IOLI crackmes level 1 to 4

Goal

Get the right password of the binaries as provided by the georgia tech security lab for their https://omscs.gatech.edu/cs-6265-information-security-labcourse.

`crackme0x00`

```
ski@lab:~/pwn/gatech/tuts/lab01/tut01-crackme$ ./crackme0x00
IOLI Crackme Level 0x00
Password: 1234
Invalid Password!
```

Providing a guessed password we get a wrong password message. we can use gdb to disassemble and debug our simple binary. when reversing any binary main is the entry point to start our analysis.

```
(gdb) ds main
Dump of assembler code for function main:
   0x080486a3 <+0>:    push   ebp
   0x080486a4 <+1>:    mov    ebp,esp
   0x080486a6 <+3>:    sub    esp,0x10
   0x080486a9 <+6>:    push   0x80487f4
   0x080486ae <+11>:    call   0x8048470 <puts@plt>
   0x080486b3 <+16>:    add    esp,0x4
   0x080486b6 <+19>:    push   0x804880c
   0x080486bb <+24>:    call   0x8048430 <printf@plt>
   0x080486c0 <+29>:    add    esp,0x4
   0x080486c3 <+32>:    lea    eax,[ebp-0x10]
   0x080486c6 <+35>:    push   eax
   0x080486c7 <+36>:    push   0x80487f1
   0x080486cc <+41>:    call   0x8048480 <scanf@plt>
   0x080486d1 <+46>:    add    esp,0x8
   0x080486d4 <+49>:    push   0x8048817
   0x080486d9 <+54>:    lea    eax,[ebp-0x10]
   0x080486dc <+57>:    push   eax
   0x080486dd <+58>:    call   0x8048420 <strcmp@plt>
   0x080486e2 <+63>:    add    esp,0x8
   0x080486e5 <+66>:    test   eax,eax
   0x080486e7 <+68>:    jne    0x8048705 <main+98>
   0x080486e9 <+70>:    push   0x804881e
   0x080486ee <+75>:    call   0x8048470 <puts@plt>
   0x080486f3 <+80>:    add    esp,0x4
   0x080486f6 <+83>:    push   0x804882d
   0x080486fb <+88>:    call   0x80485f6 <print_key>
   0x08048700 <+93>:    add    esp,0x4
   0x08048703 <+96>:    jmp    0x8048712 <main+111>
   0x08048705 <+98>:    push   0x804883c
   0x0804870a <+103>:    call   0x8048470 <puts@plt>
   0x0804870f <+108>:    add    esp,0x4
   0x08048712 <+111>:    mov    eax,0x0
```

```
   0x08048717 <+116>:    leave
   0x08048718 <+117>:    ret
End of assembler dump.
(gdb)
```

From the above disassembly code we can deduce that we are scanning the password from the user using the scanf function. our input is the stored in the `ebp-0x10` buffer and then compared with the value being pushed at the address `0x8048817`. we put a breakpoint at start of our binary and then check the value at the address being compared.

```
(gdb) start
Temporary breakpoint 1 at 0x80486a9: file crackme0x00.c, line 12.
Starting program: /home/ski/pwn/gatech/tuts/lab01/tut01-crackme/crackme0x00

Temporary breakpoint 1, main (argc=1, argv=0xffffd234) at crackme0x00.c:12
12    in crackme0x00.c
(gdb) x/s 0x8048817
0x8048817:    "250381"
(gdb) #we then put another breakoint at main+63
(gdb) break *main+63
Breakpoint 2 at 0x80486e2: file crackme0x00.c, line 15.
(gdb) c
Continuing.
IOLI Crackme Level 0x00
Password: 250381

Breakpoint 2, 0x080486e2 in main (argc=1, argv=0xffffd234) at crackme0x00.c:15
15    in crackme0x00.c
(gdb)
```

because our input is equal to our to the string being compared to we get a password ok message which is our goal.

```
ski@lab:~/pwn/gatech/tuts/lab01/tut01-crackme$ ./crackme0x00
IOLI Crackme Level 0x00
Password: 250381
Password OK :)
crackme0x00: Please insert your kflag.ko to get the flag!: No such file or directory
```

