

Heiko's Blog

Just some thoughts

Blacklisting Graphics Driver

When running a VM with GPU passthrough, that GPU should be bound to the VFIO driver. To make this happen, we need to prevent the regular graphics driver from binding to the passthrough GPU and instead bind the vfio-pci driver.

The most common way to do this is by blacklisting the graphics driver. This works in most cases, but what if you need the graphics driver for another GPU, e.g. the host GPU?

A much better way is to [create a module alias](#) for the device to pass through, or to use the “driver override feature” in conjunction with the initial RAM disk.

Table of Contents



- [1. Determine the PCI IDs of the passthrough devices](#)
- [2. Using a “module alias”](#)
- [3. Using the “driver_override” feature](#)
- [4. Using the “driver_override” feature in conjunction with the initial RAM disk \(initramfs\)](#)
- [5. Blacklisting the graphics driver](#)

Determine the PCI IDs of the passthrough devices

First we need to find the PCI ID(s) of the graphics card and perhaps other

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use. To find out more, including how to control cookies, see here: [Cookie Policy](#)

and PCI IDs, enter:

```
lspci | grep VGA
```

Here is the output on my system:

```
01:00.0 VGA compatible controller: NVIDIA Corporation GF106GL
[Quadro 2000] (rev a1)
02:00.0 VGA compatible controller: NVIDIA Corporation Device 13c2
(rev a1)
```

The first card under 01:00.0 is the Quadro 2000 I use for the Linux host. The other card under 02:00.0 I want to pass through to the guest.

Let's find its PCI IDs:

```
lspci -nn | grep 02:00.
```

Substitute “02:00.” with the bus number of the graphics card you wish to pass through, without the trailing “0”. This way we get the PCI IDs for both the graphics device and the audio device residing on the graphics card. Here is the output on my computer:

```
02:00.0 VGA compatible controller [0300]: NVIDIA Corporation
Device [10de:13c2] (rev a1)
02:00.1 Audio device [0403]: NVIDIA Corporation Device
[10de:0fbb] (rev a1)
```

Write down the **bus numbers** (in our example 02:00.0 and 02:00.1), as well as the **PCI IDs** (10de:13c2 and 10de:0fbb). We need them both in the next part.

Below I describe 4 methods to bind the GPU to the vfio-pci driver. My preferred method is option 3 “Using the “driver_override” feature in conjunction with the initial RAM disk (initramfs)”.

Using a “module alias”

Binding the vfio-pci driver to the passthrough device using a **module alias** has

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

Run the following command:

```
cat /sys/bus/pci/devices/0000:02:00.0/modalias
```

where “02:00.0” is the PCI bus number of your graphics card, preceded by “0000:”. The output will look something like:

```
pci:v000010DEd000013C2sv00001458sd00003679bc03sc00i00
```

Repeat above command with the PCI bus number of the audio part:

```
cat /sys/bus/pci/devices/0000:02:00.1/modalias
```

where 0000:02:00.1 is the PCI bus number of your graphics card’ audio device.

Open or create /etc/modprobe.d/local.conf:

```
xed admin:///etc/modprobe.d/local.conf
```

and copy and paste the results from the two cat /sys/... commands above. Then precede the lines with “alias” and append the lines with “vfio-pci”, as shown below:

```
alias pci:v000010DEd000013C2sv00001458sd00003679bc03sc00i00 vfio-pci
```

```
alias pci:v000010DEd00000FBBsv00001458sd00003679bc04sc03i00 vfio-pci
```

At the end of that file, add the following line:

```
options vfio-pci ids=10de:13c2,10de:0fbb
```

where 10de:13c2 and 10de:0fbb are the PCI IDs for your graphics card’ VGA and audio part.

When using a **4.1 or newer kernel** and boot the guest using UEFI (OVMF), you can also add the following option below the options vfio-pci entry:

```
options vfio-pci disable_vga=1
```

The above entry helps [prevent VGA arbitration from interfering with host devices](#).

Save the file and exit the editor.

Update the initramfs by running:

```
sudo update-initramfs -u
```

Help: Please provide feedback on how it worked or didn't work for you by posting a comment below. Thanks!

Using the “driver_override” feature

If you have **identical graphics cards** for both host and passthrough guest, use the driver_override feature of newer kernels as described below.

Create the driver-override script file:

```
xed admin:///sbin/vfio-pci-override-vga.sh
```

and copy/paste the following script:

```
#!/bin/sh
```

```
DEVS="0000:02:00.0 0000:02:00.1"
```

```
for DEV in $DEVS; do
```

```
echo "vfio-pci" > /sys/bus/pci/devices/$DEV/driver_override
```

```
done
```

```
modprobe -i vfio-pci
```

where “0000:02:00.0 0000:02:00.1” are the PCI bus numbers (graphics and audio part) of the graphics card you wish to pass through.

Make the script executable:

```
sudo chmod 755 /sbin/vfio-pci-override-vga.sh
```

Open or create /etc/modprobe.d/local.conf:

```
xed admin:///etc/modprobe.d/local.conf
```

and paste the following line:

```
install vfio-pci /sbin/vfio-pci-override-vga.sh
```

Update the initramfs by running:

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use. To find out more, including how to control cookies, see here: [Cookie Policy](#)

Reboot your PC.

