Proxmox - Source routing with OpenVZ & Linux

If, like me, you have to run lots of OpenVZ-based virtual server hosts, you will likely have encountered the fun that is reverse-path filtering, or 'rp_filter'. This is the function of the kernel that rejects 'martian' IP addresses arriving on any given interface. This is usually a good thing, until you wish to connect your OpenVZ host to two separate networks and have it route IP addresses from both subnets to & from your guests via the VENET-style interfaces.

Essentially, despite differing source addresses, only one default gateway exists to send traffic to IPs not within the connected subnets and thus, traffic on any "secondary" subnet is rejected as a martian when leaving the host's interface that is connected to its default gateway.

Some people would use bridged intefaces, although this is sadly not an option for me right now. Whilst the performance of VENET is supposedly better, we also have a large install-base of VENET guests that do not wish to be disturbed. So for now I still need a way to make this work with VENET interfaces (and also VETH if required later).

There are two methods around the return path filtering, with the first being a terrible hack that should only be used temporarily, if at all… If you echo '1' to `/proc/sys/net/ipv4/conf/all/log_martians`, you will be able to see which interface is filtering martian packets. With that information you can then simply disable the rp_filter function by echoing '0' to /proc/sys/net/ipv4/conf/INTERFACE/rp_filter and martians won't be filtered.

However, this isn't a sensible option. A better solution is to actually create a routing rule to alter the default gateway used, based on the source subnet. It took me a little bit of digging, but I eventually managed to get this working after combing a few sources (including, but not limited to, the iproute2 man file).

For reference, here's my routing table showing two networks and two /32 IPs assigned to a guest's VENET interface (note that the networks are /23's, not /24's!):

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.9.159      0.0.0.0         255.255.255.255 UH    0      0        0 venet0
10.0.125.53     0.0.0.0         255.255.255.255 UH    0      0        0 venet0
10.0.8.0        0.0.0.0         255.255.254.0   U     0      0        0 br0
10.0.124.0      0.0.0.0         255.255.254.0   U     0      0        0 br1
0.0.0.0         10.0.9.1        0.0.0.0         UG    0      0        0 br0
```

Start by opening `/etc/iproute2/rt_tables` in your favourite editor. You'll need to append a line to the bottom to create a new routing table:

```
# cat /etc/iproute2/rt_tables
#
# reserved values
#
255     local
254     main
```

```
253    default
0    unspec
#
# local
#
#1    inr.ruhep
100    vlan4
```

As you can see, I've appended a new table named 'vlan4′ (picking a sensible name *helps*, in my case this is the VLAN name for 10.0.124.0/23) and given it a priority of 100. As per my understanding, the priority should be decremented for each subsequent table defined.

Now you need to use ip to define the new rules & routing behaviour, taking advantage of the new table we've defined. First, create a rule matching traffic from your secondary subnet:

```
ip rule add from 10.0.124.0/23 iif venet0 table vlan4
```

For reference, the 'iif' attribute is not a mistake; "iif" not "if". This was also a key part of the setup, as it *only* classifies traffic originating from the VENET interfaces, no-where else.

Now add a route to define the new default gateway for our new table of classified traffic and apply it:

```
ip route add default via 10.0.125.1 dev br1 table vlan4
ip route flush cache
```

You should now find that guest traffic from either network is routed correctly without having to change any rp_filter settings. At any time you can use the following two commands to see your configuration:

```
ip rule show
ip route show table vlan4
```

Be sure to re-apply the 'ip rule' and 'ip route' statements on your next reboot; under Scientific Linux 6.0 I've used the /etc/rc.local file, but you can just as easily apply them on ifup in Debian's network configuration.

Eg for 86.x network:

```
cp /etc/iproute2/rt_tables /etc/iproute2/rt_tables.org
nano /etc/iproute2/rt_tables
## Append 100  vlan85

ip rule add from 192.168.85.0/24 iif venet0 table vlan85
ip route add default via 192.168.85.1 dev vmbr2 table vlan85
ip route flush cache
```