod	CE	C	⊗	÷
A	B	7	8	9	×
C	D	4	5	6	−
E	F	1	2	3	+
(	)	±	0	.	=

FPU 숨기기

EAX	00003039	
EBX	00000030	'0'
ECX	67DD27DE	
EDX	0019F7F2	
EBP	0019F8C4	
ESP	0019F8C4	
ESI	00403528	12.00403
EDI	80006010	
EIP	0040107D	12.00401

EFLAGS 00000344

ZF 1

PF 1

AF 0

OF 0

SF 0

DF 0

CF 0

TF 1

IF 1

LastError 00000000 (ERROR\_S

LastStatus C0000034 (STATUS\_

GS 002B FS 0053

ES 002B DS 002B

CS 0023 SS 002B

ST(0) 00000000000000000000

ST(1) 00000000000000000000

ST(2) 00000000000000000000

ST(3) 00000000000000000000

ST(4) 00000000000000000000

ST(5) 3FFF8000000000000000

ST(6) 3FFF8000000000000000

ST(7) 3FFF8000000000000000

x87TagWord FFFF

x87TW\_0 3 (비어 있음) x87TW\_1

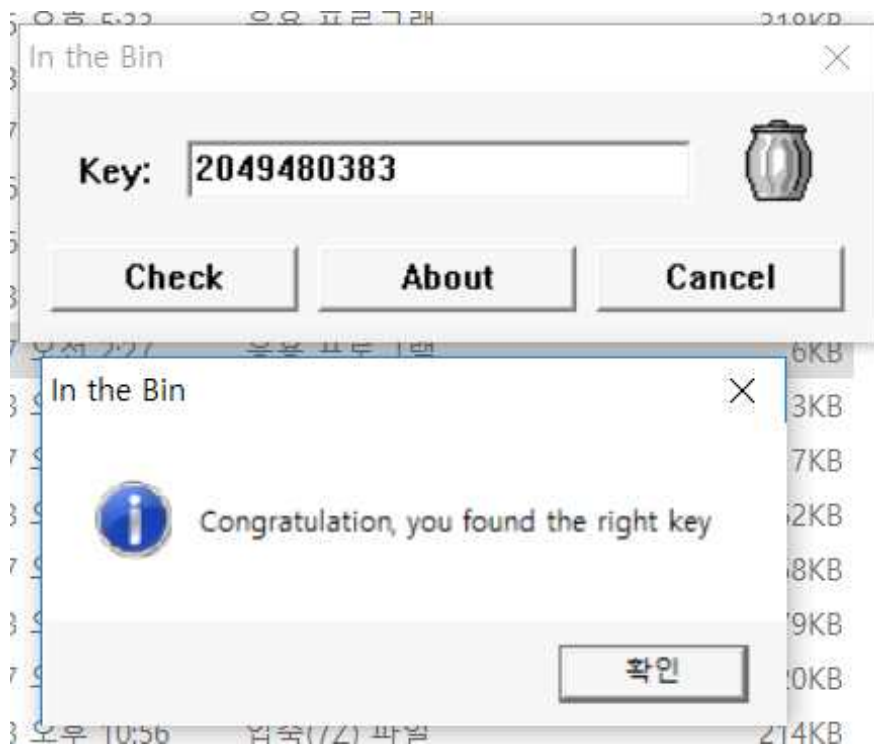
x87TW\_2 3 (비어 있음) x87TW\_3

3039로 트레이싱 할 때 입력했던 12345문자열을 16진수로 바꾼 값



결국 key값은 2049480383





2049480383

00000D30	49 6E 20 74 68 65 20 42 69 6E 00 32 30 34 39 34	In the Bin. 20494
00000D40	38 30 33 38 33 00 00 00 00 00 00 00 00 00 00	80383.....
00000D50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000D60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000D70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000D80	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

문자열이 들어가는 자리는

총 10자리 +null = 11자리


0D3B부터 ~ 0D45

In the Bin

Key:



In the Bin

 2049480383

조합하면

2049480383 + 0D3B0D45

20494803830D3B0D45

Clear