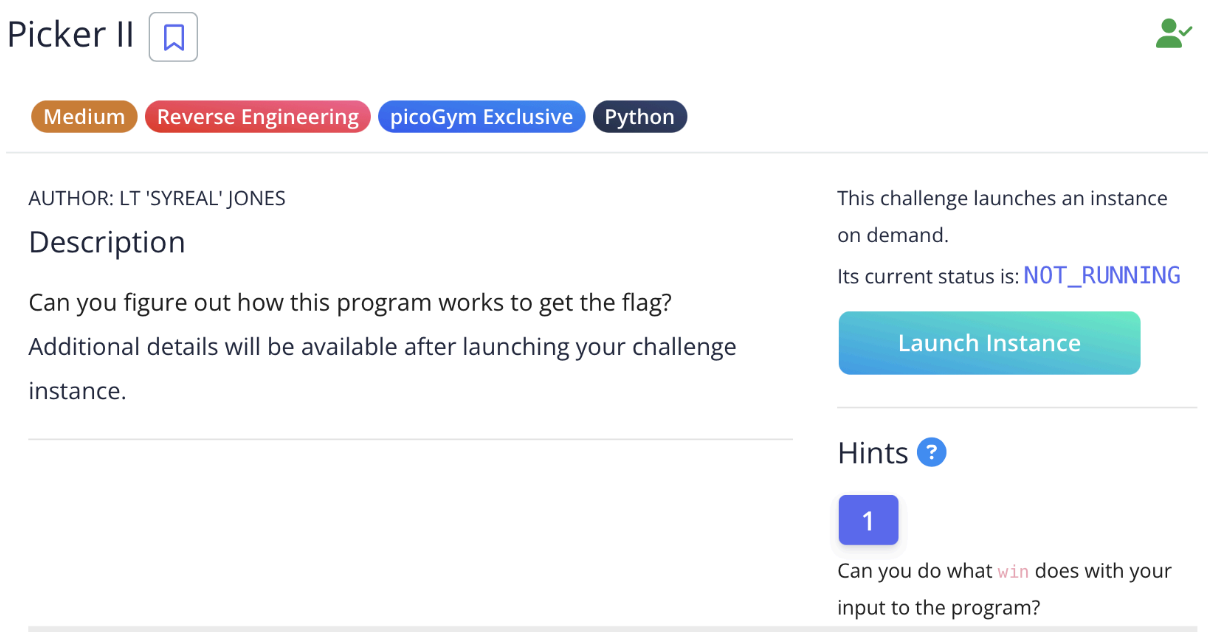


Picker II

Medium · Reverse Engineering · picoGym Exclusive · Python

AUTHOR: LT 'SYREAL' JONES

Description

Can you figure out how this program works to get the flag?
Additional details will be available after launching your challenge instance.

This challenge launches an instance on demand.

