Demo PCI probe using WSL

1) Identify your PCI device using lspci -nn command:

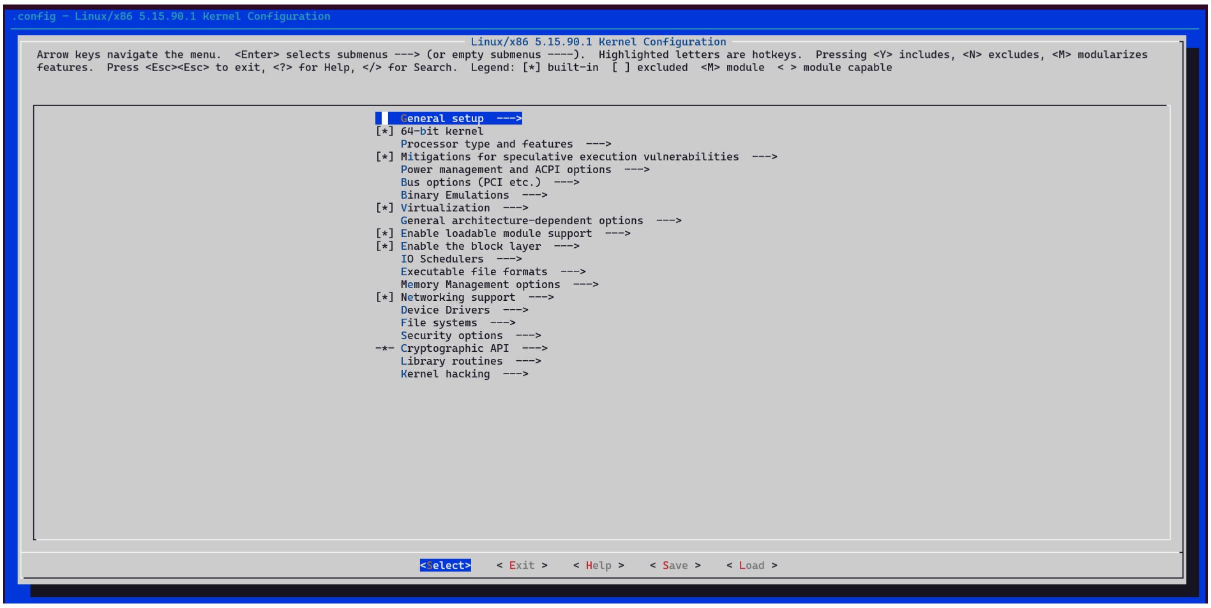
```
root@9077BG3:/home/wa# lspci -nn
2e76:00:00.0 3D controller [0302]: Microsoft Corporation Device [1414:008e]
7581:00:00.0 3D controller [0302]: Microsoft Corporation Device [1414:008e]
```

In my case 0x1414 is VendorID and 0x008e is DeviceID.

2) Identify currently used driver using lspci -vvv command:

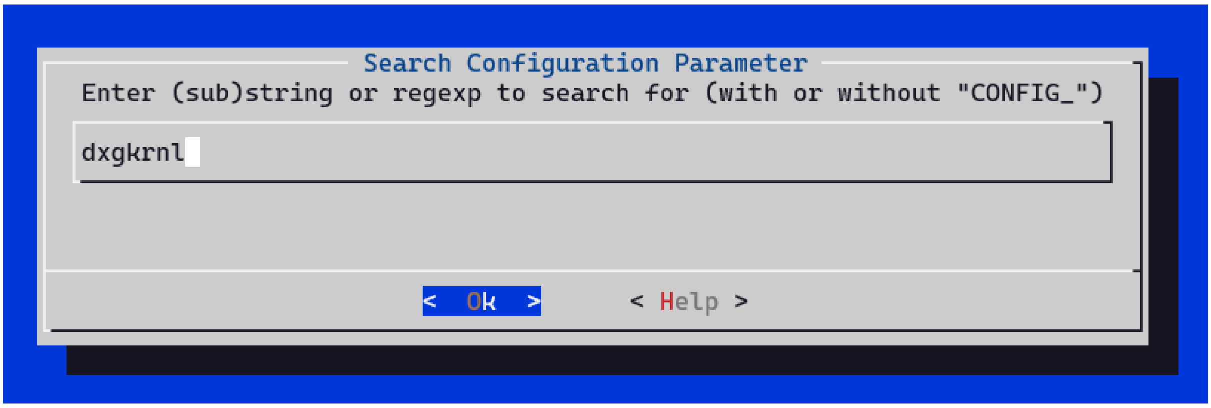
```
root@9077BG3:/home/wa# lspci -vvv
2e76:00:00.0 3D controller: Microsoft Corporation Device 008e
        Physical Slot: 202270445
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr-
Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort-
<TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Capabilities: [40] Null
        Kernel driver in use: dxgkrnl
lspci: Unable to load libkmod resources: error -2

7581:00:00.0 3D controller: Microsoft Corporation Device 008e
        Physical Slot: 1527642535
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr-
Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort-
<TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Capabilities: [40] Null
        Kernel driver in use: dxgkrnl
```

