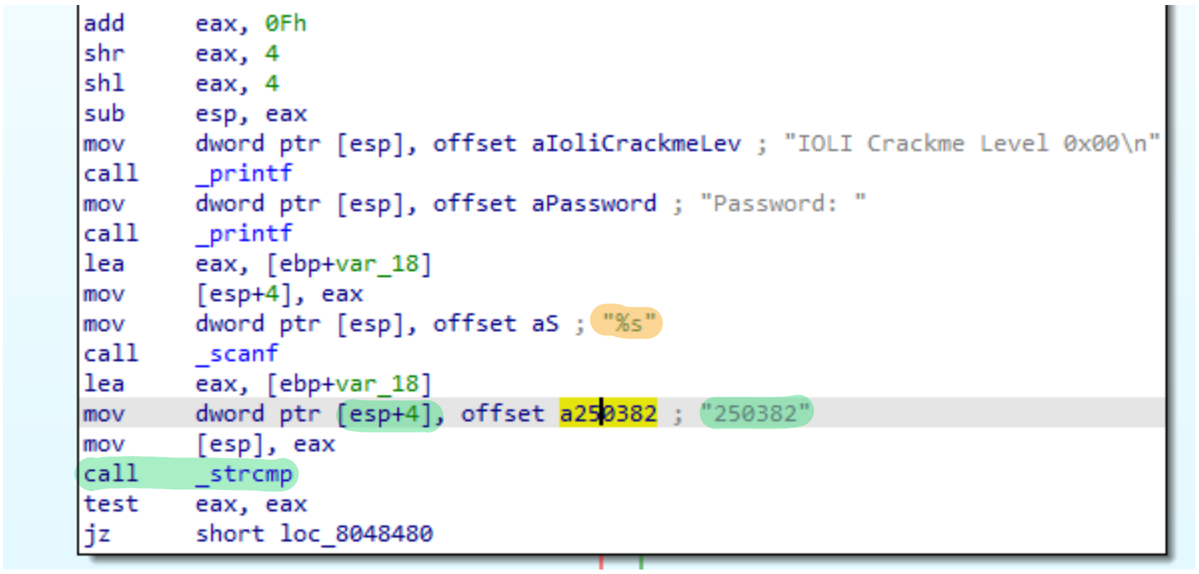
Lab 1:

crackme0x00

Steps:

1) Dissemble the binary using IDA
2) Switch to Graph view
3) Identify the main function.
4) Identify the `scanf` function asking for the password. The function `scanf` is using a format specifier `%s`
5) Trace to the `strcmp` function and find the password string "*250382*"

main function



```
add     eax, 0Fh
shr     eax, 4
shl     eax, 4
sub     esp, eax
mov     dword ptr [esp], offset aIoliCrackmeLev ; "IOLI Crackme Level 0x00\n"
call    _printf
mov     dword ptr [esp], offset aPassword ; "Password: "
call    _printf
lea     eax, [ebp+var_18]
mov     [esp+4], eax
mov     dword ptr [esp], offset aS ; "%s"
call    _scanf
lea     eax, [ebp+var_18]
mov     dword ptr [esp+4], offset a250382 ; "250382"
mov     [esp], eax
call    _strcmp
test    eax, eax
jz      short loc_8048480
```

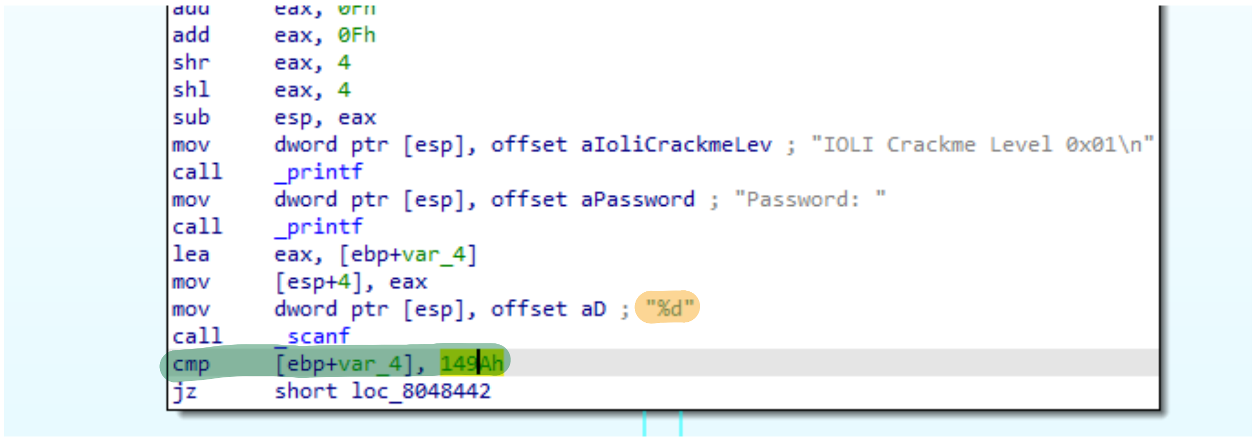
crackme0x01

Steps:

1) Dissemble the binary using IDA
2) Follow the steps from crackme0x00 to identify the `scanf` function. The function `scanf` is using a format specifier `%d` i.e. decimal.

