jamk

Reverse Engineering

Lab 02

Timo Lehosvuo, M3426

Report
Reverse Engineering, Marko Silokunnas
21.2.2021
ICT

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

Sisällys

1 Lab 02

Just like in Lab01 I started reverse engineering the file by checking what the main function does. I found that it holds a function “check_password” and I started looking for more clues there:

```
push    ebp
mov     ebp, esp
sub     esp, 28h        ; char *
mov     eax, [ebp+arg_4]
mov     ecx, [ebp+arg_0]
lea     edx, aPassword  ; "Password: "
mov     [ebp+var_4], 0
mov     [ebp+var_8], ecx
mov     [ebp+var_C], eax
mov     [ebp+var_10], 0
mov     [esp+28h+var_28], edx
call    _printf
lea     ecx, aD         ; "%d"
lea     edx, [ebp+var_10]
mov     [esp+28h+var_28], ecx
mov     [esp+28h+var_24], edx
mov     [ebp+var_14], eax
call    ___isoc99_scanf
mov     ecx, [ebp+var_10]
mov     [esp+28h+var_28], ecx
mov     [ebp+var_18], eax
call    check_password
xor     eax, eax
add     esp, 28h
pop     ebp
retn
endp
```

Figure 1: Main function.

After looking through the “check_password” function I found out that it holds a function called “CORRECT” and I checked what it does:

```
push    ebp
mov     ebp, esp
sub     esp, 28h        ; char *
mov     eax, [ebp+arg_0]
lea     ecx, asc_80485F7 ; "%x"
lea     edx, [ebp+var_8]
mov     [ebp+var_4], eax
mov     [ebp+var_8], 0
mov     eax, CORRECT
mov     [esp+28h+var_28], eax
mov     [esp+28h+var_24], ecx
mov     [esp+28h+var_20], edx
call    ___isoc99_sscanf
mov     ecx, [ebp+var_8]
cmp     ecx, [ebp+var_4]
mov     [ebp+var_C], eax
jnz     loc_80484F6
lea     eax, aCorrect   ; "correct\n"
mov     [esp+28h+var_28], eax
call    _printf
mov     [ebp+var_10], eax
jmp     loc_8048507
```

Figure 2: check_password function.

The “CORRECT” function looked a lot like the solution for the “lab00” so I immediately thought that answer is “0xFACE”:

```
                public CORRECT
CORRECT         dd offset a0xface       ; DATA XREF: check_password+1C↑r
_data           ends                    ; "0xFACE"
```

Figure 3: CORRECT function.

The “0x” in the beginning made me think that its hexadecimal so I removed it changed the rest to decimal and got “64206”:

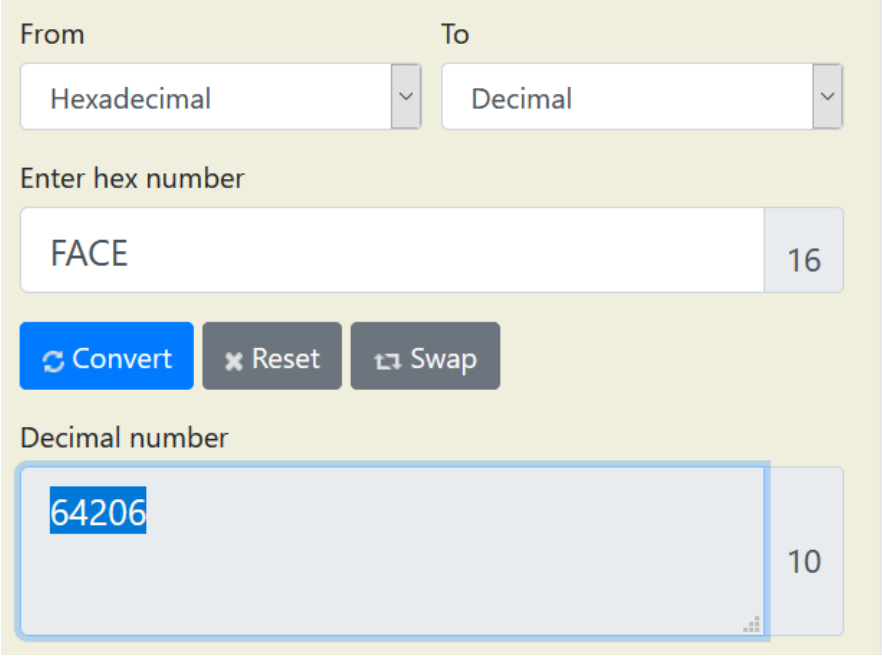

Figure 4: Hexadecimal to decimal.

Finally, I tested if it was the correct answer and it was:

```
root@kali:~# cd Desktop/labs/
root@kali:~/Desktop/labs# ./lab02
Password: 64206
correct
root@kali:~/Desktop/labs#
```

Figure 5: correct password.

Obviously, this is not a technical way to solve the lab but it had a lot of obvious clues, so I managed to solve it really quick.

2. Time spent

Solving the lab:	15 minutes
Total for both labs:	7 hours 15 minutes