`crackme0x001`

This is similiar to the previous challenge. we can open the binary in gdb and start our analysis in the main function.

```
(gdb) ds main
Dump of assembler code for function main:
   0x08048486 <+0>:    push   ebp
   0x08048487 <+1>:    mov    ebp,esp
   0x08048489 <+3>:    sub    esp,0x4
   0x0804848c <+6>:    push   0x8048570
   0x08048491 <+11>:    call   0x8048330 <puts@plt>
   0x08048496 <+16>:    add    esp,0x4
```

```
   0x08048499 <+19>:    push   0x8048588
   0x0804849e <+24>:    call   0x8048320 <printf@plt>
   0x080484a3 <+29>:    add    esp,0x4
   0x080484a6 <+32>:    lea    eax,[ebp-0x4]
   0x080484a9 <+35>:    push   eax
   0x080484aa <+36>:    push   0x8048593
   0x080484af <+41>:    call   0x8048340 <scanf@plt>
   0x080484b4 <+46>:    add    esp,0x8
   0x080484b7 <+49>:    mov    eax,DWORD PTR [ebp-0x4]
   0x080484ba <+52>:    cmp    eax,0xc8e
   0x080484bf <+57>:    jne    0x80484d0 <main+74>
   0x080484c1 <+59>:    push   0x8048596
   0x080484c6 <+64>:    call   0x8048330 <puts@plt>
   0x080484cb <+69>:    add    esp,0x4
   0x080484ce <+72>:    jmp    0x80484dd <main+87>
   0x080484d0 <+74>:    push   0x80485a5
   0x080484d5 <+79>:    call   0x8048330 <puts@plt>
   0x080484da <+84>:    add    esp,0x4
   0x080484dd <+87>:    mov    eax,0x0
   0x080484e2 <+92>:    leave
   0x080484e3 <+93>:    ret
End of assembler dump.
(gdb) p/d 0xc8e
$1 = 3214
(gdb)
```

From the above disassembled code we can see the password is 3214.

```
ski@lab:~/pwn/gatech/tuts/lab01/tut01-crackme$ ./crackme0x01
IOLI Crackme Level 0x01
Password: 3214
Password OK :)
```

crackme0x02

```
(gdb) ds main
Dump of assembler code for function main:
   0x08048486 <+0>:    push   ebp
   0x08048487 <+1>:    mov    ebp,esp
   0x08048489 <+3>:    sub    esp,0x4
   0x0804848c <+6>:    push   0x8048570
   0x08048491 <+11>:    call   0x8048330 <puts@plt>
   0x08048496 <+16>:    add    esp,0x4
   0x08048499 <+19>:    push   0x8048588
   0x0804849e <+24>:    call   0x8048320 <printf@plt>
   0x080484a3 <+29>:    add    esp,0x4
   0x080484a6 <+32>:    lea    eax,[ebp-0x4]
   0x080484a9 <+35>:    push   eax
   0x080484aa <+36>:    push   0x8048593
   0x080484af <+41>:    call   0x8048340 <scanf@plt>
   0x080484b4 <+46>:    add    esp,0x8
   0x080484b7 <+49>:    mov    eax,DWORD PTR [ebp-0x4]
   0x080484ba <+52>:    imul   eax,eax,0x159
```

```
   0x080484c0 <+58>:    cmp    eax,0x122c1c
   0x080484c5 <+63>:    jne    0x80484d6 <main+80>
   0x080484c7 <+65>:    push   0x8048596
   0x080484cc <+70>:    call   0x8048330 <puts@plt>
   0x080484d1 <+75>:    add    esp,0x4
   0x080484d4 <+78>:    jmp    0x80484e3 <main+93>
   0x080484d6 <+80>:    push   0x80485a5
   0x080484db <+85>:    call   0x8048330 <puts@plt>
   0x080484e0 <+90>:    add    esp,0x4
   0x080484e3 <+93>:    mov    eax,0x0
   0x080484e8 <+98>:    leave
   0x080484e9 <+99>:    ret
End of assembler dump.
(gdb) p/d 0x122c1c/0x159
$1 = 3452
(gdb) # we are multiplying our user password with 0x159 and then compare with 0x122c1c
(gdb) c
The program is not being run.
```

