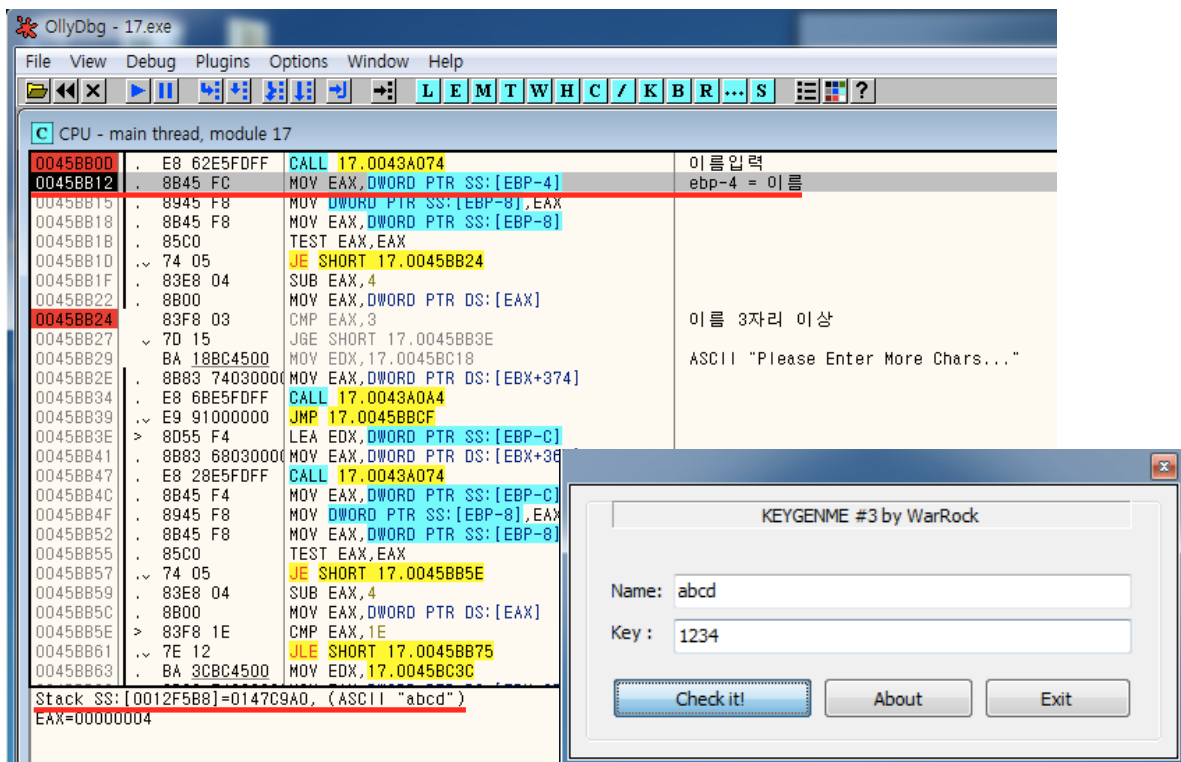
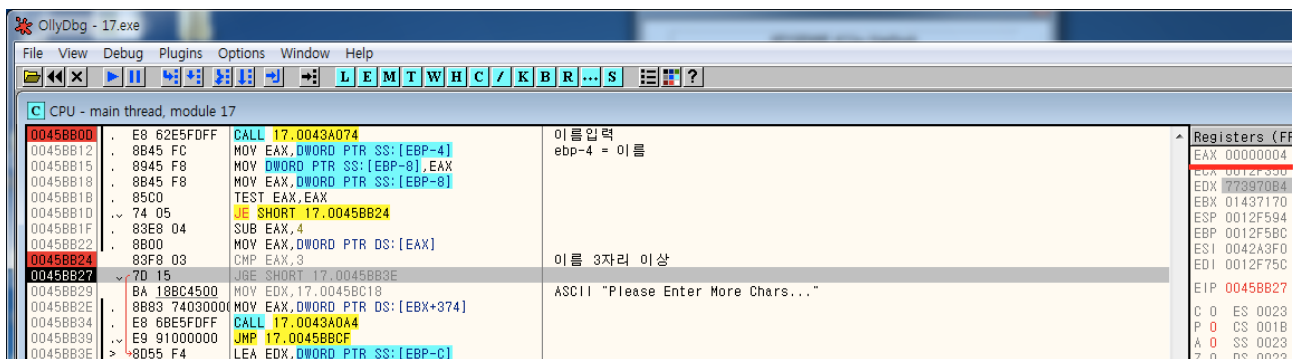


베이직 17번과 같은 문제입니다. 이름과 키를 입력받고 인증하는 프로그램입니다.

