Si ya has leído y ejecutado este hack donde explicaba como pasar una tarjeta PCI o PCIe a una máquina virtual de ProxmoxVE tienes gran parte de la tarea hecha. Si no lo has hecho, sigue ese tutorial, y cuando lo acabes vuelve a éste.

Te habrás dado cuenta, entonces, que en ese minitutorial no expliqué nada sobre las tarjetas gráficas. Y fue porque las tarjetas gráficas deben ser pasadas de una forma diferente a como pasas una tarjeta PCI o PCIe normal. Las gráficas, a diferencia de cualquier otro dispositivo PCI o PCIe, son dispositivos que Linux intentará utilizar de todas, todas. Por lo que, antes de pasar la tarjeta a la MV hay que:

1. Utilizar el módulo **vfio** para que «capture» la tarjeta antes que los módulos gráficos oficiales intenten cargarla.
2. **Blacklistear** (por las dudas) los módulos que las controlan para evitar que el driver tenga que «bindear» y «des-bindear» la tarjeta cada vez que se inicia o se apaga la máquina virtual

Así que vamos a proceder con el paso 1

INDICAR A VFIO EL IDENTIFICADOR DE LA TARJETA GRÁFICA A «CAPTURAR»

Tenemos que saber que identificador de dispositivo tiene la tarjeta que debemos pasar antes de decírselo al módulo vfio. Y no hay mejor forma de hacerlo que con un nipe-script. Teniendo en cuenta que ya has instalado los p-scripts y has seguido todos los pasos para configurar PCI passtrough en ProxmoxVE explicados en este minitutorial, en vez de utilizar **lspci** ejecuta el siguiente nipe-script:

```
/root/scripts/d-scripts/IOMMU-Grupos-Mostrar.sh
```

La salida será algo parecido a esta:

```
----------------------------------------
Grupos IOMMU disponibles en el sistema:
----------------------------------------

Grupo IOMMU 17
04:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP108 [GeForce GT 1030] [10de:1d01] (rev a1)
04:00.1 Audio device [0403]: NVIDIA Corporation GP108 High Definition Audio Controller [10de:0fb8] (rev a1)

Grupo IOMMU 7
00:08.0 Encryption controller [1080]: Advanced Micro Devices, Inc. [AMD] Device [1022:1578]

Grupo IOMMU 15
02:00.0 Network controller [0280]: Qualcomm Atheros AR93xx Wireless Network Adapter [168c:0030] (rev 01)

Grupo IOMMU 5
00:03.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:157b]

Grupo IOMMU 13
00:18.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1570]
00:18.1 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1571]
00:18.2 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1572]
00:18.3 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1573]
00:18.4 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1574]
00:18.5 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1575]

Grupo IOMMU 3
00:02.4 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Device [1022:157c]

Grupo IOMMU 11
00:12.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] FCH USB EHCI Controller [1022:7908] (rev 49)

Grupo IOMMU 1
00:02.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:157b]

Grupo IOMMU 8
00:09.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:157d]

Grupo IOMMU 16
03:00.0 Ethernet controller [0200]: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe [14e4:165f]
```

```
03:00.1 Ethernet controller [0200]: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe [14e4:165f]

Grupo IOMMU 6
00:03.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Device [1022:157c]

Grupo IOMMU 14
01:00.0 SATA controller [0106]: Marvell Technology Group Ltd. 88SE9230 PCIe SATA 6Gb/s Controller [1b4b:9230] (rev
11)

Grupo IOMMU 4
00:02.5 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Device [1022:157c]

Grupo IOMMU 12
00:14.0 SMBus [0c05]: Advanced Micro Devices, Inc. [AMD] FCH SMBus Controller [1022:790b] (rev 4a)
00:14.3 ISA bridge [0601]: Advanced Micro Devices, Inc. [AMD] FCH LPC Bridge [1022:790e] (rev 11)

Grupo IOMMU 2
00:02.2 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Device [1022:157c]

Grupo IOMMU 10
00:11.0 SATA controller [0106]: Advanced Micro Devices, Inc. [AMD] FCH SATA Controller [AHCI mode] [1022:7901] (rev
49)

Grupo IOMMU 0
00:01.0 VGA compatible controller [0300]: Advanced Micro Devices, Inc. [AMD/ATI] Carrizo [1002:9874] (rev 84)
00:01.1 Audio device [0403]: Advanced Micro Devices, Inc. [AMD/ATI] Kabini HDMI/DP Audio [1002:9840]

Grupo IOMMU 9
00:10.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] FCH USB XHCI Controller [1022:7914] (rev 20)
```

