# Peach's Crackme (solution tutorial by @Danofred0)

http//crackmes.one

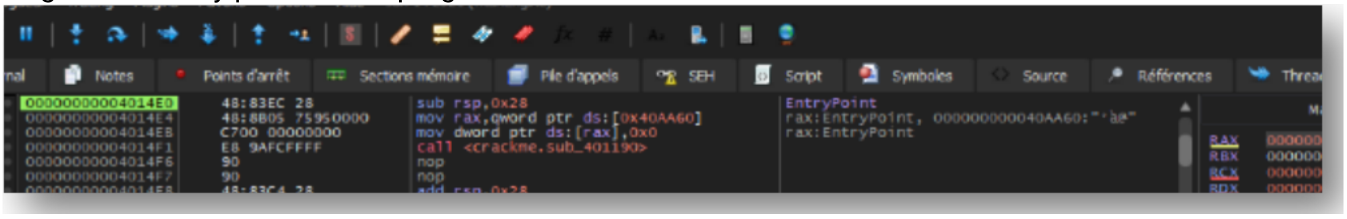
## I. Analyse

Get start by launch the crackme, try many username and password to see how does it work. Then, continue by using Detect It Easy (die) to get information about this crackme.

- ➔ Entry point : 0x00000000004014e0
- ➔ mode : CLI
- ➔ arch : x64

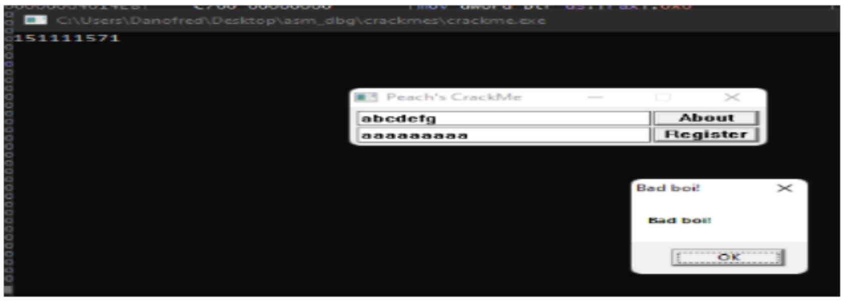
## II. Disassemble this using x64dbg (or another debugger)

Let's go at the entry point of this program.



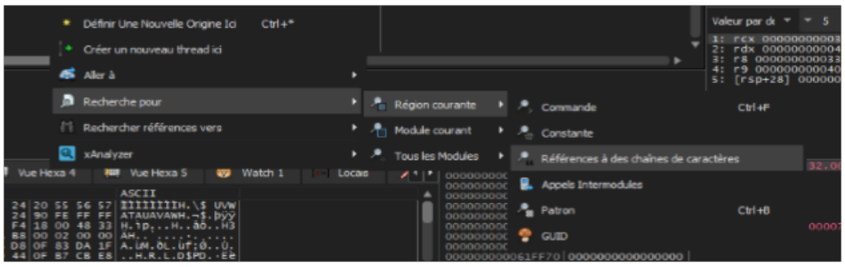
```
00000000004014E0  48:83EC 28         sub rsp,0x28                           EntryPoint
00000000004014E4  48:8B05 75950000   mov rax,qword ptr ds:[0x40AA60]        rax:EntryPoint, 000000000040AA60:"`àê"
00000000004014EB  C700 00000000      mov dword ptr ds:[rax],0x0             rax:EntryPoint
00000000004014F1  E8 9AFCFFFF        call <crackme.sub_401190>
00000000004014F6  90                 nop
00000000004014F7  90                 nop
00000000004014F8  48:83C4 28         add rsp,0x28
```

If you press F8 03 times, you can see that the program is so running at the first call ' call <crackme.sub_401190> '. This function is called inside the main function. Then i'm trying 'abcdefg' like a username and 'aaaaaaaaa' like a password. I'm see that the program call the message box with the bad message «Bad boi!».