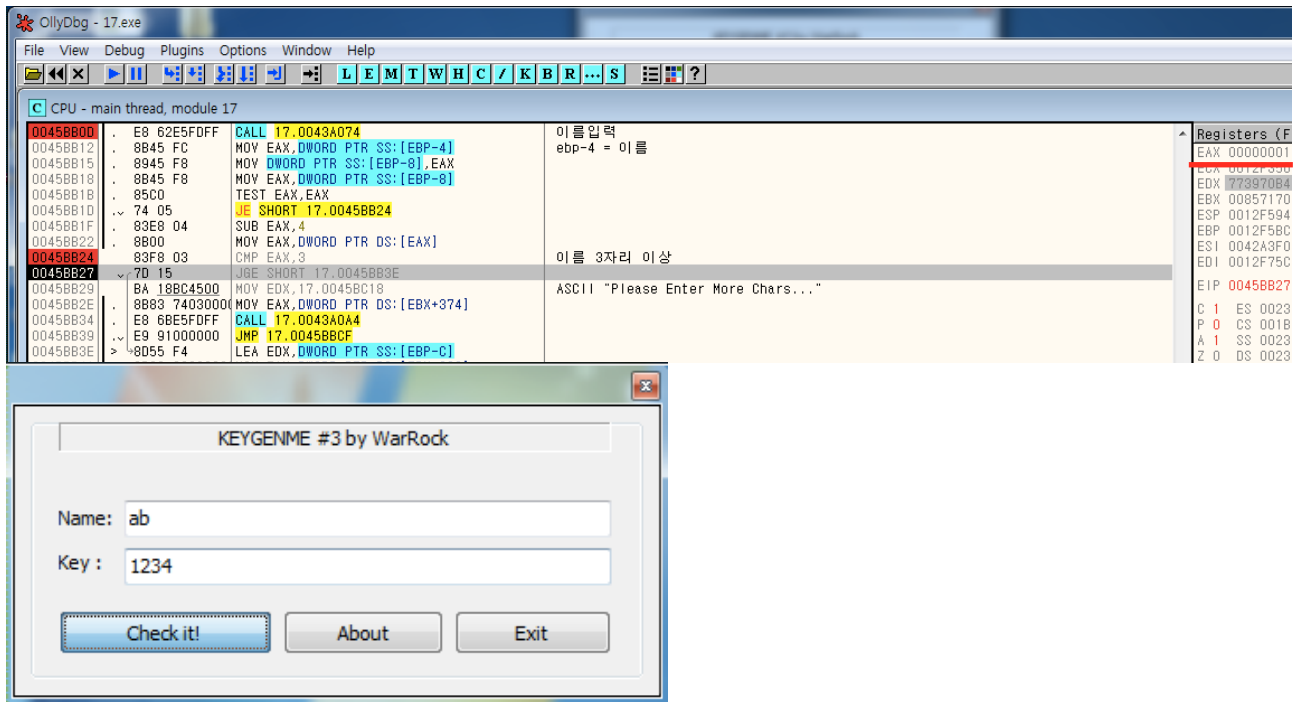


0045BB0D 주소의 함수가 이름을 불러오는 함수이고 ebp-4에 그 이름이 저장됨을 알 수 있습니다.

