

PF_RING Installation on CentOS 5.3

by: **Hargyo Tri Nugroho** (hargyo [at] lc [dot] vism [dot] org)

Keyword: PFRING, PF_RING, CentOS 5.3, WARNING: "HARD_TX_UNLOCK" [net/ring/ring.ko] undefined!

Luca Deri, the author, has done very great job on this. However, many people found difficulties in installing this PF_RING. Hopefully, this documentation could help you :).

1. Introduction

PF_RING is a high speed packet capture library that turns a commodity PC into an efficient and cheap network measurement box suitable for both packet and active traffic analysis and manipulation. Moreover, PF_RING opens totally new markets as it enables the creation of efficient application such as traffic balancers or packet filters in a matter of lines of codes.

2. Download

1. We will download the PF_RING using subversion. If your network is under http proxy then you have to configure the svn as follows: edit `~/.subversion/servers`, and under the [global] section uncomment and edit `http-proxy-host`, and `http-proxy-port` e.g:

```
[global]
# http-proxy-exceptions = *.exception.com, www.internal-site.org
http-proxy-host = proxy.csie.ncu.edu.tw
http-proxy-port = 3128
```

2. Define what directory will be used to save the PF_RING source code. In this tutorial we will use `/root/my_pfring/`

```
[root@black ~]# mkdir my_pfring
[root@black ~]# cd my_pfring
```

3. Download the PF_RING source code

```
[root@black my_pfring]# svn co https://svn.ntop.org/svn/ntop/trunk/PF_RING
```

4. After all file has been downloaded then you will get a folder named PF_RING under the directory `/root/my_pfring/`

```
[root@black PF_RING]# ls
doc kernel mkpatch.sh README userland
```

5. Cek what kernel version you have and **edit** 'mkpatch.sh' according to your the kernel version. The configuration below means that you will use 2.6.18-128.el5

```
# kernel identifiers.
VERSION=${VERSION:-2}
PATCHLEVEL=${PATCHLEVEL:-6}
SUBLEVEL=${SUBLEVEL:-18}
KERNEL_VERSION=$VERSION.$PATCHLEVEL.$SUBLEVEL
EXTRAVERSION=${EXTRAVERSION:--128.el5-$PATCH}
```

6. Run the `mkpatch.sh`

```
[root@black PF_RING]# sh mkpatch.sh
Creating patch for Linux kernel linux-2.6.18 ...
Edit this file (mkpatch.sh) for a different kernel version
.....truncated...
```

```
6. Patching file net/Kconfig ... done
diff --unified --recursive --new-file linux-2.6.18-128.el5-PF_RING > linux-2.6.18-128.el5-PF_RING.patch
Making Linux patch file. This could take some time, please wait ... done
Your patch file is now in /root/my_pfring/PF_RING/workspace/linux-2.6.18-128.el5-PF_RING.patch.gz
```

Don't be confused with these 2 lines above. By running 'mkpatch.sh', the script will create folder 'workspace/linux-2.6.18-1-686-smp-PF_RING' as the location of the patched kernel source code.

7. Go to the folder 'workspace'. After running `mkpatch.sh` you will these files under folder 'workspace' and

```
[root@black PF_RING]# cd workspace/
[root@black workspace]# ls
linux-2.6.18                linux-2.6.18.tar.gz
linux-2.6.18-128.el5-PF_RING PF_RING
linux-2.6.18-128.el5-PF_RING.patch.gz
```

8. Get into folder 'linux-2.6.18-128.el5-PF_RING' and apply these patch:

```
[root@black workspace]# cd linux-2.6.18-128.el5-PF_RING
```

```
---
diff -u linux-2.6.18-128.el5-PF_RING/include/linux/netdevice.h linux-2.6.18-128.el5-PF_RING-yoyo/include/linu
/netdevice.h
```

```
--- linux-2.6.18-128.el5-PF_RING/include/linux/netdevice.h 2006-09-20 11:42:06.000000000 +0800
+++ linux-2.6.18-128.el5-PF_RING-yoyo/include/linux/netdevice.h 2009-07-26 20:11:54.000000000 +0800
@@ -913,12 +913,15 @@
```

```
clear_bit(__LINK_STATE_RX_SCHEDULED, &dev->state);
}
```

```
-static inline void netif_tx_lock(struct net_device *dev)
+static inline void __netif_tx_lock(struct net_device *dev, int cpu)
```

```
{
spin_lock(&dev->_xmit_lock);
- dev->xmit_lock_owner = smp_processor_id();
+ dev->xmit_lock_owner = cpu;
+}
+static inline void netif_tx_lock(struct net_device *dev)
+{
+ __netif_tx_lock(dev, smp_processor_id());
+}
-
```

```
static inline void netif_tx_lock_bh(struct net_device *dev)
{
spin_lock_bh(&dev->_xmit_lock);
@@ -945,6 +948,18 @@
spin_unlock_bh(&dev->_xmit_lock);
}
```

```
+#define HARD_TX_LOCK(dev, cpu) { \
+ if ((dev->features & NETIF_F_LLTX) == 0) { \
+ __netif_tx_lock(dev, cpu); \
+ } \
+}
```

```
+#define HARD_TX_UNLOCK(dev) { \
+ if ((dev->features & NETIF_F_LLTX) == 0) { \
+ netif_tx_unlock(dev); \
+ } \
+}
```

```
static inline void netif_tx_disable(struct net_device *dev)
{
netif_tx_lock_bh(dev);
```

