

PicoCTF – Low Level Binary Intro

By zxhry

Intro to Assembly

Bit-O-Asm-1

Bit-O-Asm-1



Medium

Reverse Engineering

picoGym Exclusive

x86_64

AUTHOR: LT 'SYREAL' JONES

Hints ?

Description

1

Can you figure out what is in the `eax` register? Put your answer in the picoCTF flag format: `picoCTF{n}` where `n` is the contents of the `eax` register in the decimal number base. If the answer was `0x11` your flag would be `picoCTF{17}`.

Download the assembly dump [here](#).

Hint: As with most assembly, there is a lot of noise in the instruction dump. Find the one line that pertains to this question and don't second guess yourself!

```
~/Downloads/disassembler-dump0_a.txt - Mousepad
File Edit Search View Document Help
1 <+0>: endbr64
2 <+4>: push rbp
3 <+5>: mov rbp, rsp
4 <+8>: mov DWORD PTR [rbp-0x4], edi
5 <+11>: mov QWORD PTR [rbp-0x10], rsi
6 <+15>: mov eax, 0x30
7 <+20>: pop rbp
8 <+21>: ret
9
```

0x30 = 48

Bit-O-Asm-2

Bit-O-Asm-2

Medium

Reverse Engineering

picoGym Exclusive

x86_64

AUTHOR: LT 'SYREAL' JONES

Description

Can you figure out what is in the `eax` register? Put your answer in the picoCTF flag format: `picoCTF{n}` where `n` is the contents of the `eax` register in the decimal number base. If the answer was `0x11` your flag would be `picoCTF{17}`.
Download the assembly dump [here](#).

Hints ?

1

`PTR`'s or 'pointers', reference a location in memory where values can be stored.

Hint: `PTR`'s or 'pointers', reference a location in memory where values can be stored.

```
~/Downloads/disassembler-dump0_b.txt - Mousepad
File Edit Search View Document Help
1 |<+0>: endbr64
2 <+4>: push rbp
3 <+5>: mov rbp, rsp
4 <+8>: mov DWORD PTR [rbp-0x14], edi
5 <+11>: mov QWORD PTR [rbp-0x20], rsi
6 <+15>: mov DWORD PTR [rbp-0x4], 0x9fe1a
7 <+22>: mov eax, DWORD PTR [rbp-0x4]
8 <+25>: pop rbp
9 <+26>: ret
10
```

0x9fe1a = 654874

Bit-O-Asm-3

Bit-O-Asm-3

Medium Reverse Engineering picoGym Exclusive x86_64

AUTHOR: LT 'SYREAL' JONES

Description

Can you figure out what is in the `eax` register? Put your answer in the picoCTF flag format: `picoCTF{n}` where `n` is the contents of the `eax` register in the decimal number base. If the answer was `0x11` your flag would be `picoCTF{17}`.

Download the assembly dump [here](#).

Hints ?

1

Not everything in this disassembly listing is optimal.

Hint: Not everything in this disassembly listing is optimal.

```
~/Downloads/disassembler-dump0_c.txt - Mousepad
File Edit Search View Document Help
1 <+0>: endbr64
2 <+4>: push rbp
3 <+5>: mov rbp, rsp
4 <+8>: mov DWORD PTR [rbp-0x14], edi
5 <+11>: mov QWORD PTR [rbp-0x20], rsi
6 <+15>: mov DWORD PTR [rbp-0xc], 0x9fe1a
7 <+22>: mov DWORD PTR [rbp-0x8], 0x4
8 <+29>: mov eax, DWORD PTR [rbp-0xc]
9 <+32>: imul eax, DWORD PTR [rbp-0x8]
10 <+36>: add eax, 0x1f5
11 <+41>: mov DWORD PTR [rbp-0x4], eax
12 <+44>: mov eax, DWORD PTR [rbp-0x4]
13 <+47>: pop rbp
14 <+48>: ret
15
```

<+15>: mov DWORD PTR [rbp-0xc], 0x9fe1a → 654874

<+22>: mov DWORD PTR [rbp-0x8], 0x4 → 4

<+29>: mov eax, DWORD PTR [rbp-0xc] → eax = 654874

<+32>: imul eax, DWORD PTR [rbp-0x8] → eax = eax * 4 = 2619496

<+36>: add eax, 0x1f5 → eax += 501 = 2619997

<+41>: mov DWORD PTR [rbp-0x4], eax → [rbp-0x4] = eax

<+44>: mov eax, DWORD PTR [rbp-0x4] → return value in eax, 2619997

Bit-O-Asm-4

Bit-O-Asm-4



Medium Reverse Engineering picoGym Exclusive x86_64

AUTHOR: LT 'SYREAL' JONES

Description

Can you figure out what is in the `eax` register? Put your answer in the picoCTF flag format: `picoCTF{n}` where `n` is the contents of the `eax` register in the decimal number base. If the answer was `0x11` your flag would be `picoCTF{17}`.

Download the assembly dump [here](#).

Hints ?

1 2

Don't tell anyone I told you this, but you can solve this problem without understanding the compare/jump relationship.

Hint 1: Don't tell anyone I told you this, but you can solve this problem without understanding the compare/jump relationship.

Hint 2: Of course, if you're really good, you'll only need one attempt to solve this problem.

```
~/Downloads/disassembler-dump0_d.txt - Mousepad
File Edit Search View Document Help
1 <+0>: endbr64
2 <+4>: push rbp
3 <+5>: mov rbp, rsp
4 <+8>: mov DWORD PTR [rbp-0x14], edi
5 <+11>: mov QWORD PTR [rbp-0x20], rsi
6 <+15>: mov DWORD PTR [rbp-0x4], 0x9fe1a
7 <+22>: cmp DWORD PTR [rbp-0x4], 0x2710
8 <+29>: jle 0x5555555514e <main+37>
9 <+31>: sub DWORD PTR [rbp-0x4], 0x65
10 <+35>: jmp 0x55555555152 <main+41>
11 <+37>: add DWORD PTR [rbp-0x4], 0x65
12 <+41>: mov eax, DWORD PTR [rbp-0x4]
13 <+44>: pop rbp
14 <+45>: ret
15 |
```

<+15>: mov DWORD PTR [rbp-0x4], 0x9fe1a → 654874

<+22>: cmp DWORD PTR [rbp-0x4], 0x2710 → compare with 10000

<+29>: jle 0x5555555514e <main+37> → if ≤ 10000, jump to add

<+31>: sub DWORD PTR [rbp-0x4], 0x65 → else, subtract 101 = 654773

<+35>: jmp 0x55555555152 <main+41> → skip add

<+37>: add DWORD PTR [rbp-0x4], 0x65 → branch target if ≤ 10000

<+41>: mov eax, DWORD PTR [rbp-0x4] → return value in eax, 654773