For more information, see [VFIO GPU How To series, part 3 – Host configuration](#) and [how to get two identical GPUs sorted for KVM using vfio-pci](#). All credit goes to Alex Williamson and Linux Mint forum member O__Shovah!

Using the “driver__override” feature in conjunction with the initial RAM disk (initramfs)

In the past I used both the driver__override feature as well as the initramfs approach, but never together. A combination of both methods, however, works very well. The method is described in my new “[Creating a Windows 10 VM on the AMD Ryzen 9 3900X using Qemu 4.0 and VGA passthrough](#)” tutorial.

Blacklisting the graphics driver

An alternative to the above methods is **blacklisting the graphics driver** in the /etc/default/grub file. This is one of the most common methods, but it has its downsides:

- If you have two (different) graphics cards from the same vendor, you might want to use the blacklisted graphics driver for the Linux host.
- A service like nvidia-fallback.service may be running and needs to be disabled.
- Blacklisting a graphics driver doesn't always work. Sometimes additional steps are needed to ensure that the graphics driver isn't loaded.

Although I don't use this method anymore, I will describe it here for reference. I am also adding some steps in case you encounter problems.

Edit the /etc/default/grub file:

```
xed admin:///etc/default/grub
```

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on"
```

We need to add one of the following options to this line, depending on your hardware:

Nvidia graphics card for Windows VM:

```
modprobe.blacklist=nouveau
```

or with some newer models:

```
nouveau.modeset=0
```

Note: This blacklists the Nouveau driver and prevents it from loading. If you run two Nvidia cards and use the open Nouveau driver for your Linux host, DON'T blacklist the driver!!! Chances are the [tutorial](#) will work since the vfio-pci driver should grab the graphics card before nouveau takes control of it. The same goes for AMD cards (see below).

On some platforms, the `modprobe.blacklist=nouveau` option won't be enough. A service called `nvidia-fallback.service` may be running and needs to be disabled. To check for this service, use:

```
systemctl list-units --type service | grep -i nvidia-fallback
```

If the `nvidia-fallback` service shows up, disable it via:

```
systemctl disable nvidia-fallback.service
```

(Thanks Woody!)

AMD graphics card for Windows VM:

```
modprobe.blacklist=radeon
```

or

```
modprobe.blacklist=amdgpu
```

depending on which driver loads when you boot.

After editing, the `/etc/default/grub` file should look similar to mine:

```
GRUB_DEFAULT=0
```

```
GRUB_TIMEOUT_STYLE=countdown
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="modprobe.blacklist=nouveau quiet
intel_iommu=on"
GRUB_CMDLINE_LINUX=""
...
```

Don't forget to run update-grub:

```
sudo update-grub
```

Open or create /etc/modprobe.d/local.conf:

```
xed admin:///etc/modprobe.d/local.conf
```

At the end of that file, add the following line:

```
options vfio-pci ids=10de:13c2,10de:0fbb
```

where 10de:13c2 and 10de:0fbb are the PCI IDs for your graphics card' VGA and audio part (see [Determine the PCI IDs of the passthrough devices](#)).

Save the file and exit the editor.

Update the initramfs using:

```
sudo update-initramfs -u
```

Reboot, then check if it worked:

```
lspci -nk
```

Go through the output and look for the graphics card you wish to pass through.

The output should be similar to this:

```
02:00.0 0300: 10de:13c2 (rev a1)
```

```
Subsystem: 1458:3679
```

Kernel driver in use: vfio-pci

```
Kernel modules: nvidiafb, nouveau, nvidia_drm, nvidia
```

```
02:00.1 0403: 10de:0fbb (rev a1)
```

```
Subsystem: 1458:3679
```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

```
Kernel modules: snd_hda_intel
```

Notice the lines “Kernel driver in use: vfio-pci”. If it shows a different driver, something went wrong (be sure to check the right graphics card!).

In case blacklisting the graphics driver didn’t work, you can add the following step:

Open or create `/etc/modprobe.d/graphics-drivers.conf`:

```
root admin:///etc/modprobe.d/graphics-drivers.conf
```

and paste the following line:

```
blacklist nouveau
```

(Change “nouveau” to the driver name you found with `lspci -nk`)

Update the initramfs by running:

```
sudo update-initramfs -u
```

Reboot your PC and check again with `lspci -nk`.

If this method of blacklisting the driver doesn’t work, use the “[module alias](#)” method I recommended in the beginning.

If this post has been helpful, click the “Like” button below. Don’t forget to share this page with your friends.

Share this:



Like this:

Loading...

[Passing Through a Nvidia RTX 2070 Super GPU](#)

January 31, 2021

In "Hardware"

[Running Windows 10 on Linux using KVM with VGA Passthrough](#)


July 20, 2017

In "Linux"

[Creating a Windows 10 kvm VM on the AMD Ryzen 9 3900X using Qemu 4.0 and VGA Passthrough](#)

March 24, 2020

In "Linux"

 Heiko Sieger / December 21, 2018 / Linux, Virtualization / blacklist, driver override, gpu passthrough, kvm, modalias, pci passthrough, vga passthrough, virtualization

You must [log in](#) to post a comment.

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

Heiko's Blog / [Privacy Policy](#) / Proudly powered by WordPress

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use. To find out more, including how to control cookies, see here: [Cookie Policy](#)