The above disassembled code,we can see we are scanning some user input and then multiply by the given specified value `0x159` and then compare the result with `0x122c1c`

From some simple math, the value is 3452.

```
ski@lab:~/pwn/gatech/tuts/lab01/tut01-crackme$ ./crackme0x02
IOLI Crackme Level 0x02
Password: 3452
Password OK :)
```

`crackme0x03`

This is a bit tricky compared to the previous three challenges. it contains 3 functions . Main, test and shift function.

from the disassembled code we are pushing two paramaters to the test function.

```
(gdb) ds main
Dump of assembler code for function main:
   0x08048546 <+0>:    push   ebp
   0x08048547 <+1>:    mov    ebp,esp
   0x08048549 <+3>:    sub    esp,0x4
   0x0804854c <+6>:    push   0x8048638
   0x08048551 <+11>:    call   0x8048350 <puts@plt>
   0x08048556 <+16>:    add    esp,0x4
   0x08048559 <+19>:    push   0x8048650
   0x0804855e <+24>:    call   0x8048340 <printf@plt>
   0x08048563 <+29>:    add    esp,0x4
   0x08048566 <+32>:    lea    eax,[ebp-0x4]
   0x08048569 <+35>:    push   eax
   0x0804856a <+36>:    push   0x804865b
   0x0804856f <+41>:    call   0x8048360 <scanf@plt>
   0x08048574 <+46>:    add    esp,0x8
   0x08048577 <+49>:    mov    eax,DWORD PTR [ebp-0x4]
```

```
   0x0804857a <+52>:    push   0x52b23
   0x0804857f <+57>:    push   eax
   0x08048580 <+58>:    call   0x8048511 <test>
   0x08048585 <+63>:    add    esp,0x8
   0x08048588 <+66>:    mov    eax,0x0
   0x0804858d <+71>:    leave
   0x0804858e <+72>:    ret
End of assembler dump.
```

we hand decompile this to c.

```
test(user_input,0x52b23) /*user_input is our scanned password */
```

we then disassemble the test function to understand the logic.

```
(gdb) ds test
Dump of assembler code for function test:
   0x08048511 <+0>:    push   ebp
   0x08048512 <+1>:    mov    ebp,esp
   0x08048514 <+3>:    sub    esp,0x4
   0x08048517 <+6>:    mov    eax,DWORD PTR [ebp+0x8]
   0x0804851a <+9>:    cmp    eax,DWORD PTR [ebp+0xc]
   0x0804851d <+12>:    jne    0x8048531 <test+32>
   0x0804851f <+14>:    push   0x8048614
   0x08048524 <+19>:    call   0x80484b6 <shift>
   0x08048529 <+24>:    add    esp,0x4
   0x0804852c <+27>:    mov    DWORD PTR [ebp-0x4],eax
   0x0804852f <+30>:    jmp    0x8048541 <test+48>
   0x08048531 <+32>:    push   0x8048626
   0x08048536 <+37>:    call   0x80484b6 <shift>
   0x0804853b <+42>:    add    esp,0x4
   0x0804853e <+45>:    mov    DWORD PTR [ebp-0x4],eax
   0x08048541 <+48>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08048544 <+51>:    leave
   0x08048545 <+52>:    ret
End of assembler dump.
```

we can see the value stored at ebp+0x8 and ebp+0xc are being compared.if the values are equal we continue our execution to the shift function.

```
ski@lab:~/pwn/gatech/tuts/lab01/tut01-crackme$ rax2 0x52b23
338723
ski@lab:~/pwn/gatech/tuts/lab01/tut01-crackme$ ./crackme0x03
IOLI Crackme Level 0x03
Password: 338723
Password OK!!! :)
```