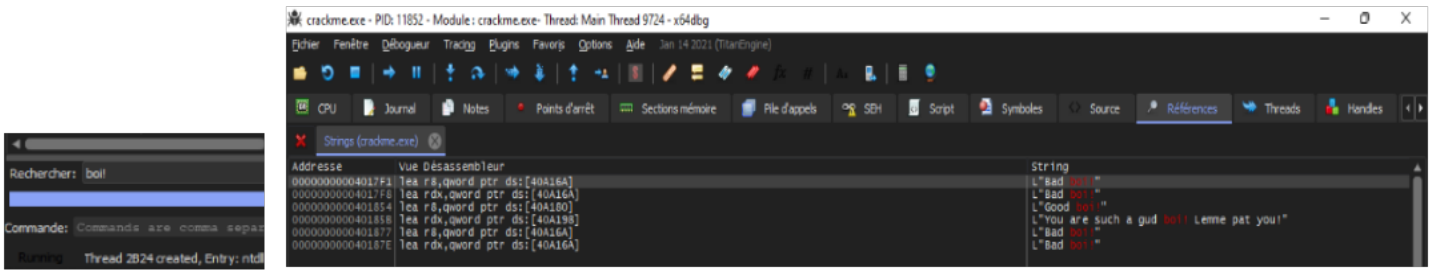


```
C:\Users\Danofred\Desktop\asm_dbg\crackmes\crackme.exe
151111571
```

Now, i'm just try to find this bad message inside all the strings that are present inside the crackme by doing :
Right Click >> Search For >> Current module >> Strings References



Then just filter the strings by entry the bad message inside the filter



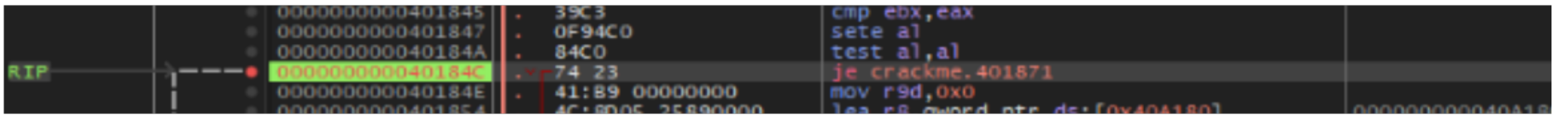
crackme.exe - PID: 11852 - Module: crackme.exe - Thread: Main Thread 9724 - x64dbg
Rechercher: boi!
Strings (crackme.exe)

```
Addresse          Vue Désassembleur                  String
00000000004017F1  lea r8,qword ptr ds:[40A16A]       L"Bad boi!"
00000000004017F8  lea rdx,qword ptr ds:[40A16A]      L"Bad boi!"
0000000000401854  lea r8,qword ptr ds:[40A180]       L"Good boi!"
000000000040185B  lea rdx,qword ptr ds:[40A198]      L"You are such a gud boi! Lemme pat you!"
0000000000401877  lea r8,qword ptr ds:[40A16A]       L"Bad boi!"
000000000040187E  lea rdx,qword ptr ds:[40A16A]      L"Bad boi!"
```

You can see, the bad message and the good message here. Just double click on the good message. You will redirect in the CPU part. If you scroll up this, you will see another bad message, two or three more time.

```
000000000040184A  84C0              test al,al
000000000040184C  74 23             je crackme.401871                          <- CONDITIONAL JUMP
000000000040184E  41:B9 00000000    mov r9d,0x0
0000000000401854  4C:8D05 25890000  lea r8,qword ptr ds:[0x40A180]             000000000040A180:L"Good boi!"
                                                                               000000000040A198:L"You are such a gud boi!
000000000040185B  48:8D15 36890000  lea rdx,qword ptr ds:[0x40A198]            Lemme pat you!"
0000000000401862  48:8B4D 20        mov rcx,qword ptr ss:[rbp+0x20]
0000000000401866  48:8B05 BBDB0000  mov rax,qword ptr ds:[<&MessageBoxW>]      ;Message box is call here
000000000040186D  FFD0              call rax
000000000040186F  EB 21             jmp crackme.401892
0000000000401871  41:B9 00000000    mov r9d,0x0
0000000000401877  4C:8D05 EC880000  lea r8,qword ptr ds:[0x40A16A]             000000000040A16A:L"Bad boi!"
000000000040187E  48:8D15 E5880000  lea rdx,qword ptr ds:[0x40A16A]            000000000040A16A:L"Bad boi!"
0000000000401885  48:8B4D 20        mov rcx,qword ptr ss:[rbp+0x20]
0000000000401889  48:8B05 98DB0000  mov rax,qword ptr ds:[<&MessageBoxW>]      ; MessageBox is call here
0000000000401890  FFD0              call rax
```

It isn't important for me for the moment, this part interess me because if see here a conditional jump. And i think that, this is the who say if the crackme show the good and bad message. Just put a breakpoint and run the click on Register again inside the crackme to verify that it's true.



