vPF_RING User's Guide

High Speed Packet Capture On Virtual Machines

Version 1.0.0
July 2011

1.Table of Contents

2. Introduction

vPF_RING is a high speed packet capture framework that turns a Virtual Machine running on a commodity PC into an efficient network measurement box.

2.1. What's New with vPF_RING User's Guide?

- Release 1.0 (July 2011)
  - Initial vPF_RING users guide.

2.2. Terminology

Throughout this document we use the following terms:

- Guest
  This term indicates the virtual machine operating system. In other words this is the virtual machine that is running the virtualized Linux environment.

- Host
  The host is the physical machine (bare hardware) on which the hypervisor runs. Virtual machines are sitting on top of the hypervisor.

2.3. Prerequisites

Below you can find the list of main vPF_RING prerequisites:

- Linux kernel 2.6.30 or better, with KVM support.
- 64 bit host Linux.
- OS guests supported by vPF_RING are limited to Linux.
- Hardware system with virtualization support enabled in the BIOS (required by KVM).

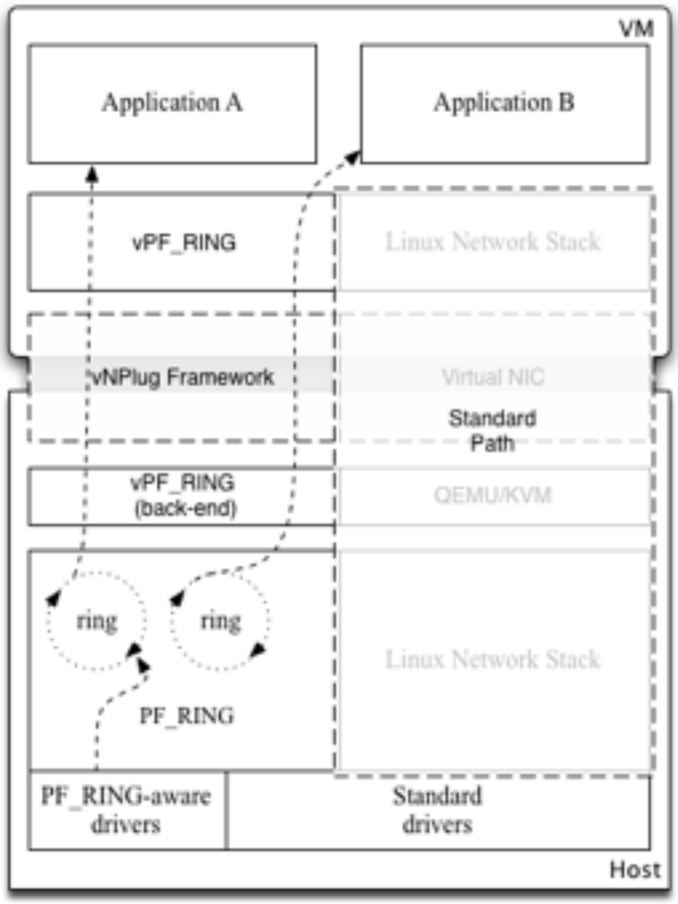
2.4. License

All vPF_RING components are distributed in source format and stored inside <PF_RING>/vPF_RING.

In order to develop applications on top of vPF_RING, it is necessary to get the vPF_RING SDK. Please refer to chapter 5.

3.Welcome to vPF_RING

vPF_RING's architecture is depicted in the figure below.



The main building blocks from the bottom are:

- Specialized PF_RING-aware drivers (optional) (host side) that allow to further enhance packet capture by efficiently copying packets from the driver to PF_RING without passing through the kernel data structures. For further information please refer to the PF_RING User's Guide.
- The standard PF_RING kernel module (host side).
- The vPF_RING backend (host side), that interacts with the standard PF_RING module on the host side, and the user-space library on the guest side by means of the vNPlug Framework.
- The vNPlug Framework (host and guest side), that provides a direct mapping of the PF_RING memory structures on the guest and a reliable communication channel. This framework comes as a QEMU patch on the host side, and a kernel module on the guest side.
- The user-space vPF_RING library that provides transparent PF_RING-support to user-space applications on the VM.

