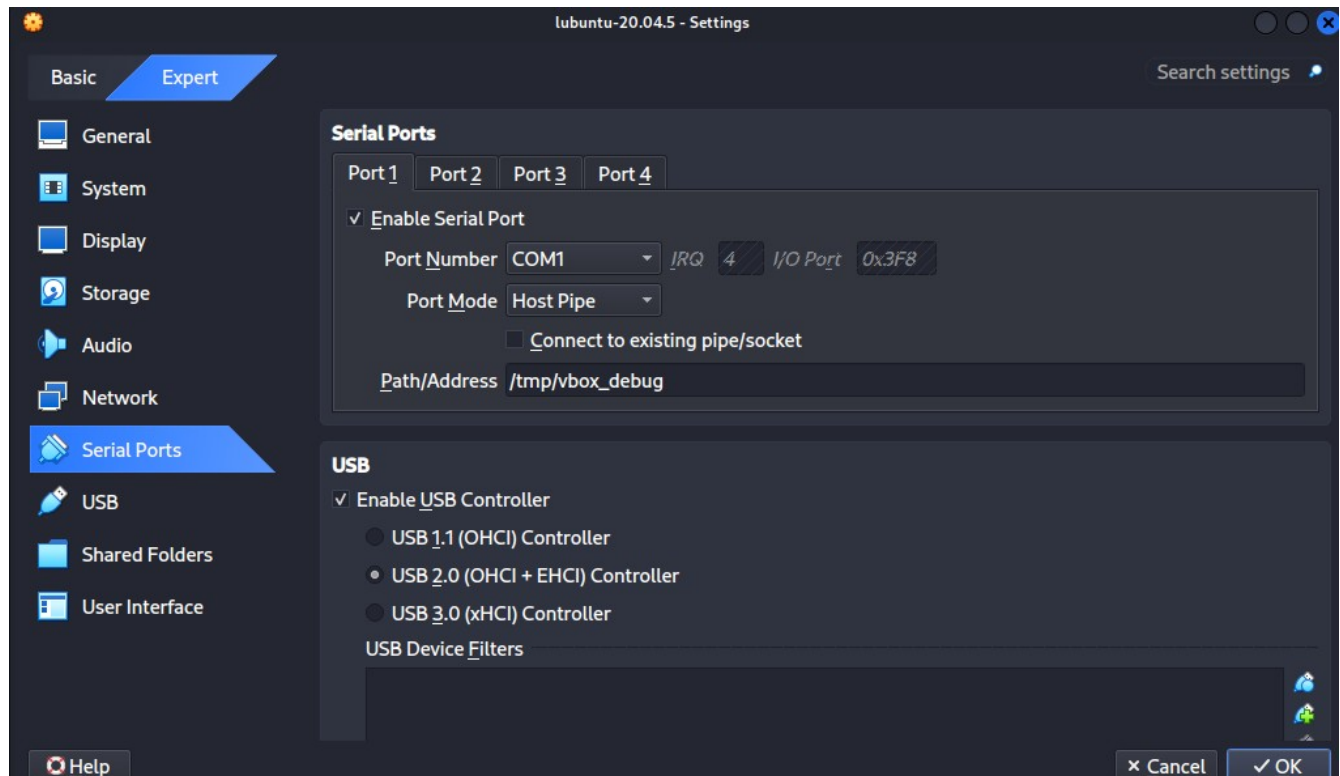


6.2. Teknik debugging Linux Kernel pada virtualbox

Debugging linux kernel diperlukan saat proses pembuatan LKM (Loadable Kernel Module) dan proses exploit development untuk linux kernel.

Untuk debugging linux kernel kita akan menggunakan virtualbox, socat, gdb, gdb-multiarch dan gef. Untuk gdb dan gef sudah diinstall sebelumnya jadi tidak perlu kita atur lagi.

Pertama tama pada virtualbox untuk guest os, pergi ke pengaturan Settings, lalu pilih Serial Ports



- Centang pada bagian Enable Serial Port
- Port Number : COM1
- Port Mode : Host Pipe
- Uncheck pada bagian Connect to existing pipe / socket
- Path/Address : /tmp/vbox_debug
- Klik ok

Selanjutnya jalankan guest os.

Agar bisa mendebug linux kernel di host os, kita perlu mendownload vmlinux dari guest os ke kali linux kita, untuk itu kita bisa upload vmlinux dari guest os ke mesin host os dengan ftp misalnya.

Pada contoh kali ini saya sudah menginstall openssh server di host os, saya menggunakan sftp untuk transfer vmlinux ke host os.