De ahí se puede observar que el grupo 17 tiene una tarjeta nVidia con su correspondiente tarjeta de audio y el grupo 0 tiene otra tarjeta distinta. Además se puede observar también que una tarjeta es identificada como 00:00 (la del grupo 0) y la otra como 04:00 (la del grupo 17). La que tiene más baja numeración es siempre la tarjeta gráfica integrada en el procesador o en la APU (siempre que cuente con una). Las otras serían las «pinchadas». Entonces, en la salida de arriba vemos que hay una tarjeta externa con id 04:00, que además pone nVidia y que justamente es la que queremos pasarle a la máquina virtual. Si vamos hasta el final de cada línea, tanto de la gráfica como de su tarjeta de audio, encontraremos la información del **vendor** y del **device**. En este caso:

```
10de:1d01 (rev a1)
10de:ofb8 (rev a1)
```

Ahora vamos a pasárselo al vfio mediante un archivo de configuración en /etc/modprobe.d/. Para ello ejecuta como root:

```
echo "options vfio-pci ids=10de:1d01,10de:0fb8" > /etc/modprobe.d/pci-passthrough.conf
```

Y vfio ya quedará configurado.

Si tienes dos tarjetas gráficas, una para el host y otra para pasar a las máquinas virtuales, te interesará agregar **disable_vga=1** al final de la línea del archivo pci-passthrough.conf para que el módulo de la tarjeta gráfica del host obtenga acceso exclusivo a los gráficos.

El contenido del archivo te quedaría, por ejemplo, asi:

```
blacklist amdgpu
options vfio-pci ids=1002:67ef,1002:aae0 disable_vga=1
```

Ahora vamos con el paso 2:

BLACKLISTEAR EN EL HOST EL MÓDULO QUE CONTROLA LA GRÁFICA

Para saber que módulo gráfico es el que tenemos que blacklistear esta vez si que podemos usar lspci. Para ello ejecutamos:

```
lspci -k
```

... y la parte de la salida referente a la gráfica será algo como esto:

```
04:00.0 VGA compatible controller: NVIDIA Corporation GP108 [GeForce GT 1030] (rev a1)
        Subsystem: Micro-Star International Co., Ltd. [MSI] GP108
        Kernel driver in use: nouveau
        Kernel modules: nvidiafb, nouveau
04:00.1 Audio device: NVIDIA Corporation GP108 High Definition Audio Controller (rev a1)
        Subsystem: Micro-Star International Co., Ltd. [MSI] Device 8c98
        Kernel driver in use: snd_hda_intel
        Kernel modules: snd_hda_intel
```

Donde lo que vemos, en este caso, es que los módulos gráficos controlando la tarjeta nVidia son:

```
nvidiafb, nouveau
```

El módulo nvidiafb ya está blacklisteado por defecto en Proxmox dado que tiene bugs conocidos. Lo podemos ver en el archivo /etc/modprobe.d/pve-blacklist.conf, que es el archivo de blacklists por defecto de Proxmox. Entonces, nos quedará agregar sólo nouveau a la «lista negra». Para ello ejecutamos:

```
echo "blacklist nouveau" > /etc/modprobe.d/pci-passthrough.conf
```

Una vez agregado el módulo a la lista negra tendremos que regenerar la initramfs. Para ello ejecutamos como root:

```
update-initramfs -u -k all
```

Y reiniciamos el sistema con:

```
shutdown -r now
```

Después de reiniciar, y antes de intentar pasarle la tarjeta gráfica a la máquina virtual, no estaría de más controlar que el módulo vfio sea el que esté controlando los dos dispositivos que le hemos indicado antes. Para ello, volvemos a ejecutar:

```
lspci -k
```

... y esta vez, en vez de ser nouveau el driver usado para controlar la gráfica, será el propio vfio, como muestra esta salida:

```
04:00.0 VGA compatible controller: NVIDIA Corporation GP108 [GeForce GT 1030] (rev a1)
        Subsystem: Micro-Star International Co., Ltd. [MSI] GP108
        Kernel driver in use: vfio-pci
        Kernel modules: nvidiafb, nouveau
04:00.1 Audio device: NVIDIA Corporation GP108 High Definition Audio Controller (rev a1)
        Subsystem: Micro-Star International Co., Ltd. [MSI] Device 8c98
        Kernel driver in use: vfio-pci
        Kernel modules: snd_hda_intel
```