and also,

```
diff -u linux-2.6.18-128.el5-PF_RING/net/core/dev.c linux-2.6.18-128.el5-PF_RING-yoyo/net/core/dev.c
--- linux-2.6.18-128.el5-PF_RING/net/core/dev.c 2009-07-26 21:51:01.000000000 +0800
```

```
+++ linux-2.6.18-128.el5-PF_RING-yoyo/net/core/dev.c 2009-07-26 20:16:07.000000000 +0800
@@ -1569,6 +1569,7 @@
```

```
return 0;
}
```

```

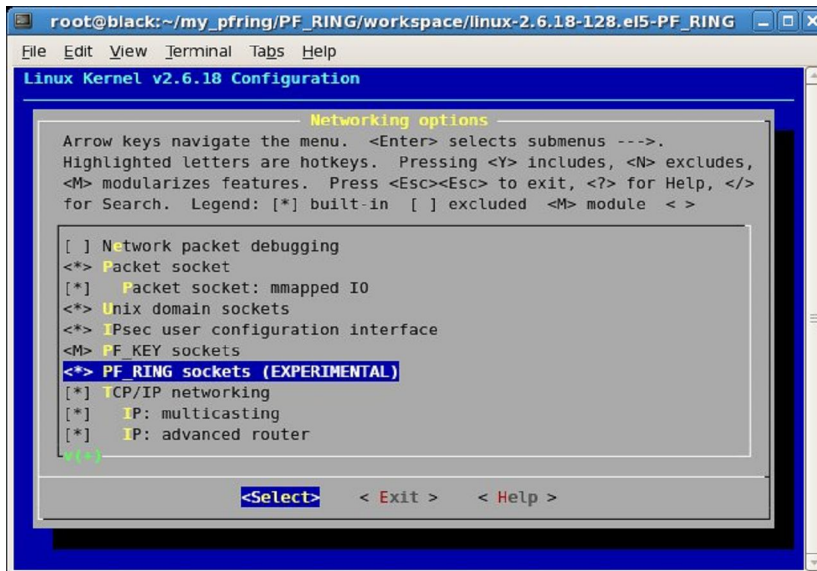
+/*
#define HARD_TX_LOCK(dev, cpu) { \
    if ((dev->features & NETIF_F_LLTX) == 0) { \
        netif_tx_lock(dev); \
@@ -1580,6 +1581,7 @@
        netif_tx_unlock(dev); \
    } \
}
+*/ I only uncomment this since I am too lazy to delete them :D

/**
 * dev_queue_xmit - transmit a buffer
--

```

8. After applying those patches, now you have to configure the kernel using 'menuconfig':
 [root@black linux-2.6.18-128.e15-PF_RING]# make menuconfig

After you get this screen then you can configure your kernel. Enable the 'PF_RING sockets' under 'Networkin Options'. Please be noticed that in this tutorial I am using standard configuration. But you are free to configure your own way.



After it is all done save your configuration to file '.config' .

```

# configuration written to .config
#

```

```

*** End of Linux kernel configuration.
*** Execute 'make' to build the kernel or try 'make help'.

```

```

[root@black linux-2.6.18-128.e15-PF_RING]#

```

10. Now, you have to edit file 'PF_RING/workspace/linux-2.6.18-128.e15-PF_RING/drivers/atm/he.c' at line 788 : change *static int __init* to *static int __devinit*

```

return 0;
}

```

```

//static int __init <-----before
static int __devinit
he_init_group(struct he_dev *he_dev, int group)
{
    int i;

```

11. After that compile the modules and install it

```
[root@black linux-2.6.18-128.el5-PF_RING]# make
[root@black linux-2.6.18-128.el5-PF_RING]# make modules
if you get some warning..just ignore it, unless you know the better solution :D
[root@black linux-2.6.18-128.el5-PF_RING]# make modules_install
[root@black linux-2.6.18-128.el5-PF_RING]# make install
```

12. Make sure that this new kernel already added in the boot menu otherwise you have to add it. :)

```
title CentOS (2.6.18) PF_RING
root (hd0,6)
kernel /boot/vmlinuz-2.6.18 ro root=LABEL=/1 rhgb quiet
initrd /boot/initrd-2.6.18.img
```

13. Reboot and get into your Linux with new (PF_RING) kernel.

3. Libpfring and Libpcap Installation

Both libpfring and libpcap are distributed in source format. They can be compiled as follows:

```
[root@black PF_RING]# ls
doc kernel mkpatch.sh README userland workspace
[root@black PF_RING]# cd userland
[root@black userland]# ls
c++ examples lib libpcap-1.0.0-ring Makefile perl
[root@black userland]#
[root@black userland]# cd lib
[root@black userland]# ls
[root@black userland]# make
[root@black userland]# make install
[root@black userland]# cd ../libpcap-1.0.0-ring/
[root@black userland]# ./configure
[root@black userland]# make
```

You can try some examples provided by the author under folder '**examples**'. Compile and enjoy it :)

4. PF_RING Device Configuration

Once PF_RING is activated, a new entry /proc/net/pf_ring is created. For example:

```
[root@black examples]# ls /proc/net/pf_ring/
info plugins_info
[root@black examples]# cat /proc/net/pf_ring/info
Version      : 3.9.5
Ring slots   : 4096
Slot version  : 9
Capture TX   : Yes [RX+TX]
IP Defragment : No
Transparent mode : Yes
Total rings   : 0
Total plugins : 0
[root@black examples]#
```

PF_RING allows users to install plugins for handling custom traffic. Those plugins are also registered in the pf_ring /proc tree and can be listed by typing the plugins_info file.