3) Trace the control flow to `cmp` statement, find the hex comparison between `149A` and value at address `$ebp - 4`. `$ebp` is a register that holds the address of a stack frame pointer.
4) Convert the hex value `149A` to the decimal `5274` because `scanf` function requires a decimal `%d` input.

<u>main function</u>



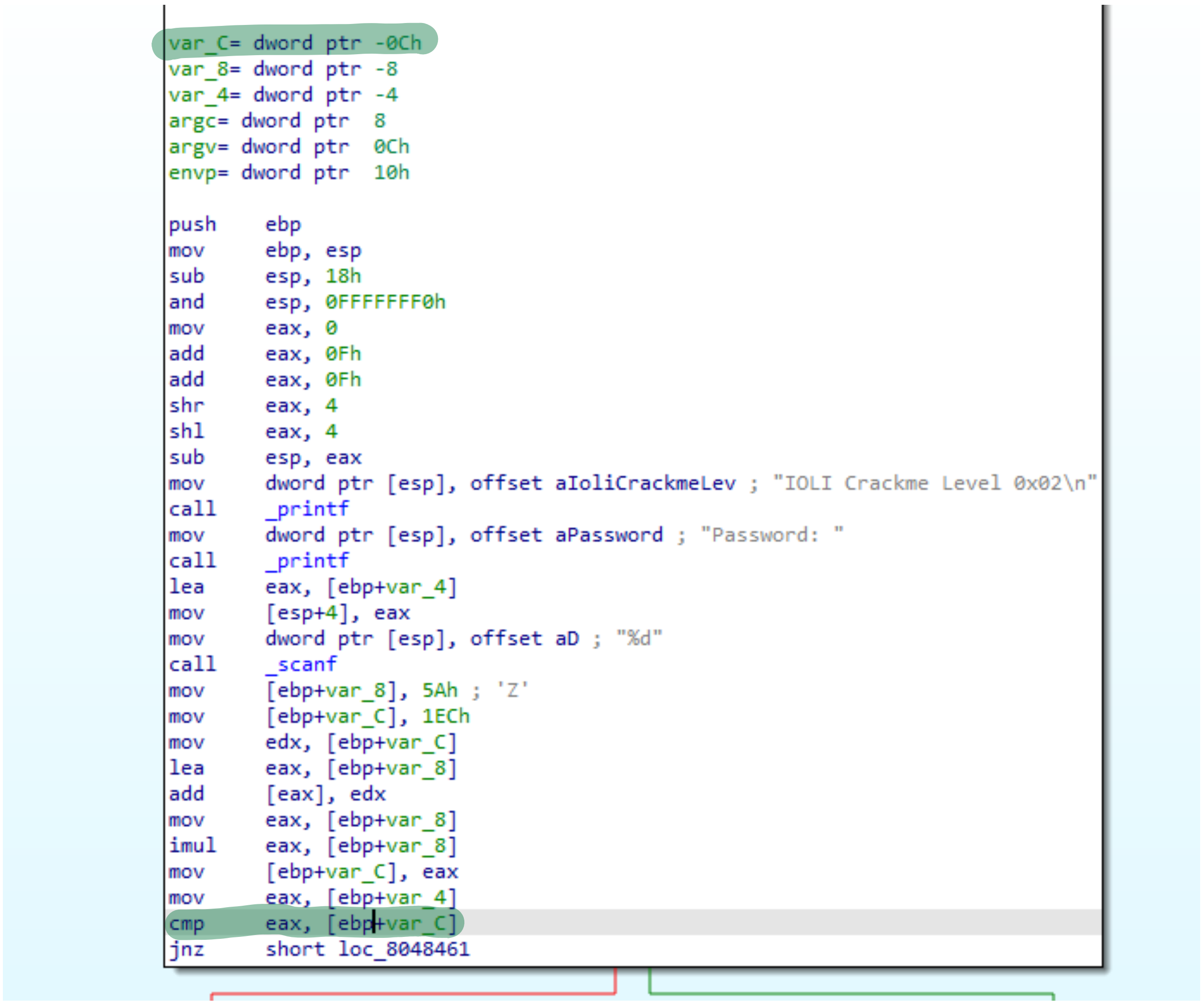
```
add     eax, 0Fh
add     eax, 0Fh
shr     eax, 4
shl     eax, 4
sub     esp, eax
mov     dword ptr [esp], offset aIoliCrackmeLev ; "IOLI Crackme Level 0x01\n"
call    _printf
mov     dword ptr [esp], offset aPassword ; "Password: "
call    _printf
lea     eax, [ebp+var_4]
mov     [esp+4], eax
mov     dword ptr [esp], offset aD ; "%d"
call    _scanf
cmp     [ebp+var_4], 149Ah
jz      short loc_8048442
```