Incoming packets are copied by the kernel module on the host side into a memory ring allocated at creation time, and directly read by the user-space applications on the VM.

Applications can issue standard PF_RING API calls, described in the PF_RING User's Guide.

4. vPF_RING Installation

vPF_RING source code is distributed with PF_RING. Download vPF_RING as explained in http://www.ntop.org/products/pf_ring/vpf_ring

The <PF_RING>/vPF_RING source code layout is the following:

- README
- docs/
- guest/
- host/
- img/

4.1. Host side

This section explains how to compile the prerequisites for the host side.

4.1.1.Installation Prerequisites

The vPF_RING installation expects that PF_RING is compiled and installed on the host system.

```
host $ cd <PF_RING>/kernel
host $ make
(as root do)
host # make install
host # insmod pf_ring.ko

host $ cd <PF_RING>/userland/lib
host $ ./configure
host $ make
(as root do)
host # make install
```

Note: if you want to use a PF_RING-aware drivers with transparent_mode or other settings, please refer to the PF_RING User's Guide.

4.1.2.Patched QEMU Installation

Compile and install the patched QEMU (part of this distribution) as explained below:

```
host $ cd <PF_RING>/vPF_RING/host
host $./configure
host $ make
(as root do)
host # make install
```

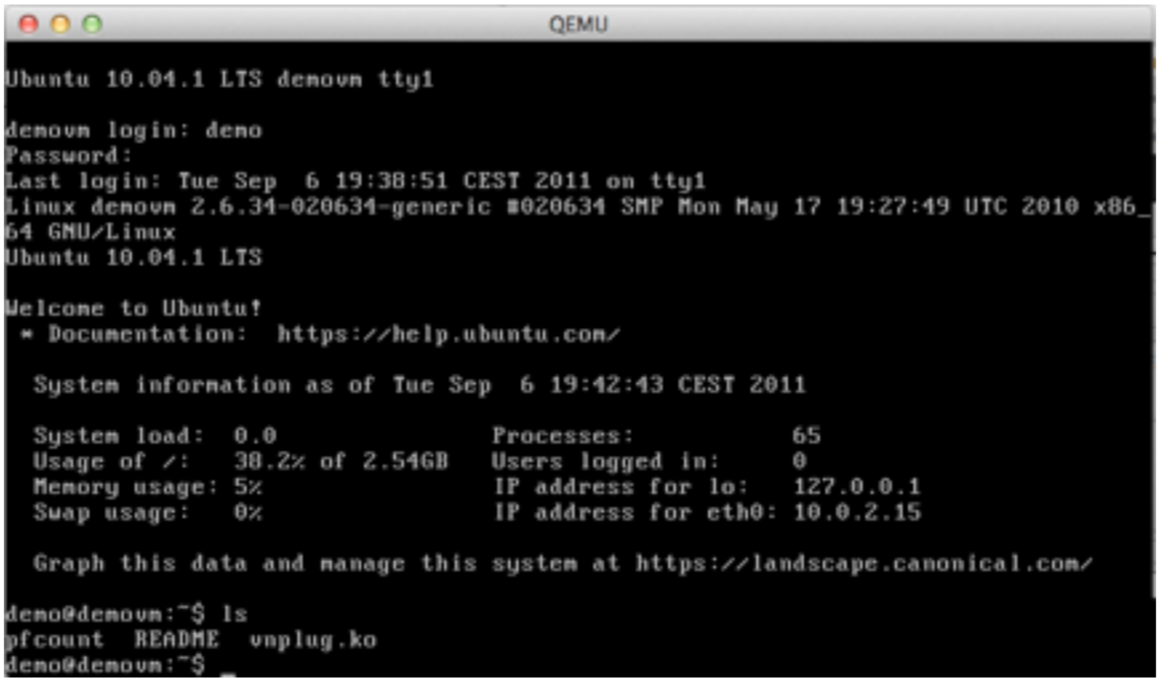
Note that you cannot use the QEMU binary that comes with your distribution, as we need to patch it in order to support vPF_RING.