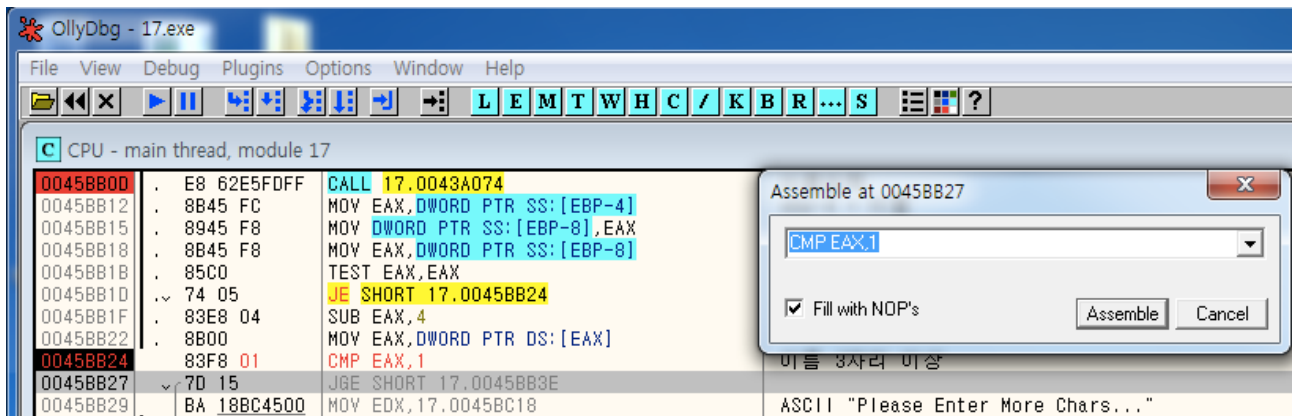


주석 부분을 보시면 이름의 글자수를 EAX에 받아와 비교합니다.

글자수가 3자리 이상이면 이 구문은 그냥 넘어가게 됩니다.



한 글자만 입력하면 EAX에 1이 들어가고 키 텍스트박스에는 이름을 더 길게 입력하라는 문구가 출력됩니다.



문제의 힌트에서 이름의 길이가 20이라고 했으니 비교구문의 3을 2로 수정합니다.

Address	Disassembly	Comment
0045BB8A	MOV EAX, DWORD PTR DS:[EBX+368]	
0045BB90	CALL 17.0043A074	
0045BB95	MOV EAX, DWORD PTR SS:[EBP-18]	
0045BB98	LEA EDX, DWORD PTR SS:[EBP-14]	
0045BB9B	CALL 17.0045B850	패스워드 만들어진다
0045BBA0	MOV EDX, DWORD PTR SS:[EBP-14]	
0045BBA3	POP EAX	
0045BBA4	CALL 17.00404C3C	비교
0045BBA9	JNZ SHORT 17.0045B8C5	
0045BBAB	PUSH 40	
0045BBAD	MOV ECX, 17.0045B8C4	ASCII "Good Boy!!!"
0045BBB2	MOV EDX, 17.0045B8C7	ASCII "Well done!"
0045BBB7	MOV EAX, DWORD PTR DS:[45E9C0]	
0045BBBC	MOV EAX, DWORD PTR DS:[EAX]	
0045BBBE	CALL 17.0045B8C7	
0045BBB3	JMP SHORT 17.0045B8C5	
0045BBB5	PUSH 140	
0045BBB8	CALL <JMP.&kernel32.Sleep>	[Timeout = 333. ms Sleep

BreakPoint가 지정되어 있는 두 주소에서 패스워드가 만들어지고 입력한 패스워드와 비교하게 됩니다.

Address	Disassembly	Comment
0045B885	MOV DWORD PTR SS:[EBP-10], EAX	
0045B888	MOV EAX, DWORD PTR SS:[EBP-4]	
0045B88B	TEST EAX, EAX	
0045B88D	JE SHORT 17.0045B894	
0045B88F	SUB EAX, 4	
0045B892	MOV EAX, DWORD PTR DS:[EAX]	
0045B894	TEST EAX, EAX	
0045B896	JLE SHORT 17.0045B8C4	
0045B898	MOV ECX, 1	
0045B89D	MOV EBX, DWORD PTR SS:[EBP-4]	
0045B8A0	MOVZX ESI, BYTE PTR DS:[EBX+ECX-1]	esi에 한글자씩
0045B8A5	ADD ESI, EDX	
0045B8A7	IMUL ESI, ESI, 772	한글자 * 772
0045B8AD	MOV EDX, ESI	
0045B8AF	IMUL EDX, ESI	결과2 = 결과1 * 결과1
0045B8B2	ADD ESI, EDX	결과2 + 결과1
0045B8B4	OR ESI, ESI	
0045B8B6	IMUL ESI, ESI, 474	*474
0045B8B8	ADD ESI, ESI	
0045B8BE	MOV EDX, ESI	
0045B8C0	INC ECX	
0045B8C1	DEC EAX	
0045B8C2	JNZ SHORT 17.0045B89D	
0045B8C4	MOV EAX, DWORD PTR SS:[EBP-4]	
0045B8C7	TEST EAX, EAX	
0045B8C9	JE SHORT 17.0045B8D0	

패스워드가 만들어지는 루틴입니다.

주석을 보면 알고리즘이 어떻게 이루어져 있는지 알 수 있습니다.

이 루틴은 이름 문자열의 길이만큼 반복합니다.

밑에 내려가면 더 많은 루틴이 있고 각각 특정 문자열을 만들어 냅니다.