<u>crackme0x02</u>

Steps:

1) Dissemble the binary using IDA
2) Follow the steps from crackme0x00 to identify the `scanf` function. This time as well the `scanf` is using a format specifier `%d` i.e. decimal.
3) Trace the control flow to `cmp` statement, find comparison between a value at `$eax` (return value from `scanf`) and `$ebp - 0xc`. Conclusion, the value at `$ebp - 0xc` must be the password.

main function



```
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 18h
and     esp, 0FFFFFFF0h
mov     eax, 0
add     eax, 0Fh
add     eax, 0Fh
shr     eax, 4
shl     eax, 4
sub     esp, eax
mov     dword ptr [esp], offset aIoliCrackmeLev ; "IOLI Crackme Level 0x02\n"
call    _printf
mov     dword ptr [esp], offset aPassword ; "Password: "
call    _printf
lea     eax, [ebp+var_4]
mov     [esp+4], eax
mov     dword ptr [esp], offset aD ; "%d"
call    _scanf
mov     [ebp+var_8], 5Ah ; 'Z'
mov     [ebp+var_C], 1ECh
mov     edx, [ebp+var_C]
lea     eax, [ebp+var_8]
add     [eax], edx
mov     eax, [ebp+var_8]
imul    eax, [ebp+var_8]
mov     [ebp+var_C], eax
mov     eax, [ebp+var_4]
cmp     eax, [ebp+var_C]
jnz     short loc_8048461
```

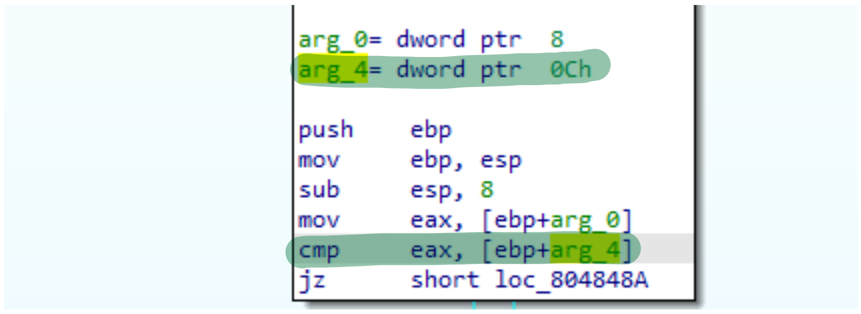
4) Run the program using gdb, set the breakpoint at `0x0804844e` i.e. `cmp` statement.
5) Print the value at `$ebp-0xc` using command `x /d $ebp-0xc`.

crackme0x03

Steps:

1) Dissemble the binary using IDA
2) Follow the steps from crackme0x00 to identify the `scanf` function. This time as well the `scanf` is using a format specifier `%d` i.e. decimal.
3) Trace the control flow to `cmp` statement in the test function, there is a comparison between a value at `$eax` - first parameter i.e. input, and `$ebp + 0xc` - second parameter of test function. Conclusion, the value at `$ebp + 0xc` must be the password.

test function



```
arg_0= dword ptr  8
arg_4= dword ptr  0Ch

push    ebp
mov     ebp, esp
sub     esp, 8
mov     eax, [ebp+arg_0]
cmp     eax, [ebp+arg_4]
jz      short loc_804848A
```