Its current status is: `NOT_RUNNING`

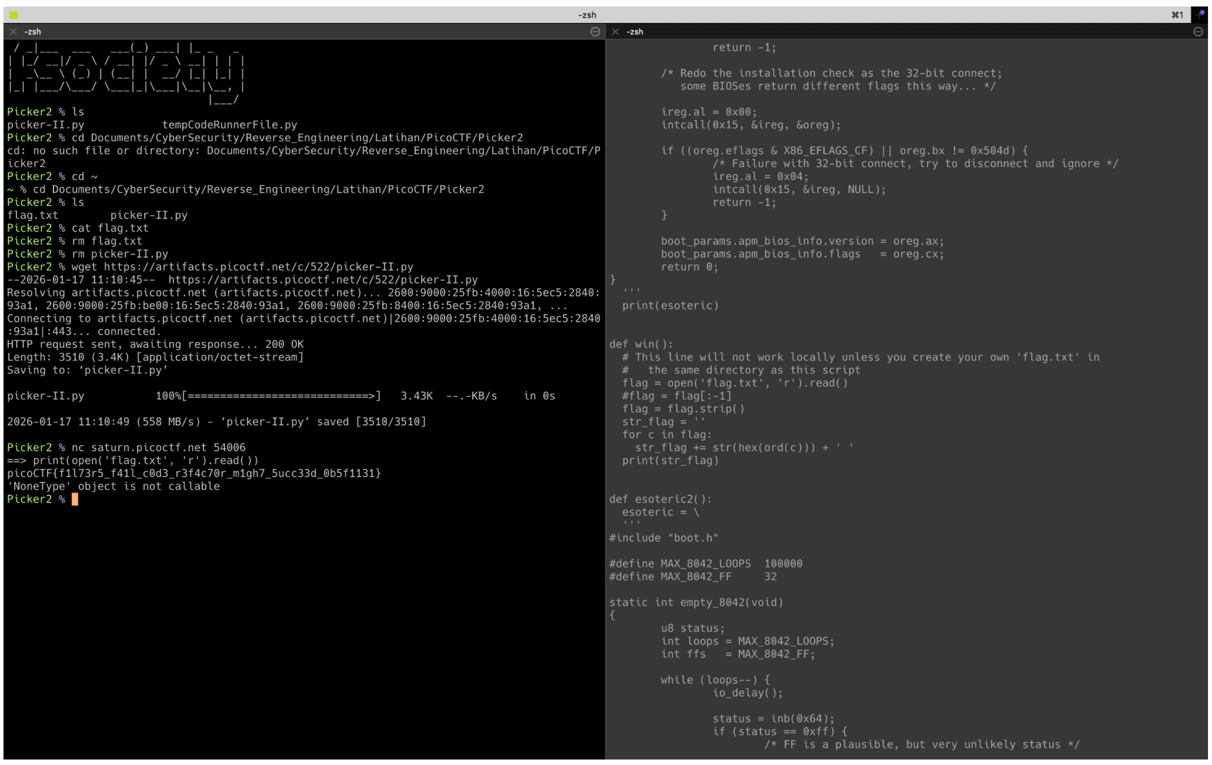
Launch Instance

Hints

1

Can you do what `win` does with your input to the program?

Solusi :



```
 / _|___  ___   ___(_) ___| |_ _   _
| |_/ __|/ _ \ / __| |/ _ \ __| | | |
|  _\__ \ (_) | (__| |  __/ |_| |_| |
|_| |___/\___/ \___|_|\___|\__|\__, |
                               |___/
Picker2 % ls
picker-II.py            tempCodeRunnerFile.py
Picker2 % cd Documents/CyberSecurity/Reverse_Engineering/Latihan/PicoCTF/Picker2
cd: no such file or directory: Documents/CyberSecurity/Reverse_Engineering/Latihan/PicoCTF/P
icker2
Picker2 % cd ~
~ % cd Documents/CyberSecurity/Reverse_Engineering/Latihan/PicoCTF/Picker2
Picker2 % ls
flag.txt        picker-II.py
Picker2 % cat flag.txt
Picker2 % rm flag.txt
Picker2 % rm picker-II.py
Picker2 % wget https://artifacts.picoctf.net/c/522/picker-II.py
--2026-01-17 11:10:45--  https://artifacts.picoctf.net/c/522/picker-II.py
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 2600:9000:25fb:4000:16:5ec5:2840:
93a1, 2600:9000:25fb:be00:16:5ec5:2840:93a1, 2600:9000:25fb:8400:16:5ec5:2840:93a1, ...
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|2600:9000:25fb:4000:16:5ec5:2840
:93a1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3510 (3.4K) [application/octet-stream]
Saving to: ‘picker-II.py’

picker-II.py          100%[============================>]   3.43K  --.-KB/s    in 0s

2026-01-17 11:10:49 (558 MB/s) - ‘picker-II.py’ saved [3510/3510]

Picker2 % nc saturn.picoctf.net 54006
==> print(open('flag.txt', 'r').read())
picoCTF{f1l73r5_f41l_c0d3_r3f4c70r_m1gh7_5ucc33d_0b5f1131}
'NoneType' object is not callable
Picker2 %
```

```
                return -1;

        /* Redo the installation check as the 32-bit connect;
           some BIOSes return different flags this way... */

        ireg.al = 0x00;
        intcall(0x15, &ireg, &oreg);

        if ((oreg.eflags & X86_EFLAGS_CF) || oreg.bx != 0x504d) {
                /* Failure with 32-bit connect, try to disconnect and ignore */
                ireg.al = 0x04;
                intcall(0x15, &ireg, NULL);
                return -1;
        }

        boot_params.apm_bios_info.version = oreg.ax;
        boot_params.apm_bios_info.flags   = oreg.cx;
        return 0;
}
  '''
  print(esoteric)


def win():
  # This line will not work locally unless you create your own 'flag.txt' in
  #   the same directory as this script
  flag = open('flag.txt', 'r').read()
  #flag = flag[:-1]
  flag = flag.strip()
  str_flag = ''
  for c in flag:
    str_flag += str(hex(ord(c))) + ' '
  print(str_flag)


def esoteric2():
  esoteric = \
  '''
#include "boot.h"

#define MAX_8042_LOOPS  100000
#define MAX_8042_FF     32

static int empty_8042(void)
{
        u8 status;
        int loops = MAX_8042_LOOPS;
        int ffs   = MAX_8042_FF;

        while (loops--) {
                io_delay();

                status = inb(0x64);
                if (status == 0xff) {
                        /* FF is a plausible, but very unlikely status */
```