3) In your WSL kernel build directory execute make menuconfig. By default it is not possible, you need to install libncurses-dev

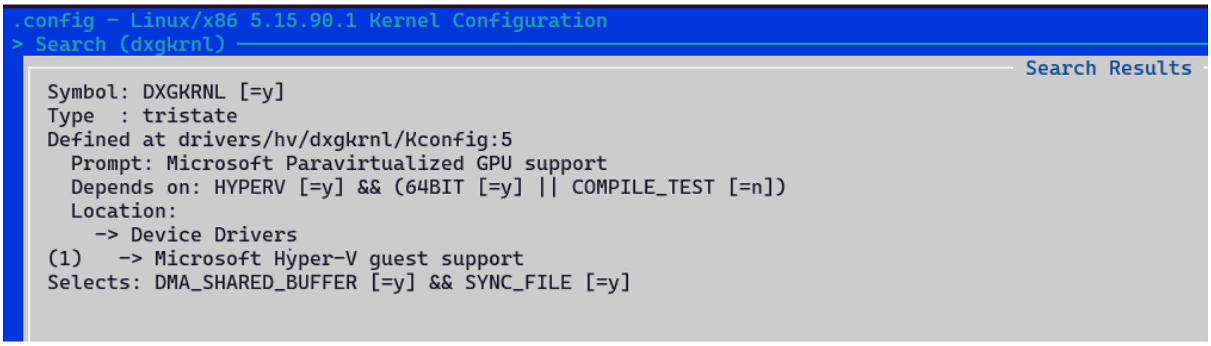


```
.config - Linux/x86 5.15.90.1 Kernel Configuration
                              Linux/x86 5.15.90.1 Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
 features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                    General setup  --->
                [*] 64-bit kernel
                    Processor type and features  --->
                [*] Mitigations for speculative execution vulnerabilities  --->
                    Power management and ACPI options  --->
                    Bus options (PCI etc.)  --->
                    Binary Emulations  --->
                [*] Virtualization  --->
                    General architecture-dependent options  --->
                [*] Enable loadable module support  --->
                [*] Enable the block layer  --->
                    IO Schedulers  --->
                    Executable file formats  --->
                    Memory Management options  --->
                [*] Networking support  --->
                    Device Drivers  --->
                    File systems  --->
                    Security options  --->
                -*- Cryptographic API  --->
                    Library routines  --->
                    Kernel hacking  --->

              <Select>    < Exit >    < Help >    < Save >    < Load >
```

4) Press ‘/’ character on your keyboard and find an interesting driver. In my case it is **dxgkrnl**



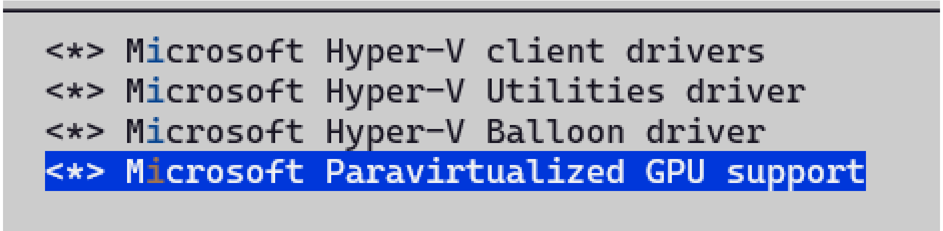
```
Search Configuration Parameter
Enter (sub)string or regexp to search for (with or without "CONFIG_")
dxgkrnl
```



```
.config - Linux/x86 5.15.90.1 Kernel Configuration
> Search (dxgkrnl)
                                                              Search Results
Symbol: DXGKRNL [=y]
Type  : tristate
Defined at drivers/hv/dxgkrnl/Kconfig:5
  Prompt: Microsoft Paravirtualized GPU support
  Depends on: HYPERV [=y] && (64BIT [=y] || COMPILE_TEST [=n])
  Location:
    -> Device Drivers
(1)   -> Microsoft Hyper-V guest support
Selects: DMA_SHARED_BUFFER [=y] && SYNC_FILE [=y]
```