4.2. Guest side

In order to run the VM with vNPlug/vPF_RING support use the "-device vnplug" parameter that instructs QEMU to enable vPF_RING support. In order to run a guest you need to supply as parameter a virtual machine disk image. For your convenience we provide you a simple demo image you can find inside <PF_RING>/vPF_RING/img. The default user of the demo image is 'demo' with password 'VPF_RING'. Once you are inside the VM, you can 'sudo su' to switch to root.

```
host # modprobe kvm_intel
host # /usr/local/kvm/bin/qemu-system-x86_64 \
        -hda ubuntu-amd64.img \
        -boot c \
        -m 512 \
        -vnc 0.0.0.0:0 \
        -device vnplug
```

After you started your VM, you can connect to it using a VNC client that will connect to the IP address of the server running QEMU. Optionally you can configure a virtual network interface to ssh to it. Please refer to KVM guide for more information, or use the utility scripts present in <PF_RING>/vPF_RING/img.



```
Ubuntu 10.04.1 LTS demovm tty1

demovm login: demo
Password:
Last login: Tue Sep  6 19:38:51 CEST 2011 on tty1
Linux demovm 2.6.34-020634-generic #020634 SMP Mon May 17 19:27:49 UTC 2010 x86_
64 GNU/Linux
Ubuntu 10.04.1 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Sep  6 19:42:43 CEST 2011

  System load:  0.0               Processes:           65
  Usage of /:   38.2% of 2.54GB   Users logged in:     0
  Memory usage: 5%                IP address for lo:   127.0.0.1
  Swap usage:   0%                IP address for eth0: 10.0.2.15

  Graph this data and manage this system at https://landscape.canonical.com/

demo@demovm:~$ ls
pfcount  README  vnplug.ko
demo@demovm:~$ _
```

Note: when using the preconfigured VM, please refer to the README file present in the VM image inside the home directory.

5. vPF_RING SDK

1.vNPlug Kernel Module Installation (Guest side)

In order for the vNPlug framework to work properly, you should load the acpiphp kernel module (hotplug support), otherwise it won't be able to dynamically map ring memory.

```
(as root do)
guest # modprobe acpiphp
```

Compile and install the vNPlug kernel module.

```
guest $ cd <PF_RING>/vPF_RING/guest/kernel
guest $ make
(as root do)
guest # make headers_install
guest # insmod vnplug.ko
```

2.vPF_RING Library Installation

Copy the vPF_RING user-space module.

```
guest $ cp libpfring_mod_virtual_XXX.a <PF_RING>/userland/lib/libs/
guest $ cp pfring_mod_virtual.h <PF_RING>/userland/lib/
```

Compile and install the PF_RING library with VPF_RING support

```
guest $ cd <PF_RING>/userland/lib
guest $ ./configure
guest $ make
(as root do)
guest # make install
```

Example: compile and run pfcount

```
guest $ cd <PF_RING>/userland/examples
guest $ make pfcount
(as root do)
guest #./pfcount -i host:eth0
```

6.Using vPF_RING

Before using any PF_RING application the pf_ring kernel module should be loaded on the host side.

```
host # insmod <PF_RING>/kernel/pf_ring.ko
```

Note: if you want to use a PF_RING-aware drivers with transparent_mode or other settings, please refer to the PF_RING User's Guide.

On the guest side both the standard hotplug module and the vnplug module should be loaded.

```
guest # modprobe acpiphp
guest # insmod vnplug.ko
```

6.1. Checking PF_RING Device Configuration

As with standard PF_RING, when a ring is activated a new entry /proc/net/pf_ring is created on the host.

```
host # cat /proc/net/pf_ring/info
Version              : 4.7.1
Ring slots           : 4096
Slot version         : 13
Capture TX           : Yes [RX+TX]
IP Defragment        : No
Socket Mode          : Standard
Transparent mode     : Yes (mode 0)
Total rings          : 0
Total plugins        : 2
```

6.2. Libpfring and Libpcap

As vPF_RING results in a standard PF_RING module which is hidden by the PF_RING API, both libpfring and libpcap can be compiled and used as described in the PF_RING User's Guide.

Note: in order to indicate to the library to use the vPF_RING module, you need to prepend 'host:' to the device name (e.g. host:ethX@Y).