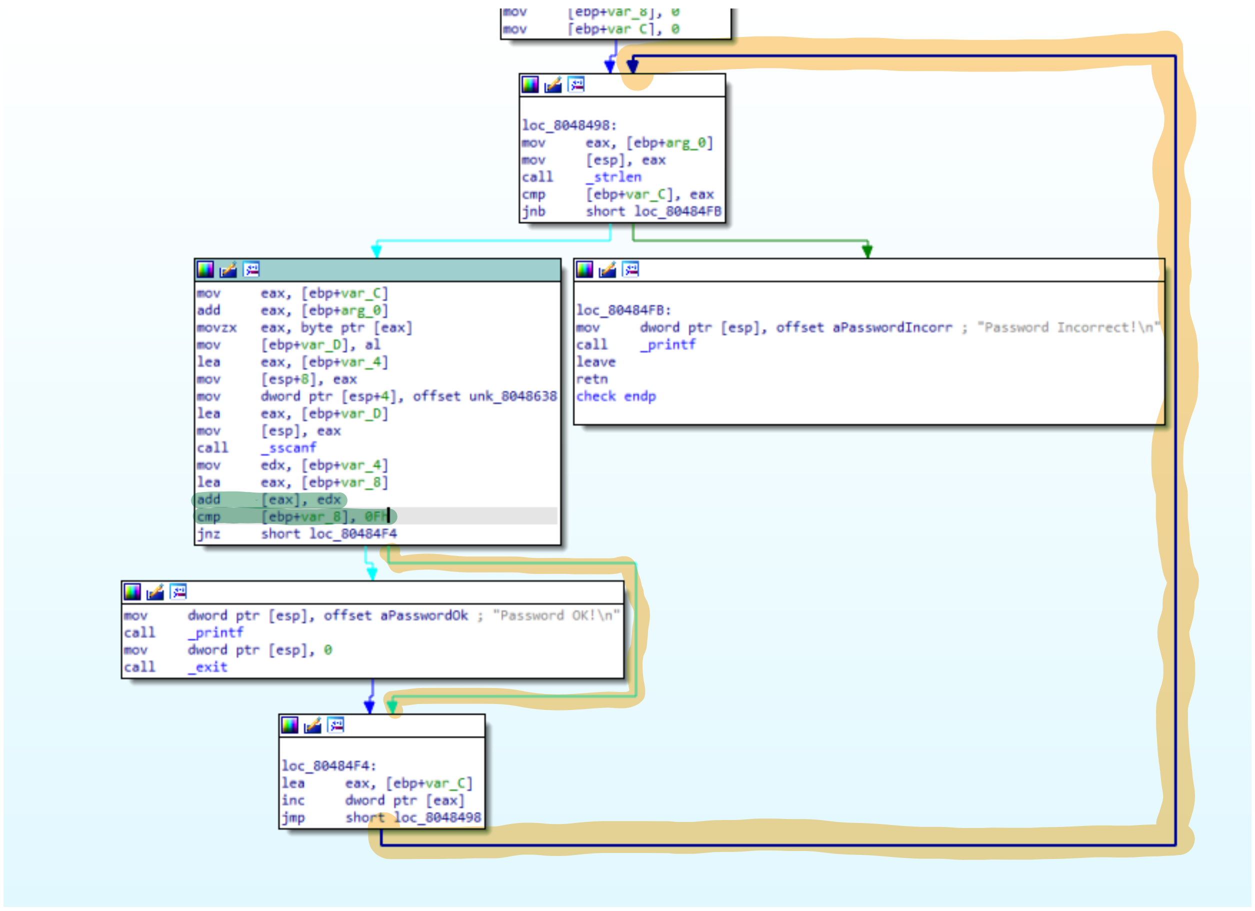
4) Run the program using gdb, set the breakpoint at `0x08048477` i.e. `cmp` statement.
5) To get the password, print the value at `$ebp+0xc` using command `x /d $ebp+0xc`.

crackme0x04

Steps:

1) Dissemble the binary using IDA
2) Follow the steps from crackme0x00 to identify the `scanf` function. This time as well, the `scanf` is using a format specifier `%s` i.e. string.
3) Trace the control flow to the `check` function. It performs following steps
    a) Read the next character from input as decimal integer
    b) Add to previous character/integer from the input
    c) Check, if the sum is equal to 15 i.e. `0x0F`. If true, terminate the loop and print “password ok!”
    d) Else repeat step a) through d) until all characters are read from the input

check function



```
mov     [ebp+var_8], 0
mov     [ebp+var_C], 0
```

```
loc_8048498:
mov     eax, [ebp+arg_0]
mov     [esp], eax
call    _strlen
cmp     [ebp+var_C], eax
jnb     short loc_80484FB
```

```
mov     eax, [ebp+var_C]
add     eax, [ebp+arg_0]
movzx   eax, byte ptr [eax]
mov     [ebp+var_D], al
lea     eax, [ebp+var_4]
mov     [esp+8], eax
mov     dword ptr [esp+4], offset unk_8048638
lea     eax, [ebp+var_D]
mov     [esp], eax
call    _sscanf
mov     edx, [ebp+var_4]
lea     eax, [ebp+var_8]
add     [eax], edx
cmp     [ebp+var_8], 0Fh
jnz     short loc_80484F4
```

```
loc_80484FB:
mov     dword ptr [esp], offset aPasswordIncorr ; "Password Incorrect!\n"
call    _printf
leave
retn
check endp
```

```
mov     dword ptr [esp], offset aPasswordOk ; "Password OK!\n"
call    _printf
mov     dword ptr [esp], 0
call    _exit
```

```
loc_80484F4:
lea     eax, [ebp+var_C]
inc     dword ptr [eax]
jmp     short loc_8048498
```

4) This means, password is any number with a sum of digits 15.
5) Try password 195 or 915 or 519 etc.