Eso significa que vfio ya tiene la gráfica capturada y lista para pasársela a una máquina virtual, con lo que ya podemos proceder a hacerlo. Aquí se abren 4 posibilidades:

Posibilidad 1: Máquina virtual BIOS, gráfica PCI

Se indica en Internet para este tipo de VGA passthrough configurar la máquina virtual con:

```
hostpci0: 04:00,x-vga=on
```

… pero KK. No funciona.

Posibilidad 2: Máquina virtual BIOS, gráfica PCIe

Se indica en Internet para este tipo de VGA passthrough configurar la máquina virtual con:

```
machine: q35
hostpci0: 04:00,pcie=1,x-vga=on
```

… pero KK. No funciona. Porque aparentemente la gráfica debe tener una rom de video de tipo 1, es decir, no compatible con UEFI. Pero ni eso, da miles de problemas y no funciona.

Posibilidad 3: Máquina virtual UEFI, tarjeta PCI

Para las máquinas virtuales UEFI las tarjetas gráficas tienen que tener una rom de tipo 3. Es decir, una rom de video compatible con UEFI. Y eso se antoja demasiado moderno para la tecnología PCI por lo que no creo que encuentres muchas tarjetas gráficas PCI con rom de video de tipo 3. Por no decir ninguna.

Posibilidad 4: Máquina virtual EUFI, tarjeta PCIe

La única que funciona. Hasta ahora probada en máquina virtual de ProxmoxVE con Windows 10 y con Ubuntu. La gráfica debe tener una rom de video de tipo 3. Es decir, la rom de video debe tener compatibilidad con UEFI. Al archivo de configuración de la máquina virtual de Windows 10 o de Ubuntu se le deben agregar estos cambios (en el caso de la tarjeta de arriba):

```
bios: ovmf
machine: q35
hostpci0: 04:00,pcie=1,x-vga=on
```

Y después de guardar los cambios y de iniciar la máquina virtual, el monitor que tengas conectado a la gráfica que externa mostrará el escritorio de la máquina virtual a la que se le has pasado. Faena heavy donde las haya, pero desafío conseguido. No son todos los operativos posibles, ni todas las versiones de Windows que nos gustaría. Pero de todos modos ¿quién a día de hoy no utiliza Windows 10?

EXTRA 1: A la fecha de redacción de este hack, la combinación QEMU/KVM no es compatible con la funcionalidad Resizable BAR mediante la que el procesador podria acceder a toda la memoria VRAM de la tarjeta gráfica. Si en máquinas virtuales de Windows el sistema nos deshabilita la tarjeta gráfica (Código 43), es aconsejable desactivar Re-size BAR (o Above 4G decoding) en la BIOS. Esto solucionará inmediatamente el problema del código 43.

EXTRA 2: Es posible indicar a la máquina virtual que utilice un VBIOS específico, distinto al que viene de stock con la tarjeta gráfica, de forma que, en esa máquina virtual específica, la gráfica se pase con un VBIOS diferente. Esto lo hacemos indicando el parámetro:

`,romfile=YestonRX560TechPowerUp.rom`

..en el archivo de configuración de la máquina virtual, de forma que la línea nos podría quedar así:

`hostpci0: 04:00,pcie=1,romfile=YestonRX560TechPowerUp.rom,x-vga=on`

Luego, deberíamos ubicar ese archivo .rom en una carpeta específica del host proxmox. Esta sería:

`/usr/share/kvm/`

EXTRA 3: Si pasamos una tarjeta gráfica de AMD y configuramos el display de la máquina virtual como «ninguno» (none), no hace falta indicar que la tarjeta gráfica se pase como Primary GPU (x-vga=on). Eso está pensado principalmente para tarjetas NVidia.

EXTRA 4: ROM-BAR hace que la máquina virtual pueda ver el VBIOS de la gráfica de forma que, por ejemplo, podamos verlo con GPU-Z Algunos dispositivos que no son tarketas gráfcas puede que necesiten esto deshabilitado.