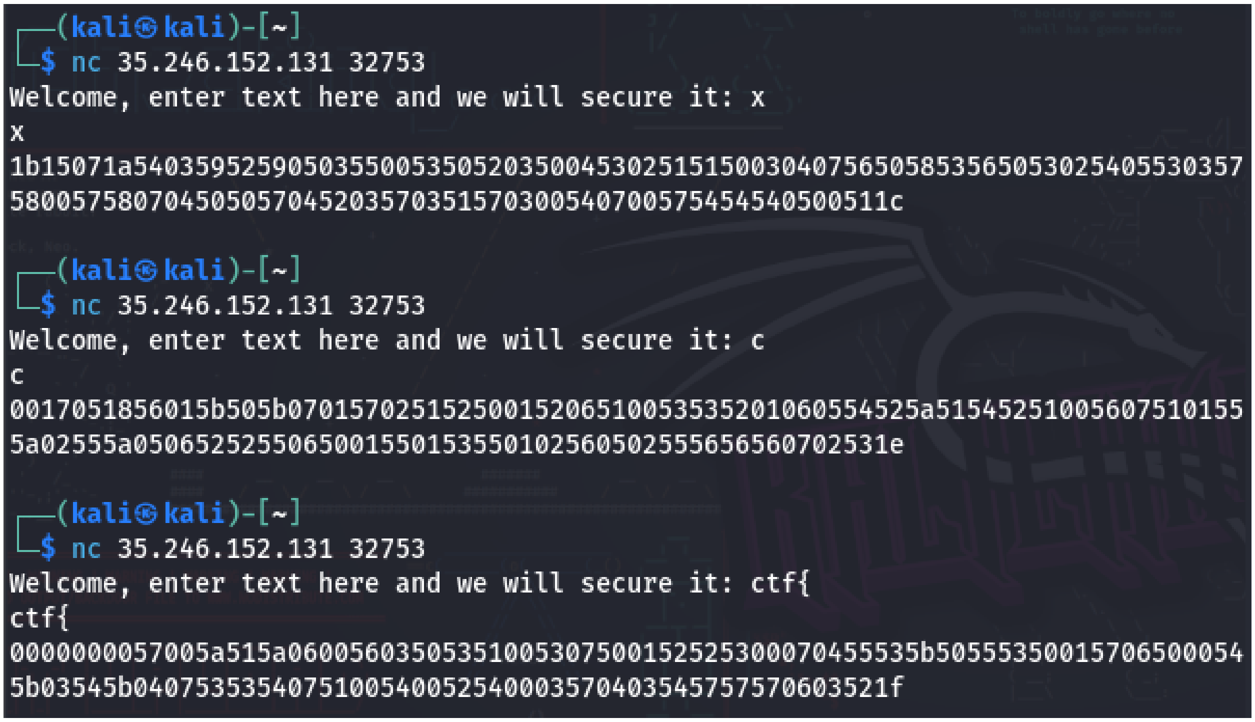
After messing around a bit on the server, I discovered this:

