Debug and fix Virtuozzo / KVM broken Hypervisor error: 'PrlSDKError('SDK error: 0x80000249: Unable to connect to Virtuozzo. You may experience a connection problem or the server may be down.' on CentOS Linux howto

Author : admin

I've recently yum upgraded a CentOS Linux server runinng Virtuozzo kernel and Virtuozzo virtualization Virtual Machines to the latest available **CentOS Linux release 7.9.2009 (Core)** just to find out after the upgrade there was issues with both virtuozzo (VZ) way to list installed VZ enabled VMs reporting **Unable to connect to Virtuozzo** error like below:

```
[root@CENTOS etc]# prlctl list -a
Unable to connect to Virtuozzo. You may experience a connection problem or the server may be down. Contact your Virtuozzo administrator for assistance.
```

Even the **native QEMU KVM VM**s installed on the Hypervisor system failed to work to list and bring up the VMs producing another unexplainable error with v**irsh unable to connect to the hypervisor**

socket

```
[root@CENTOS etc]# virsh list --all
error: failed to connect to the hypervisor
error: Failed to connect socket to '/var/run/libvirt/libvirt-sock': No such file or directory
```

In **dmesg** cmd ***kernel log*** **messages** the error found looked as so:

```
[root@CENTOS etc]# dmesg|grep -i sdk
```

[5.314601] PrlSDKError('SDK error: 0x80000249: Unable to connect to Virtuozzo. You may experience a connection problem or the server may be down. Contact your Virtuozzo administrator for assistance.',)

To fix it I had to experiment a bit based on some suggestions from Google results as usual and what turned to be the cause is a now obsolete setting for disk probing that is breaking **libvirtd** ...

Disable allow_disk_format_probing in /etc/libvirt/qemu.conf

The fix to **PrlSDKError('SDK error: 0x80000249: Unable to connect to Virtuozzo comes to commenting a parameter inside**

/etc/libvirt/qemu.conf

which for historical reasons seems to be turned on by default it is like this

allow_disk_format_probing = 1

Resolution is to either change the value to 0 or completely comment the line**:**

```
[root@CENTOS etc]# grep allow_disk_format_probing /etc/libvirt/qemu.conf
# If allow_disk_format_probing is enabled, libvirt will probe disk
#allow_disk_format_probing = 1
#allow_disk_format_probing = 1
```

Debug problems with Virtuozzo services and validate virtualization setup

What really helped to debug the issue was to check the extended status info of **libvirtd.service vzevent vz.service libvirtguestd.service prl-disp systemd services**

```
[root@CENTOS etc]# systemctl -l status libvirtd.service vzevent
vz.service libvirtguestd.service prl-disp
...
```

Here I had to analyze the errors and googled a little bit about it

Once this is changed I had to of course **restart libvirtd.service and rest of virtuozzo / kvm services**

```
[root@CENTOS etc]# systemctl restart libvirtd.service ibvirtd.service vzevent
vz.service libvirtguest.service prl-disp
...
```

Another useful tool part of a standard **VZ install that I've used to make sure each of the Host OS Hypervisor components is running smoothly is *virt-host-validate*** (tool is part of l**ibvirt-client rpm package**)

```
[root@CENTOS etc]# virt-host-validate
```

```
QEMU: Checking for hardware virtualization                 : PASS
QEMU: Checking if device /dev/kvm exists                   : PASS
QEMU: Checking if device /dev/kvm is accessible            : PASS
QEMU: Checking if device /dev/vhost-net exists             : PASS
QEMU: Checking if device /dev/net/tun exists               : PASS
QEMU: Checking for cgroup 'memory' controller support      : PASS
QEMU: Checking for cgroup 'memory' controller mount-point  : PASS
QEMU: Checking for cgroup 'cpu' controller support         : PASS
QEMU: Checking for cgroup 'cpu' controller mount-point     : PASS
QEMU: Checking for cgroup 'cpuacct' controller support     : PASS
QEMU: Checking for cgroup 'cpuacct' controller mount-point : PASS
QEMU: Checking for cgroup 'cpuset' controller support      : PASS
QEMU: Checking for cgroup 'cpuset' controller mount-point  : PASS
QEMU: Checking for cgroup 'devices' controller support     : PASS
QEMU: Checking for cgroup 'devices' controller mount-point : PASS
QEMU: Checking for cgroup 'blkio' controller support       : PASS
QEMU: Checking for cgroup 'blkio' controller mount-point   : PASS
QEMU: Checking for device assignment IOMMU support         : PASS
QEMU: Checking if IOMMU is enabled by kernel               : WARN (IOMMU
appears to be disabled in kernel. Add intel_iommu=on to kernel cmdline arguments)
LXC: Checking for Linux >= 2.6.26                          : PASS
LXC: Checking for namespace ipc                            : PASS
LXC: Checking for namespace mnt                            : PASS
LXC: Checking for namespace pid                            : PASS
LXC: Checking for namespace uts                            : PASS
LXC: Checking for namespace net                            : PASS
LXC: Checking for namespace user                           : PASS
LXC: Checking for cgroup 'memory' controller support       : PASS
LXC: Checking for cgroup 'memory' controller mount-point   : PASS
LXC: Checking for cgroup 'cpu' controller support          : PASS
LXC: Checking for cgroup 'cpu' controller mount-point      : PASS
LXC: Checking for cgroup 'cpuacct' controller support      : PASS
LXC: Checking for cgroup 'cpuacct' controller mount-point  : PASS
LXC: Checking for cgroup 'cpuset' controller support       : PASS
LXC: Checking for cgroup 'cpuset' controller mount-point   : PASS
LXC: Checking for cgroup 'devices' controller support      : PASS
LXC: Checking for cgroup 'devices' controller mount-point  : PASS
LXC: Checking for cgroup 'blkio' controller support        : PASS
LXC: Checking for cgroup 'blkio' controller mount-point    : PASS
LXC: Checking if device /sys/fs/fuse/connections exists    : PASS
```

One thing to note here that **virt-host-validate** helped me to realize the **fuse (File system in userspace) module kernel support enabled on the HV was missing** so I've enabled temporary for this boot with

modprobe and permanently via a configuration like so**:**

```
 # to load it one time
[root@CENTOS etc]#  modprobe fuse
# to load fuse permnanently on next boot

 [root@CENTOS etc]#  echo fuse >> /etc/modules-load.d/fuse.conf
```

Disable selinux on CentOS HV

Another thing was **selinux** was enabled on the HV. Selinux is really annoying thing and to be honest I never used it on any server and though its idea is quite good the consequences it creates for daily sysadmin work are terrible so I usually disable it. It could be that a Hypervisor Host OS might work just normal with the selinux enabled but just in case I decided to remove it. This is how

```
 [root@CENTOS etc]#  sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      31
```

To temporarily change the SELinux mode from targeted to permissive with the following command:

```
 [root@CENTOS etc]#  setenforce 0
```

Edit **/etc/selinux/config** file and set the **SELINUX** mod to **disabled**

```
[root@CENTOS etc]# vim /etc/selinux/config
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Finally rebooted graceously the machine just in case with the good recommended way to reboot servers with **shutdown** command instead of **/sbin/reboot**

```
[root@CENTOS etc]# shutdown -r now
...
```

The advantage of shutdown is that it tries to shutdown each service by sending stop requests but usually this takes some time and even a shutdown request could take longer to proccess as each service such as a WebServer application is being waited to close all its network connections etc. |
However if you want to have a quick reboot and you don't care about any established network connections to third party IPs you can go for the brutal old fashioned **/sbin/reboot** :)