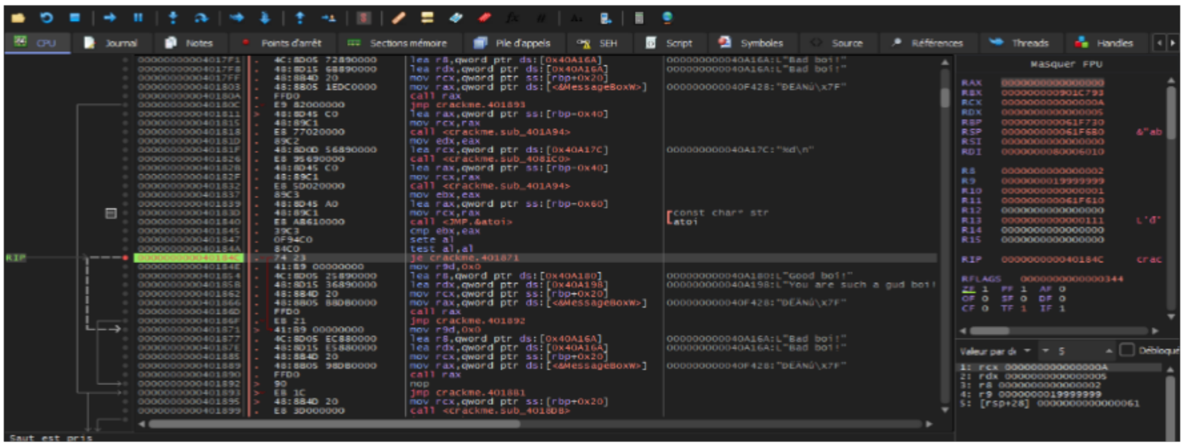
```
      0000000000401845   .  39C3              cmp ebx,eax
      0000000000401847   .  0F94C0            sete al
      000000000040184A   .  84C0              test al,al
RIP   000000000040184C   .  74 23             je crackme.401871
      000000000040184E   .  41:B9 00000000    mov r9d,0x0
      0000000000401854   .  4C:8D05 25890000  lea r8,qword ptr ds:[0x40A180]     000000000040A18
```

Right !!! The crackme is now paused at this point. If you Press F8 more man time, you will see the bad message again.

This instruction means that : Jump if the Zero Flag is Set to 1.

But the zero flag is set after the : test al, al -> test if al is set to 0.