crackme0x05

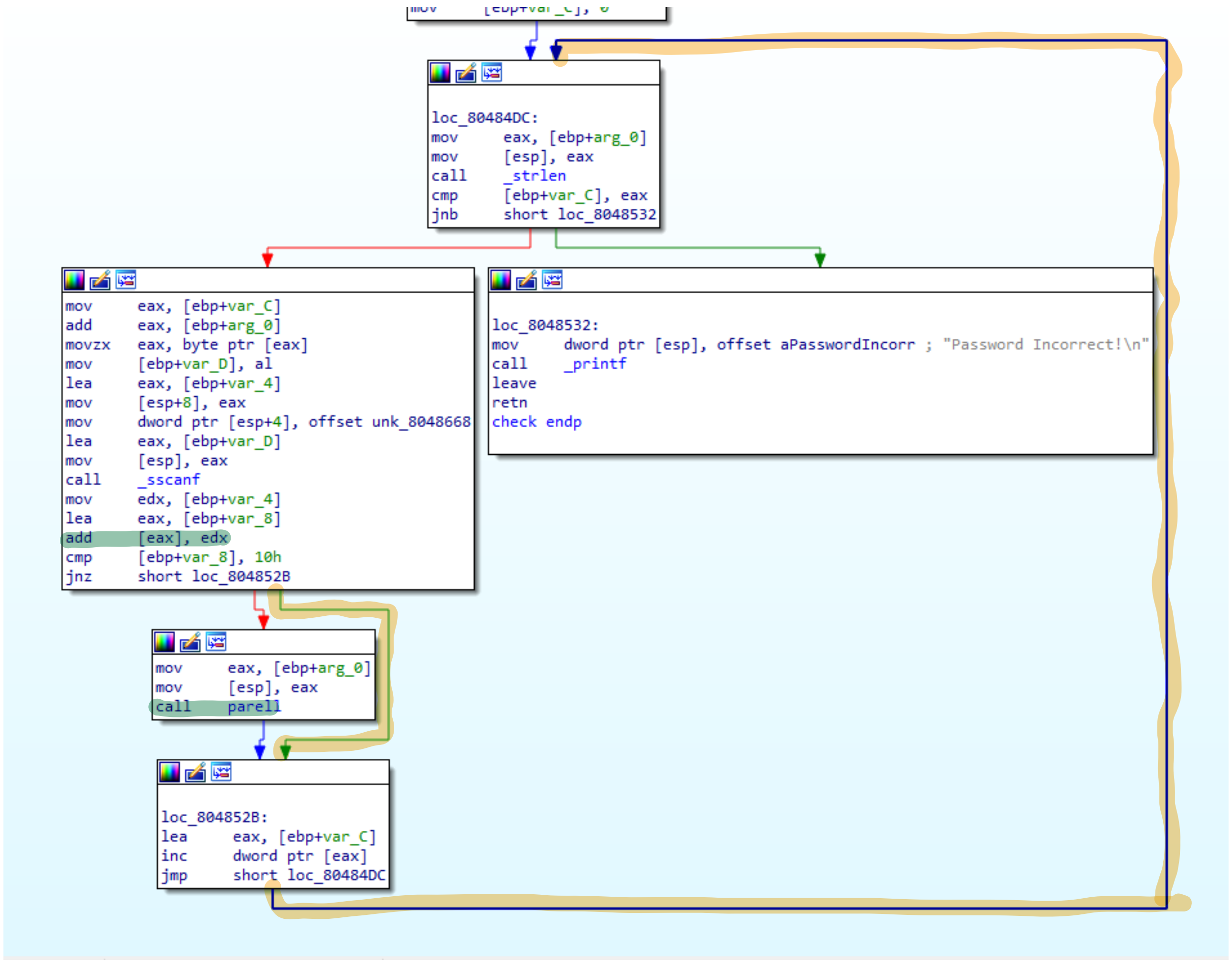
Steps:

1) Logic is similar to crackme0x04 with an additional check for even parity.
2) Dissemble the binary using IDA
3) Follow the steps from crackme0x00 to identify the `scanf` function. This time as well, the `scanf` is using a format specifier `%s` i.e. string
4) Trace the control flow to the `check` function. It performs following steps
    a) Read the next character from input as decimal integer
    b) Add to previous character/integer from the input

c) Check, if the sum is equal to 16 i.e. `0x10`. If true, call `parall` function

d) Else repeat step a) through d) until all characters are read from the input

<u>check function</u>



```
mov     [ebp+var_C], 0
```

```
loc_80484DC:
mov     eax, [ebp+arg_0]
mov     [esp], eax
call    _strlen
cmp     [ebp+var_C], eax
jnb     short loc_8048532
```

```
mov     eax, [ebp+var_C]
add     eax, [ebp+arg_0]
movzx   eax, byte ptr [eax]
mov     [ebp+var_D], al
lea     eax, [ebp+var_4]
mov     [esp+8], eax
mov     dword ptr [esp+4], offset unk_8048668
lea     eax, [ebp+var_D]
mov     [esp], eax
call    _sscanf
mov     edx, [ebp+var_4]
lea     eax, [ebp+var_8]
add     [eax], edx
cmp     [ebp+var_8], 10h
jnz     short loc_804852B
```