CPU - main thread, module 08		
0045B9BE	. E8 05CDFAFF	CALL 08.004086C8
0045B9C3	. 8B45 E0	MOV EAX,DWORD PTR SS:[EBP-20]
0045B9C6	. B9 04000000	MOV ECX,4
0045B9CB	. BA 01000000	MOV EDX,1
0045B9D0	. E8 3793FAFF	CALL 08.00404D0C
0045B9D5	. FF75 E4	PUSH DWORD PTR SS:[EBP-1C]
0045B9D8	. 68 E4BA4500	PUSH 08.0045BAE4
0045B9DD	. 8D45 DC	LEA EAX,DWORD PTR SS:[EBP-24]
0045B9E0	. 50	PUSH EAX
0045B9E1	. 8D4D D8	LEA ECX,DWORD PTR SS:[EBP-28]
0045B9E4	. 33D2	XOR EDX,EDX
0045B9E6	. 8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]
0045B9E9	. E8 DACCFAFF	CALL 08.004086C8
0045B9EE	. 8B45 D8	MOV EAX,DWORD PTR SS:[EBP-28]
0045B9F1	. B9 04000000	MOV ECX,4
0045B9F6	. BA 01000000	MOV EDX,1
0045B9FB	. E8 0C93FAFF	CALL 08.00404D0C
0045BA00	. FF75 DC	PUSH DWORD PTR SS:[EBP-24]
0045BA03	. 68 E4BA4500	PUSH 08.0045BAE4
0045BA08	. 8D45 D4	LEA EAX,DWORD PTR SS:[EBP-2C]
0045BA0B	. 50	PUSH EAX
0045BA0C	. B9 08000000	MOV ECX,8
0045BA11	. BA 01000000	MOV EDX,1
0045BA16	. 8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
0045BA19	. E8 EE92FAFF	CALL 08.00404D0C
0045BA1E	. FF75 D4	PUSH DWORD PTR SS:[EBP-2C]
0045BA21	. 68 E4BA4500	PUSH 08.0045BAE4
Stack SS:[0012F568]=0172C6B8, (ASCII "5CBAC620")		
EAX=0012F568		

위에서 만들어진 문자열을 받아옵니다.

CPU - main thread, module 08		
0045B9BE	. E8 05CDFAFF	CALL 08.004086C8
0045B9C3	. 8B45 E0	MOV EAX,DWORD PTR SS:[EBP-20]
0045B9C6	. B9 04000000	MOV ECX,4
0045B9CB	. BA 01000000	MOV EDX,1
0045B9D0	. E8 3793FAFF	CALL 08.00404D0C
0045B9D5	. FF75 E4	PUSH DWORD PTR SS:[EBP-1C]
0045B9D8	. 68 E4BA4500	PUSH 08.0045BAE4
0045B9DD	. 8D45 DC	LEA EAX,DWORD PTR SS:[EBP-24]
0045B9E0	. 50	PUSH EAX
0045B9E1	. 8D4D D8	LEA ECX,DWORD PTR SS:[EBP-28]
0045B9E4	. 33D2	XOR EDX,EDX
0045B9E6	. 8B45 F0	MOV EAX,DWORD PTR SS:[EBP-10]
0045B9E9	. E8 DACCFAFF	CALL 08.004086C8
0045B9EE	. 8B45 D8	MOV EAX,DWORD PTR SS:[EBP-28]
0045B9F1	. B9 04000000	MOV ECX,4
0045B9F6	. BA 01000000	MOV EDX,1
0045B9FB	. E8 0C93FAFF	CALL 08.00404D0C
0045BA00	. FF75 DC	PUSH DWORD PTR SS:[EBP-24]
0045BA03	. 68 E4BA4500	PUSH 08.0045BAE4
0045BA08	. 8D45 D4	LEA EAX,DWORD PTR SS:[EBP-2C]
0045BA0B	. 50	PUSH EAX
0045BA0C	. B9 08000000	MOV ECX,8
0045BA11	. BA 01000000	MOV EDX,1
0045BA16	. 8B45 F8	MOV EAX,DWORD PTR SS:[EBP-8]
0045BA19	. E8 EE92FAFF	CALL 08.00404D0C
0045BA1E	. FF75 D4	PUSH DWORD PTR SS:[EBP-2C]
0045BA21	. 68 E4BA4500	PUSH 08.0045BAE4
Stack SS:[0012F56C]=0077C6D0, (ASCII "5CBA")		

그리고 그 문자열 앞의 네자리를 뽑아냅니다.

이렇게 뽑아낸 글자들을 다 연결한 것이 최종 키가 됩니다.