Pada guest os, buka terminal, lalu ke direktori /boot:

```
cd /boot
```

```
ls vmlinuz*
```

Misal isinya :

```
vmlinuz vmlinuz-5.15.0-46-generic vmlinuz.old
```

Yang perlu kita unggah ke host os adalah vmlinuz-5.15.0-46-generic

di sini saya menggunakan sftp :

```
sftp robohax@192.168.56.1
```

```
put vmlinuz-5.15.0-46-generic
```

Setelah vmlinuz didownload ke host os selanjutnya kita atur grub di guest os:

```
sudo nano /etc/default/grub
```

Pada bagian GRUB_CMDLINE_LINUX_DEFAULT, ganti menjadi :

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet nokaslr kgdboc=ttyS0,115200"
```

Selanjutnya simpan, lalu update grub:

```
sudo update-grub
```

Selanjutnya power off guest os.

Kembali ke host os, kita akan menginstall gdb-multiarch. Gdb-multiarch ini adalah versi gdb yang bisa digunakan untuk mendebug berbagai macam arsitektur misal : x86, x64, arm64, mips,dll.

```
sudo apt install gdb-multiarch -y
```

Selanjutnya untuk mempermudah menjalankan gdb-multiarch, buat symlink:

```
sudo ln -s /usr/bin/gdb-multiarch /sbin/gm
```

Install socat di host os jika belum ada !

Langkah selanjutnya start guest os.

Pada saat guest os boot, jika pengaturan kita berhasil maka kgdb di guest os akan menunggu koneksi untuk memulai sesi kernel debugging:

```
[ 0.462511] fuse: init (API version 7.34)
[ 0.462511] integrity: Platform Keyring initialized
[ 0.462511] Key type asymmetric registered
[ 0.462511] Asymmetric key parser 'x509' registered
[ 0.462511] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 243)
[ 0.462511] io scheduler mq-deadline registered
[ 0.462511] shpchp: Standard Hot Plug PCI Controller Driver version: 0.4
[ 0.482719] ACPI: AC: AC Adapter [AC] (off-line)
[ 0.482981] input: Power Button as /devices/LNXSYSTM:00/LNXPWRBN:00/input/input0
[ 0.487379] ACPI: button: Power Button [PWRF]
[ 0.487679] input: Sleep Button as /devices/LNXSYSTM:00/LNXPBPN:00/input/input1
[ 0.488010] ACPI: button: Sleep Button [SLPF]
[ 0.488471] Serial: 8250/16550 driver, 32 ports, IRQ sharing enabled
[ 0.510129] ACPI: battery: Slot [BAT0] (battery present)
[ 0.532318] 00:02: ttyS0 at I/O 0x3f8 (irq = 4, base_baud = 115200) is a 16550A
[ 0.534248] KGDB: Registered I/O driver kgdboc
[ 0.534435] KGDB: Waiting for connection from remote gdb...

Entering kdb (current=0xffff888004249900, pid 1) on processor 0 due to NonMaskable Interrupt @ 0xffffffff811c9364
[0]kdb>
```

Pada host os jalankan socat untuk listen di port 1234 :

```
socat -d -d UNIX-CLIENT:/tmp/vbox_debug TCP-LISTEN:1234
```

Selanjutnya pergi ke direktori di mana terdapat vmlinux yang berasal dari guest os , contoh :

```
cd /home/robohax/Desktop/sploit/kernel-space/SETUP/vmlinux lubuntu 20.04/
```

```
ls -lah
```

```
total 11M
```

```
drwxrwxr-x 2 robohax robohax 4.0K Jan 27 08:39 .
```

```
drwxrwxr-x 5 robohax robohax 4.0K Jan 27 08:38 ..
```

```
-rwxrwxrwx 1 robohax robohax 11M Jan 27 08:39 vmlinux-5.15.0-46-generic
```

```
ketik sudo su
```

Jalankan gdb multiarch :