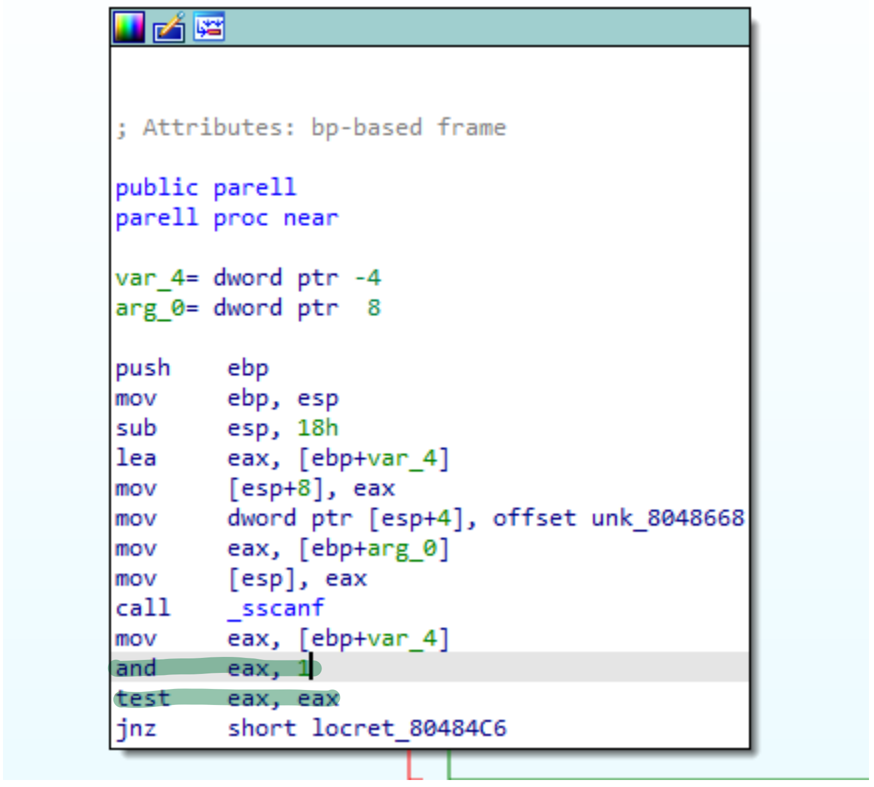
```
loc_8048532:
mov     dword ptr [esp], offset aPasswordIncorr ; "Password Incorrect!\n"
call    _printf
leave
retn
check endp
```

```
mov     eax, [ebp+arg_0]
mov     [esp], eax
call    parell
```

```
loc_804852B:
lea     eax, [ebp+var_C]
inc     dword ptr [eax]
jmp     short loc_80484DC
```

5) The `parall` function performs the parity check, by performing an `AND` operation on input(integer format) with 1

parell function



```
; Attributes: bp-based frame

public parell
parell proc near

var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 18h
lea     eax, [ebp+var_4]
mov     [esp+8], eax
mov     dword ptr [esp+4], offset unk_8048668
mov     eax, [ebp+arg_0]
mov     [esp], eax
call    _sscanf
mov     eax, [ebp+var_4]
and     eax, 1
test    eax, eax
jnz     short locret_80484C6
```

6) Conclusion, the password is any even number with a sum of the digits 16.