CPU - main thread, module 08			
00458BA0	. 8B55 EC	MOV EDX,DWORD PTR SS:[EBP-14]	
00458BA3	. 58	POP EAX	
00458BA4	. E8 9390FAFF	CALL 08.00404C3C	비교
00458BA9	~ 75 1A	JNZ SHORT 08.00458BC5	
00458BAB	. 6A 40	PUSH 40	
00458BAD	. B9 64BC4500	MOV ECX,08.00458C64	ASCII "Good Boy!!!"
00458BB2	. BA 70BC4500	MOV EDX,08.00458C70	ASCII "Well done!"
00458BB7	. A1 C0E94500	MOV EAX,DWORD PTR DS:[45E9C0]	
00458BBC	. 8B00	MOV EAX,DWORD PTR DS:[EAX]	
00458BBE	. E8 B5D0FFFF	CALL 08.00458C78	
00458BC3	~ EB 0A	JMP SHORT 08.00458BCF	
00458BC5	> 68 4D010000	PUSH 140	[Timeout = 333. ms
00458BCA	. E8 6514FBFF	CALL <JMP.&kernel32.Sleep>	Sleep
00458BCF	> 33C0	XOR EAX,EAX	
00458BD1	. 5A	POP EDX	
00458BD2	. 59	POP ECX	
00458BD3	. 59	POP ECX	
00458BD4	. 64:8910	MOV DWORD PTR FS:[EAX],EDX	
00458BD7	. 68 09BC4500	PUSH 08.00458C09	
00458BDC	> 8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
00458BDF	. E8 308CFAFF	CALL 08.00404814	
00458BE4	. 8D45 EC	LEA EAX,DWORD PTR SS:[EBP-14]	
00458BE7	. E8 288CFAFF	CALL 08.00404814	
00458BEC	. 8D45 F0	LEA EAX,DWORD PTR SS:[EBP-10]	
00458BEF	. BA 02000000	MOV EDX,2	
00458BF4	. E8 3F8CFAFF	CALL 08.00404838	
00458BF9	. 8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
Stack SS:[0012F5A8]=00798E38, (ASCII "5CBA-7E6C-5D11A563-EFDD-CEC4")			
EDX=007750E8, (ASCII "ab")			

그림을 보시면 패스워드가 보이고 처음 4글자가 아까 뽑아낸 4글자인 것을 확인할 수 있습니다.

이름을 이용한 알고리즘으로 패스워드를 만들기 때문에 프로그래밍을 통해 원하고자 하는 값을 찾을 수 있습니다.

```

3844 0x39, 0x70,
3845 0x39, 0x77,
3846 0x39, 0x78,
3847 0x39, 0x79,
3848 0x39, 0x7a,
3849 0x39, 0x30,
3850 0x39, 0x31,
3851 0x39, 0x32,
3852 0x39, 0x33,
3853 0x39, 0x34,
3854 0x39, 0x35,
3855 0x39, 0x36,
3856 0x39, 0x37,
3857 0x39, 0x38,
3858 0x39, 0x39,
3859 };
3860
3861 int esi,edx,i,j;
3862
3863 //b=esi
3864 //c=edx
3865
3866 for(i=0;i<1452;i++)
3867 {
3868     for(j=0;j<2;j++)
3869     {
3870         esi=a[i][j];
3871         esi=esi+edx;
3872         esi=esi*0x772;
3873         edx=esi;
3874         edx=edx*esi;
3875         esi=esi+edx;
3876         esi=esi*0x474;
3877         esi=esi+esi;
3878         edx=esi;
3879     }
3880     printf("%c%c : %x\n",a[i][0],a[i][1],edx);
3881     edx=0;
3882 }
3883
3884 }
3885
3886 return 0;
3887 }

```

조건에 만족되는 모든 HEX값을 알고리즘에 적용시키는 프로그램입니다.

```
areong — bash — 80x24
Cu : d5b5c530
Cv : ca56e660
Cw : 9a5d0ad0
Cx : 45c83280
Cy : cc985d70
Cz : 2ecd8ba0
C0 : 94317c00
C1 : 6698bcf0
C2 : 14650120
C3 : 9d964890
C4 : 22c9340
C5 : 4227e130
C6 : 5d883260
C7 : 544d86d0
C8 : 2677de80
C9 : d4073970
DA : 5f2848f0
DB : 31df7d20
DC : dffbb490
DD : 697cef40
DE : ce632d30
DF : eae6e60
DG : 2a5eb2d0
DH : 2173fa80
```

이렇게 알고리즘이 적용된 글자들을 확인할 수 있고 조건에 맞는 문자를 MD5한것이 정답입니다.