```
gm ./vmlinux-5.15.0-46-generic
```

Pada console gdb, ketikkan :

target remote :1234

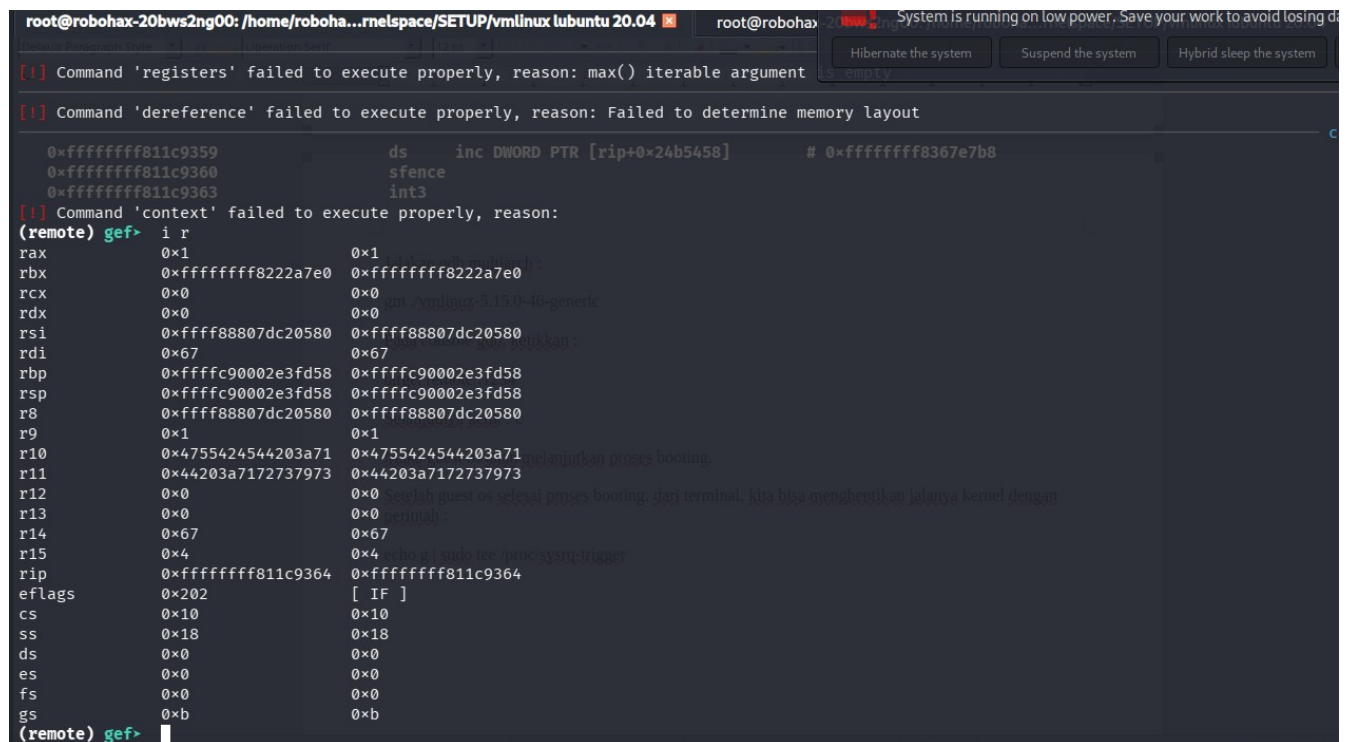
Selanjutnya ketik : c

Maka guest os akan melanjutkan proses booting.

Setelah guest os selesai proses booting, dari terminal, kita bisa menghentikan jalanya kernel dengan perintah :

echo g | sudo tee /proc/sysrq-trigger

Kembali ke jendela gdb di host os, kita bisa melihat informasi yang terdapat pada register kernel di guest os dengan perintah : info registers



```
root@robomax-20bws2ng00: /home/robomax...mlespace/SETUP/vmlinux lubuntu 20.04
[!] Command 'registers' failed to execute properly, reason: max() iterable argument
[!] Command 'dereference' failed to execute properly, reason: Failed to determine memory layout
0xffffffff811c9359          ds      inc DWORD PTR [rip+0x24b5458]      # 0xffffffff8367e7b8
0xffffffff811c9360          sfence
0xffffffff811c9363          int3
[!] Command 'context' failed to execute properly, reason:
(remote) gef> i r
rax          0x1
rbx          0xffffffff8222a7e0 0xffffffff8222a7e0
rcx          0x0
rdx          0x0
rsi          0xffff88807dc20580 0xffff88807dc20580
rdi          0x67
rbp          0xffffc90002e3fd58 0xffffc90002e3fd58
rsp          0xffffc90002e3fd58 0xffffc90002e3fd58
r8           0xffff88807dc20580 0xffff88807dc20580
r9           0x1
r10          0x4755424544203a71 0x4755424544203a71
r11          0x44203a7172737973 0x44203a7172737973
r12          0x0
r13          0x0
r14          0x67
r15          0x4
rip          0xffffffff811c9364 0xffffffff811c9364
eflags       0x202
cs           0x10
ss           0x18
ds           0x0
es           0x0
fs           0x0
gs           0xb
(remote) gef> 
```

Jika ingin menjalankan lagi kernel di guest os, ketik : c