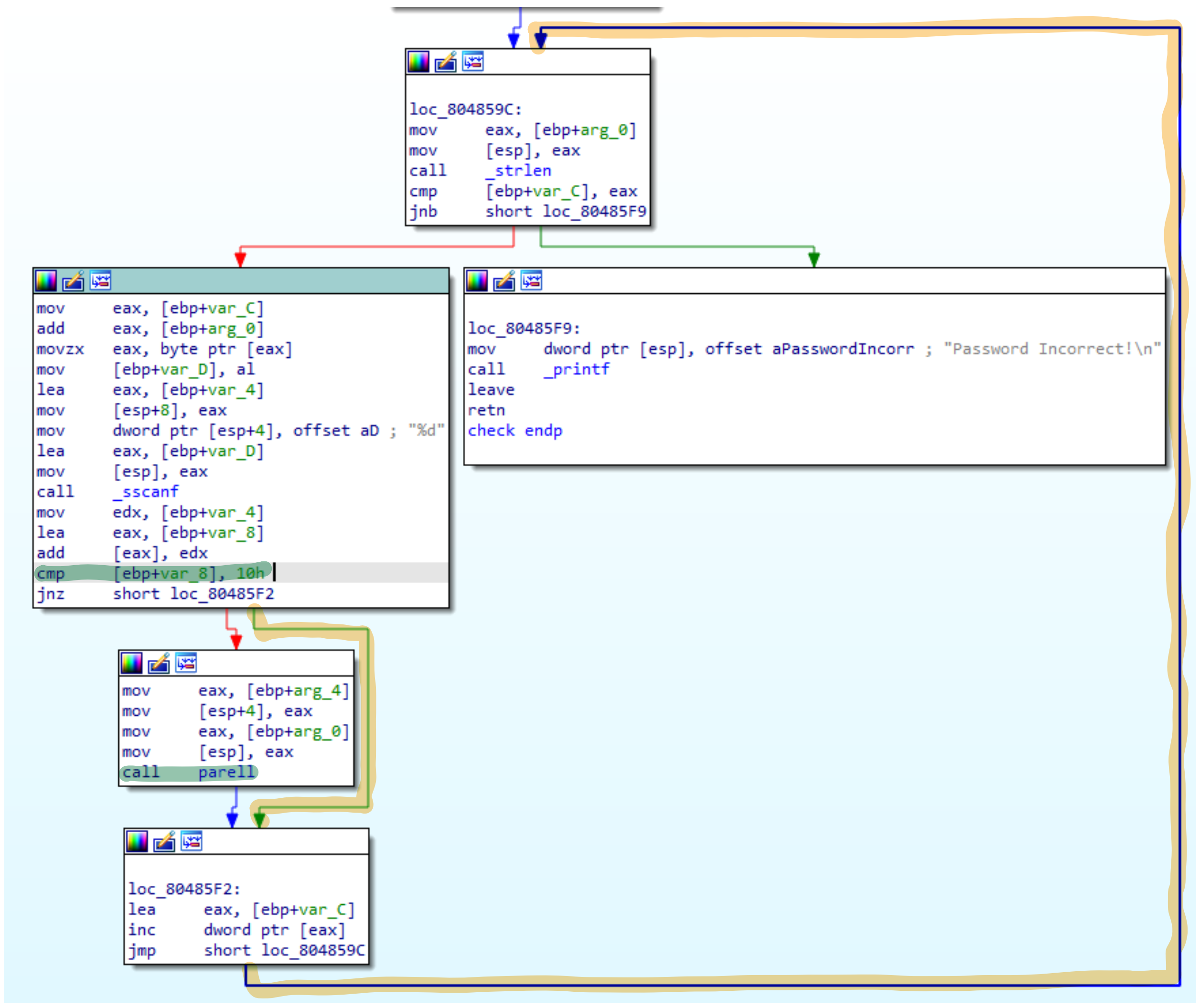
crackme0x06

Steps:

1) Logic is similar to crackme0x04 with an additional check for environment variables with the first 3 characters as "`LOL`".
2) Dissemble the binary using IDA
3) Follow the steps from crackme0x00 to identify the `scanf` function. This time as well, the `scanf` is using a format specifier `%s` i.e. string
4) Trace the control flow to the check function. It performs following steps
    a) Read the next character from input as decimal integer
    b) Add to previous character/integer from the input
    c) Check, if the sum is equal to 16 i.e. `0x10`. If true, call `parall` function

d) Else repeat step a) through d) until all characters are read from the input

check function



```
loc_804859C:
mov     eax, [ebp+arg_0]
mov     [esp], eax
call    _strlen
cmp     [ebp+var_C], eax
jnb     short loc_80485F9
```

```
mov     eax, [ebp+var_C]
add     eax, [ebp+arg_0]
movzx   eax, byte ptr [eax]
mov     [ebp+var_D], al
lea     eax, [ebp+var_4]
mov     [esp+8], eax
mov     dword ptr [esp+4], offset aD ; "%d"
lea     eax, [ebp+var_D]
mov     [esp], eax
call    _sscanf
mov     edx, [ebp+var_4]
lea     eax, [ebp+var_8]
add     [eax], edx
cmp     [ebp+var_8], 10h
jnz     short loc_80485F2
```