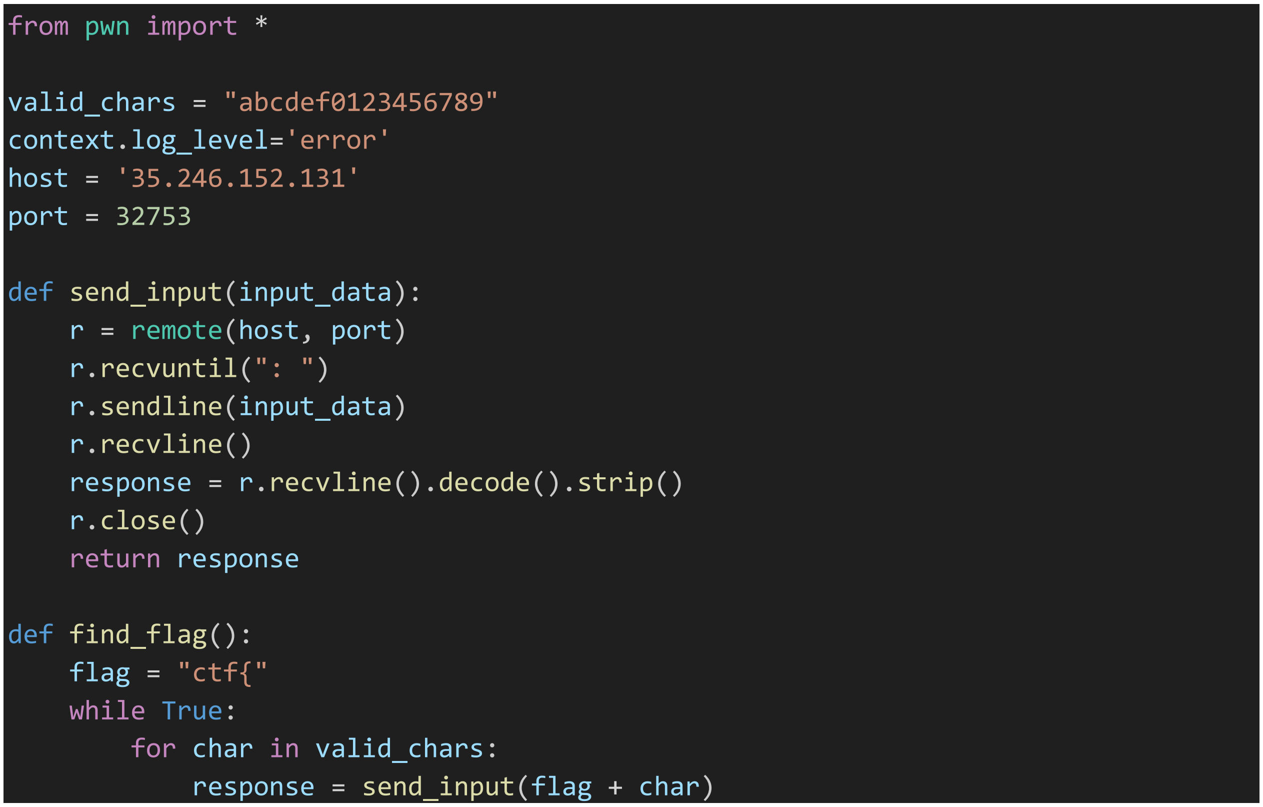


```
┌──(kali㉿kali)-[~]
└─$ nc 35.246.152.131 32753
Welcome, enter text here and we will secure it: x
x
1b15071a5403595259050355005350520350045302515150030407565058535650530254055303 57
58005758070450505704520357035157030054070057545454050051 1c

┌──(kali㉿kali)-[~]
└─$ nc 35.246.152.131 32753
Welcome, enter text here and we will secure it: c
c
0017051856015b505b07015702515250015206510053535201060554525a515452510056075101 55
5a02555a05065252550650015501535501025605025556565607025 31e

┌──(kali㉿kali)-[~]
└─$ nc 35.246.152.131 32753
Welcome, enter text here and we will secure it: ctf{
ctf{
000000005700 5a515a060056035053510053075001525253000704555 35b505553500157065000 54
5b03545b040753535407510054005254000357040354575757060352 1f
```

The server performs some sort of alien xor (pun intended) that works like this: If the character from input is correct, the server returns 00 for that character. A quick solve script looks like this:

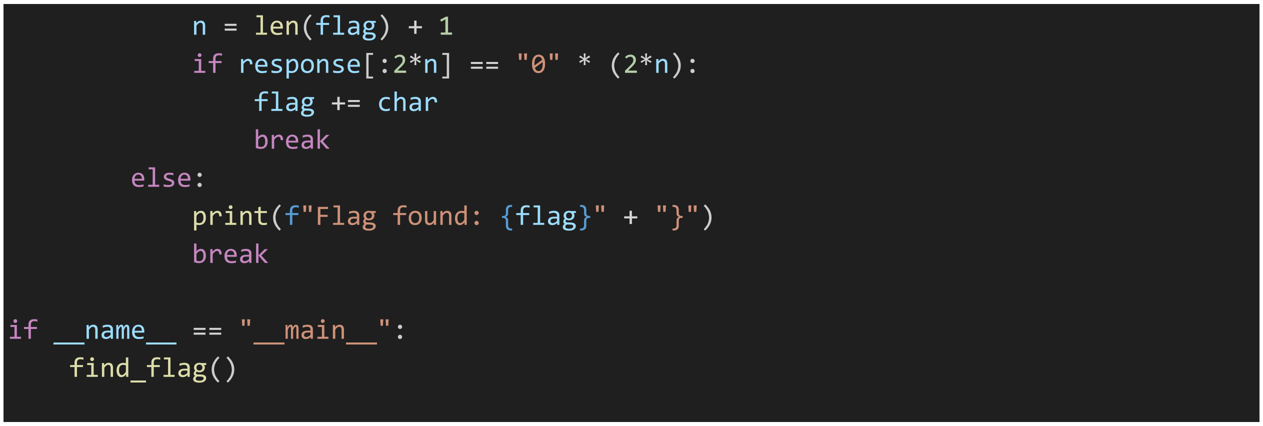


```
from pwn import *

valid_chars = "abcdef0123456789"
context.log_level='error'
host = '35.246.152.131'
port = 32753

def send_input(input_data):
    r = remote(host, port)
    r.recvuntil(": ")
    r.sendline(input_data)
    r.recvline()
    response = r.recvline().decode().strip()
    r.close()
    return response

def find_flag():
    flag = "ctf{"
    while True:
        for char in valid_chars:
            response = send_input(flag + char)
```



```
            n = len(flag) + 1
            if response[:2*n] == "0" * (2*n):
                flag += char
                break
        else:
            print(f"Flag found: {flag}" + "}")
            break

if __name__ == "__main__":
    find_flag()
```

Made with love by: AndreiCat