## Now who set the value of al ??



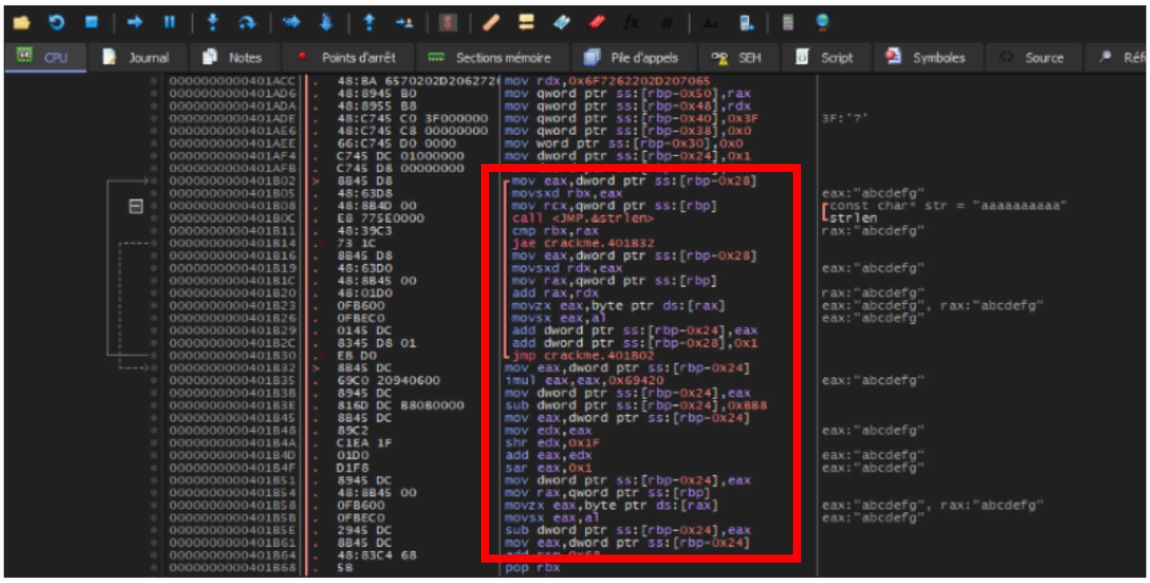
```
      00000000004017F1  .  4C:8D05 72890000  lea r8,qword ptr ds:[0x40A16A]          000000000040A16A:L"Bad boi!"
      00000000004017F8  .  48:8D15 6B890000  lea rdx,qword ptr ds:[0x40A16A]         000000000040A16A:L"Bad boi!"
      00000000004017FF  .  48:8B4D 20        mov rcx,qword ptr ss:[rbp+0x20]
      0000000000401803  .  48:8B05 1EDC0000  mov rax,qword ptr ds:[<&MessageBoxW>]   000000000040F428:"ÐÉÁNû\x7F"
      000000000040180A  .  FFD0              call rax
      000000000040180C  .  E9 82000000       jmp crackme.401893
      0000000000401811  >  48:8D45 C0        lea rax,qword ptr ss:[rbp-0x40]
      0000000000401815  .  48:89C1           mov rcx,rax
      0000000000401818  .  E8 77020000       call <crackme.sub_401A94>
      000000000040181D  .  89C2              mov edx,eax
      000000000040181F  .  48:8D0D 56890000  lea rcx,qword ptr ds:[0x40A17C]         000000000040A17C:"%d\n"
      0000000000401826  .  E8 95690000       call <crackme.sub_4081C0>
      000000000040182B  .  48:8D45 C0        lea rax,qword ptr ss:[rbp-0x40]
      000000000040182F  .  48:89C1           mov rcx,rax
      0000000000401832  .  E8 5D020000       call <crackme.sub_401A94>
      0000000000401837  .  89C3              mov ebx,eax
      0000000000401839  .  48:8D45 A0        lea rax,qword ptr ss:[rbp-0x60]
      000000000040183D  .  48:89C1           mov rcx,rax                             [const char* str
      0000000000401840  .  E8 A8610000       call <JMP.&atoi>                        [atoi
      0000000000401845  .  39C3              cmp ebx,eax
      0000000000401847  .  0F94C0            sete al
      000000000040184A  .  84C0              test al,al
RIP   000000000040184C  .  74 23             je crackme.401871
      000000000040184E  .  41:B9 00000000    mov r9d,0x0
      0000000000401854  .  4C:8D05 25890000  lea r8,qword ptr ds:[0x40A180]          000000000040A180:L"Good boi!"
      000000000040185B  .  48:8D15 36890000  lea rdx,qword ptr ds:[0x40A198]         000000000040A198:L"You are such a gud boi!
      0000000000401862  .  48:8B4D 20        mov rcx,qword ptr ss:[rbp+0x20]
      0000000000401866  .  48:8B05 BBDB0000  mov rax,qword ptr ds:[<&MessageBoxW>]   000000000040F428:"ÐÉÁNû\x7F"
      000000000040186D  .  FFD0              call rax
      000000000040186F  .  EB 21             jmp crackme.401892
      0000000000401871  >  41:B9 00000000    mov r9d,0x0
      0000000000401877  .  4C:8D05 EC880000  lea r8,qword ptr ds:[0x40A16A]          000000000040A16A:L"Bad boi!"
      000000000040187E  .  48:8D15 E5880000  lea rdx,qword ptr ds:[0x40A16A]         000000000040A16A:L"Bad boi!"
      0000000000401885  .  48:8B4D 20        mov rcx,qword ptr ss:[rbp+0x20]
      0000000000401889  .  48:8B05 98DB0000  mov rax,qword ptr ds:[<&MessageBoxW>]   000000000040F428:"ÐÉÁNû\x7F"
      0000000000401890  .  FFD0              call rax
      0000000000401892  >  90                nop
      0000000000401893  >  EB 1C             jmp crackme.401881
      0000000000401895  >  48:8B4D 20        mov rcx,qword ptr ss:[rbp+0x20]
      0000000000401899  .  E8 3D000000       call <crackme.sub_4018DB>
Saut est pris

RAX  0000000000000000
RBX  000000000901C793
RCX  000000000000000A
RDX  0000000000000005
RBP  000000000061F730
RSP  000000000061F6B0    &"ab
RSI  0000000000000000
RDI  0000000080006010

R8   0000000000000002
R9   0000000019999999
R10  0000000000000001
R11  000000000061F610
R12  0000000000000000
R13  0000000000000111    L'đ'
R14  0000000000000000
R15  0000000000000000

RIP  000000000040184C    crac

RFLAGS  0000000000000344
ZF 1   PF 1   AF 0
OF 0   SF 0   DF 0
CF 0   TF 1   IF 1

1: rcx 000000000000000A
2: rdx 0000000000000005
3: r8 0000000000000002
4: r9 0000000019999999
5: [rsp+28] 0000000000000061
```

The function atoi convert any strings to integer number, and the result is inside rax register, it's here that the value of al is set.If you look the value of ebx, you will see that is the same value that we have at the image 1.2 (inside console). I thing now that it's the good serial. Now we can find where the value of ebx is set.