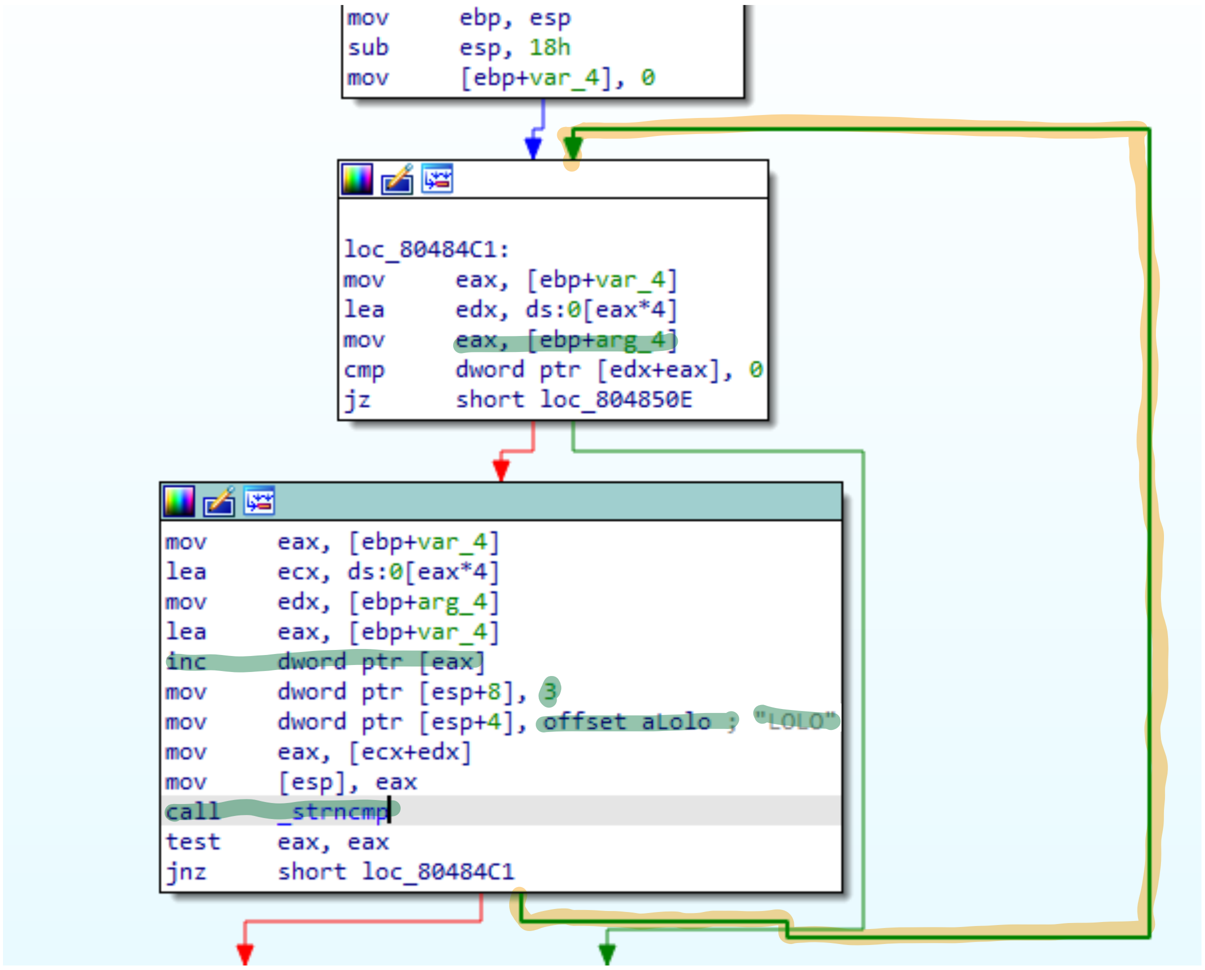
```
loc_80485F9:
mov     dword ptr [esp], offset aPasswordIncorr ; "Password Incorrect!\n"
call    _printf
leave
retn
check endp
```

```
mov     eax, [ebp+arg_4]
mov     [esp+4], eax
mov     eax, [ebp+arg_0]
mov     [esp], eax
call    parell
```

```
loc_80485F2:
lea     eax, [ebp+var_C]
inc     dword ptr [eax]
jmp     short loc_804859C
```

5) In the `parall` function, there is a call to `dummy` function which does following,
   a) Read the environment variable using the `**envp`
   b) Compare the env. variable with “`LOLO`” for first 3 letters, using `strncmp` function
   c) If match return to `parall` function else repeat step a) through c)

dummy function



```
mov     ebp, esp
sub     esp, 18h
mov     [ebp+var_4], 0
```

```
loc_80484C1:
mov     eax, [ebp+var_4]
lea     edx, ds:0[eax*4]
mov     eax, [ebp+arg_4]
cmp     dword ptr [edx+eax], 0
jz      short loc_804850E
```

```
mov     eax, [ebp+var_4]
lea     ecx, ds:0[eax*4]
mov     edx, [ebp+arg_4]
lea     eax, [ebp+var_4]
inc     dword ptr [eax]
mov     dword ptr [esp+8], 3
mov     dword ptr [esp+4], offset aLolo ; "LOLO"
mov     eax, [ecx+edx]
mov     [esp], eax
call    _strncmp
test    eax, eax
jnz     short loc_80484C1
```

6) So, to get the password to match, set env variable using cmd `set environment LOLO=1`
7) Once the environment variable is set, password is any number with a sum of digits 16.