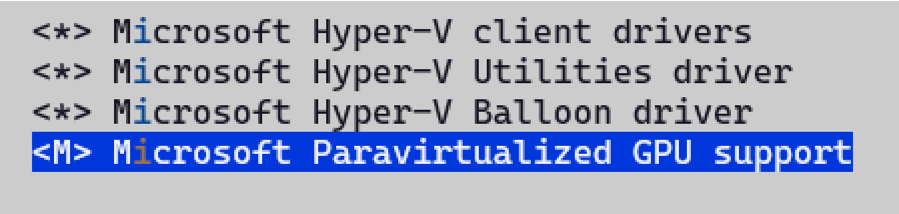
As you can see the driver was found, it is selected [=y] and it is located under Device Drivers -> Microsoft Hyper-V guest support. Go there using keyboard arrows from the main menu and select it using the keyboard space key until you change it to module.

Before:



```
<*> Microsoft Hyper-V client drivers
<*> Microsoft Hyper-V Utilities driver
<*> Microsoft Hyper-V Balloon driver
<*> Microsoft Paravirtualized GPU support
```

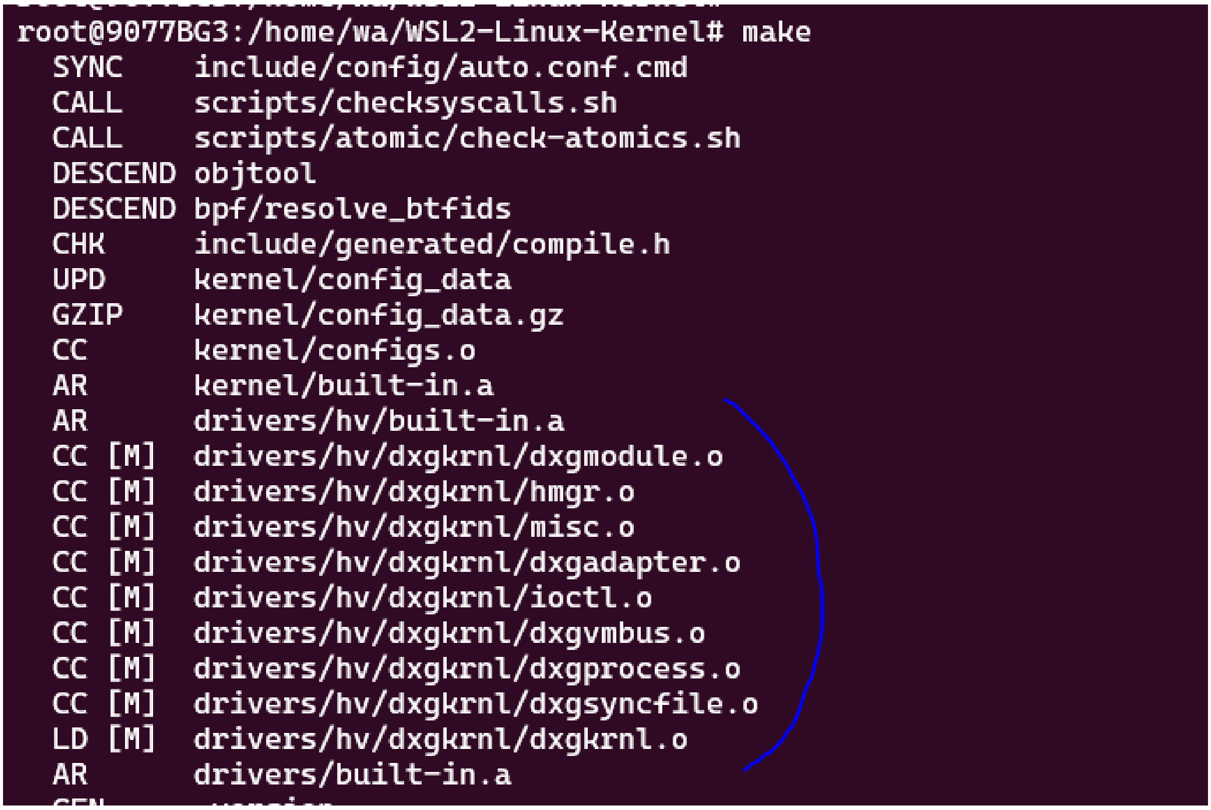
After:



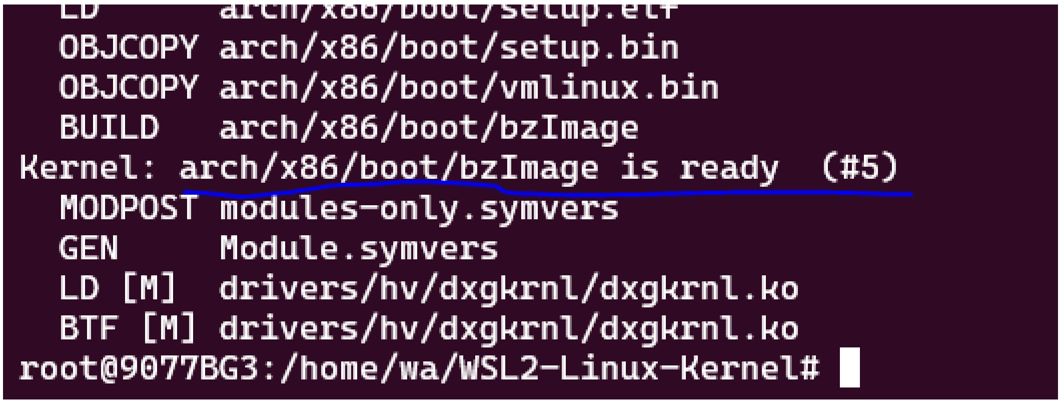
```
<*> Microsoft Hyper-V client drivers
<*> Microsoft Hyper-V Utilities driver
<*> Microsoft Hyper-V Balloon driver
<M> Microsoft Paravirtualized GPU support
```

Drivers compiled in the kernel have priority in probing so until you disable the existing driver it is impossible to take control over the device.

5) Compile kernel using make command:



```
root@9077BG3:/home/wa/WSL2-Linux-Kernel# make
  SYNC    include/config/auto.conf.cmd
  CALL    scripts/checksyscalls.sh
  CALL    scripts/atomic/check-atomics.sh
  DESCEND objtool
  DESCEND bpf/resolve_btfids
  CHK     include/generated/compile.h
  UPD     kernel/config_data
  GZIP    kernel/config_data.gz
  CC      kernel/configs.o
  AR      kernel/built-in.a
  AR      drivers/hv/built-in.a
  CC [M]  drivers/hv/dxgkrnl/dxgmodule.o
  CC [M]  drivers/hv/dxgkrnl/hmgr.o
  CC [M]  drivers/hv/dxgkrnl/misc.o
  CC [M]  drivers/hv/dxgkrnl/dxgadapter.o
  CC [M]  drivers/hv/dxgkrnl/ioctl.o
  CC [M]  drivers/hv/dxgkrnl/dxgvmbus.o
  CC [M]  drivers/hv/dxgkrnl/dxgprocess.o
  CC [M]  drivers/hv/dxgkrnl/dxgsyncfile.o
  LD [M]  drivers/hv/dxgkrnl/dxgkrnl.o
  AR      drivers/built-in.a
  GEN     version
```



```
  LD      arch/x86/boot/setup.elf
  OBJCOPY arch/x86/boot/setup.bin
  OBJCOPY arch/x86/boot/vmlinux.bin
  BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready  (#5)
  MODPOST modules-only.symvers
  GEN     Module.symvers
  LD [M]  drivers/hv/dxgkrnl/dxgkrnl.ko
  BTF [M] drivers/hv/dxgkrnl/dxgkrnl.ko
root@9077BG3:/home/wa/WSL2-Linux-Kernel#
```

6) Copy your kernel to your Windows Platform:

```
root@9077BG3:/home/wa/WSL2-Linux-Kernel# cp arch/x86/boot/bzImage
/mnt/c/Users/woab/bzImage_pci
```

7) Adjust your WSL config file to point to new kernel:

```
[wsl2]
kernel=C:\\Users\\woab\\bzImage_pci
```

8) Restart WSL:

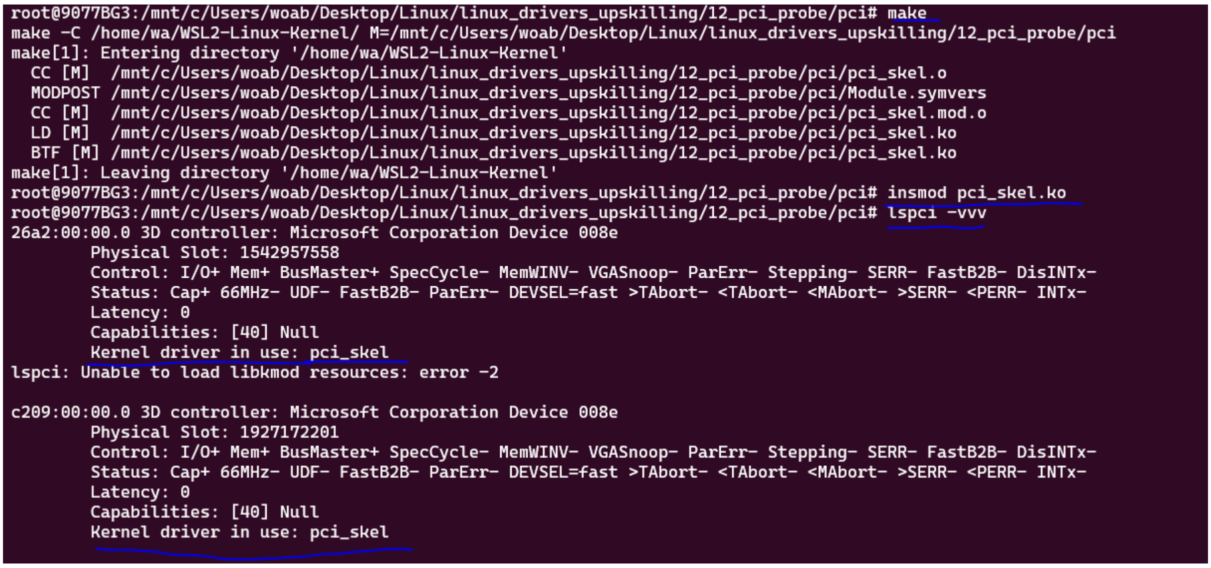
```
PS C:\Users\woab> wsl.exe --shutdown
PS C:\Users\woab>
```

9) Adjust pci_skel.c file with your PCI Device VendorID and DeviceID

```
static struct pci_device_id ids[] = {
      //{ PCI_DEVICE(PCI_ANY_ID, PCI_ANY_ID), },
      { PCI_DEVICE(0x1414, 0x008e), },
        { 0, }
};
```

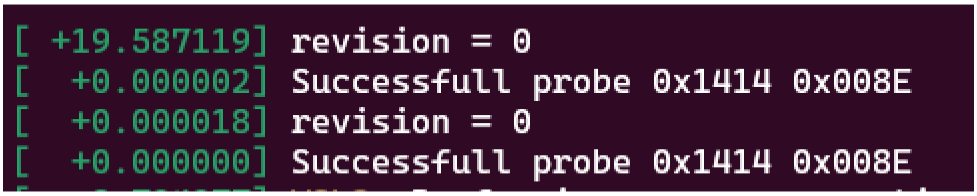
You can use PCI_ANY_ID to probe any vendor and device.

10) Now compile and insmod module:



```
root@9077BG3:/mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci# make
make -C /home/wa/WSL2-Linux-Kernel/ M=/mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci
make[1]: Entering directory '/home/wa/WSL2-Linux-Kernel'
  CC [M]  /mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci/pci_skel.o
  MODPOST /mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci/Module.symvers
  CC [M]  /mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci/pci_skel.mod.o
  LD [M]  /mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci/pci_skel.ko
  BTF [M] /mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci/pci_skel.ko
make[1]: Leaving directory '/home/wa/WSL2-Linux-Kernel'
root@9077BG3:/mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci# insmod pci_skel.ko
root@9077BG3:/mnt/c/Users/woab/Desktop/Linux/linux_drivers_upskilling/12_pci_probe/pci# lspci -vvv
26a2:00:00.0 3D controller: Microsoft Corporation Device 008e
        Physical Slot: 1542957558
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Capabilities: [40] Null
        Kernel driver in use: pci_skel
lspci: Unable to load libkmod resources: error -2

c209:00:00.0 3D controller: Microsoft Corporation Device 008e
        Physical Slot: 1927172201
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Capabilities: [40] Null
        Kernel driver in use: pci_skel
```

Dmesg output:



```
[ +19.587119] revision = 0
[  +0.000002] Successfull probe 0x1414 0x008E
[  +0.000018] revision = 0
[  +0.000000] Successfull probe 0x1414 0x008E
```

As you can see it was possible to force the existing PCI device to be handled by our driver.

11)