You can see now another functions here :

```
000000000040182F 48:89C1     mov rcx,rax
0000000000401832 E8 5D020000 call <crackme.sub_401A94>
0000000000401837 89C3        mov ebx,eax
```

The last value of ebx is set after this calling, and the result is put inside eax. It's the good place to put breakpoint again. After put a breakpoint, just click on register again and step into this call.



```
0000000000401ACC  .  48:BA 6570202D2062726  mov rdx,0x6F7262202D207065
0000000000401AD6  .  48:8945 B0             mov qword ptr ss:[rbp-0x50],rax
0000000000401ADA  .  48:8955 B8             mov qword ptr ss:[rbp-0x48],rdx
0000000000401ADE  .  48:C745 C0 3F000000    mov qword ptr ss:[rbp-0x40],0x3F          3F:'?'
0000000000401AE6  .  48:C745 C8 00000000    mov qword ptr ss:[rbp-0x38],0x0
0000000000401AEE  .  66:C745 D0 0000        mov word ptr ss:[rbp-0x30],0x0
0000000000401AF4  .  C745 DC 01000000       mov dword ptr ss:[rbp-0x24],0x1
0000000000401AFB  .  C745 D8 00000000       [illegible]
0000000000401B02  >  8B45 D8                mov eax,dword ptr ss:[rbp-0x28]           eax:"abcdefg"
0000000000401B05  .  48:63D8                movsxd rbx,eax                            [const char* str = "aaaaaaaaaa"
0000000000401B08  .  48:8B4D 00             mov rcx,qword ptr ss:[rbp]                [strlen
0000000000401B0C  .  E8 775E0000            call <JMP.&strlen>                        rax:"abcdefg"
0000000000401B11  .  48:39C3                cmp rbx,rax
0000000000401B14  .v 73 1C                  jae crackme.401B32
0000000000401B16  .  8B45 D8                mov eax,dword ptr ss:[rbp-0x28]           eax:"abcdefg"
0000000000401B19  .  48:63D0                movsxd rdx,eax
0000000000401B1C  .  48:8B45 00             mov rax,qword ptr ss:[rbp]                rax:"abcdefg"
0000000000401B20  .  48:01D0                add rax,rdx                               eax:"abcdefg", rax:"abcdefg"
0000000000401B23  .  0FB600                 movzx eax,byte ptr ds:[rax]               eax:"abcdefg"
0000000000401B26  .  0FBEC0                 movsx eax,al
0000000000401B29  .  0145 DC                add dword ptr ss:[rbp-0x24],eax
0000000000401B2C  .  8345 D8 01             add dword ptr ss:[rbp-0x28],0x1
0000000000401B30  .^ EB D0                  jmp crackme.401B02
0000000000401B32  >  8B45 DC                mov eax,dword ptr ss:[rbp-0x24]
0000000000401B35  .  69C0 20940600          imul eax,eax,0x69420                      eax:"abcdefg"
0000000000401B3B  .  8945 DC                mov dword ptr ss:[rbp-0x24],eax
0000000000401B3E  .  816D DC B80B0000       sub dword ptr ss:[rbp-0x24],0xBB8
0000000000401B45  .  8B45 DC                mov eax,dword ptr ss:[rbp-0x24]
0000000000401B48  .  89C2                   mov edx,eax                               eax:"abcdefg"
0000000000401B4A  .  C1EA 1F                shr edx,0x1F
0000000000401B4D  .  01D0                   add eax,edx                               eax:"abcdefg"
0000000000401B4F  .  D1F8                   sar eax,0x1                               eax:"abcdefg"
0000000000401B51  .  8945 DC                mov dword ptr ss:[rbp-0x24],eax
0000000000401B54  .  48:8B45 00             mov rax,qword ptr ss:[rbp]
0000000000401B58  .  0FB600                 movzx eax,byte ptr ds:[rax]               eax:"abcdefg", rax:"abcdefg"
0000000000401B5B  .  0FBEC0                 movsx eax,al                              eax:"abcdefg"
0000000000401B5E  .  2945 DC                sub dword ptr ss:[rbp-0x24],eax
0000000000401B61  .  8B45 DC                mov eax,dword ptr ss:[rbp-0x24]
0000000000401B64  .  48:83C4 68             add rsp,0x68
0000000000401B68  .  5B                     pop rbx
```

The serial Key is generate Here ! Now just try to make a keygen.

<u>README</u>

I'm so sorry for this bad english, but i'm trying to speek well ! I'm french.

I think that this little writting will help you. See you letter

KeyGen is in << peach's crackme solution.c >>