

## IPS FLEXIBLE RESPONSE

*Pierpaolo Palazzoli, Brescia, Italy*

*Fabio Mostarda, Brescia, Italy*

*Claudia Ghelfi, Brescia, Italy*

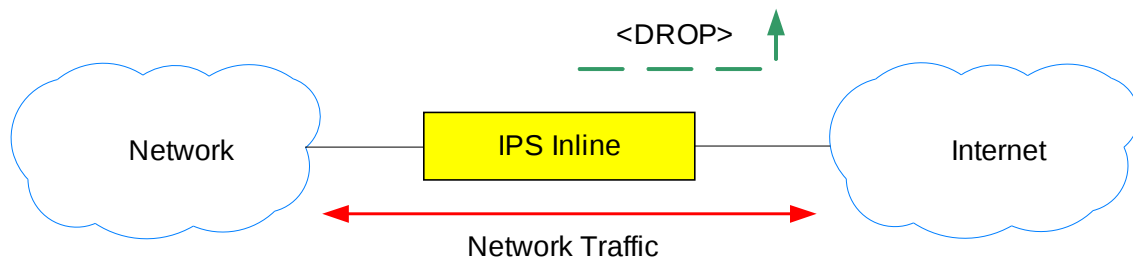
The IPS acronym references an Intrusion Prevention System, i.e. a network security device. It monitors the network traffic flow in order to feature real-time malicious traffic blocking.

An IPS can be implemented using a server wherein Snort software works as an IPS, while Iptables handles the more specific firewall tasks and, leveraging specific packages, creates packet queues to be analyzed by the system.

### SNORT

Snort is an intrusion detection and prevention system (IDPS), and can operate in two different ways, both of them totally invisible to the network flow. One of those setups, Flexible Response, is an unique feature of Snort with no equivalent on the market.

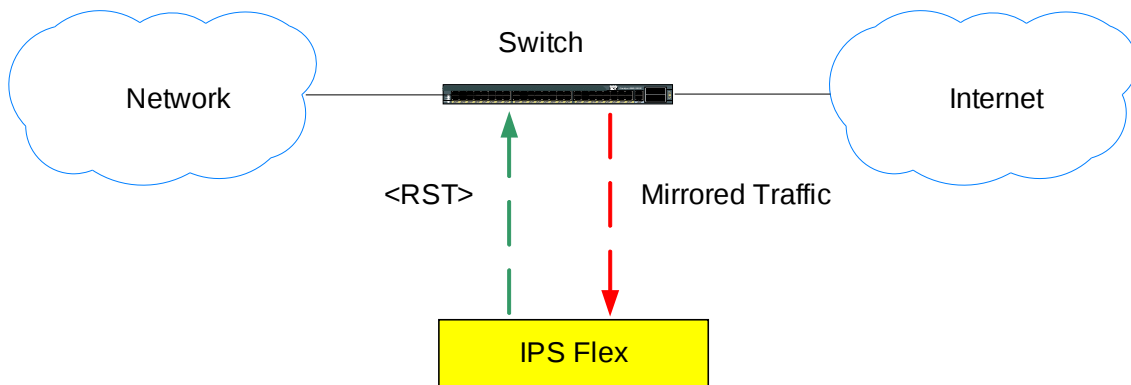
- o *Inline.* In this mode Snort works like an Ethernet bridge, that is, in order to monitor a network segment, it has to be inserted transparently with two bridged NICs.



With this setup, any packet can flow through the bridge from a network card to the other, unless it matches the drop rules; in that case, the switch opens and blocks the packet.

- o *Flexible Response.* With this second setup, all the inbound traffic is replicated by a network device (i.e. a switch) to a “mirror port”. The IPS is then linked to that port, analyzing the traffic flow and sending RST packets whenever it detects a match in the drop policies.

The “*flexible response*” expression probably references a defence strategy used in the sixties by the USA president John Kennedy. With this strategic approach, it was intended to respond to enemy threats with a proportionate retaliation, thus phasing out the old concept of massive retaliation: this had proven outdated by the Soviet weapon race.



The best performance output is achieved in inline mode, because it is able to block even attacks made up by a small number of packets.

On the contrary, the flexible response mode takes more time to identify an intrusion attempt and thus to reset the related port.

Anyway, whenever latency induced by an inline IPS is too high, it's more convenient to use the flexible response setup; Snort flexresp is a passive device, therefore its efficiency is directly related to CPU power, system setup, free memory and I/O latencies.

Snort features two Flexible Response operational modes:

- FlexRespo: doesn't really block connections, but sends error packets to the malicious attackers instead, letting them think about network unreacheability or service port outages. Every user is able to enforce its own ruleset regarding those network attacks responses.
- FlexRespo2: a brand new feature, which updates libdnet with libevent. Flexrespo2, in addition to the standard Flexrespo features, prevents response loops which may overload the CPU and disrupt the service. Additionally, it delivers multiple responses in order to enforce malicious connection terminations. It is highly suggested, whenever possible, to implement this second version.

## APPLICAZIONE

The introduction of a Flexible Response IPS is related to the adoption of network switches with mirror ports, i.e. a special port where all network traffic is mirrored to; this technology has many commercial names: e.g., SPAN (Switched Port ANalyzer) for Cisco Networks.

Mirror port traffic is replicated towards the IPS in order to be analyzed. Whenever malicious data packets are detected, Snort can work in two different ways:

- RST, thus deleting any dangerous packet;
- ICMP, communicating the information about the attack to the sender, and making the host/port unreachable.

An IPS can receive and analyze traffic coming from different mirror ports, using a single output port to forward RST or ICMPs.

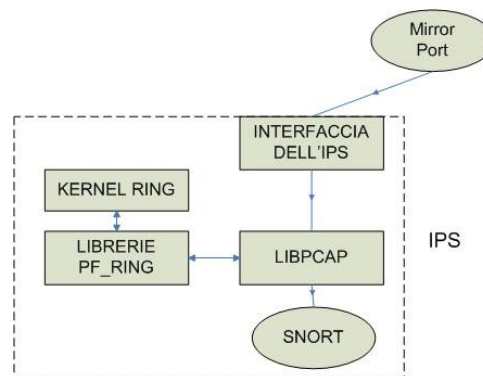
To implement a Snort IPS in Flexible Response mode, the following steps are suggested:

### 1. Installation of the following base packages:

*apt-get install vim gcc make build-essential libtool automake autoconf flex bison libpcre3-dev psmisc ethtool apache2 ntop ntpdate subversion apache2 kernel-package libncurses5-dev fakeroot wget bzip2 flex bison zlibc zlib1g-dev*

### 2. PF\_RING setup and Libpcap install:

PF\_RING is a network socket that speeds up packet capture speed. Therefore, if it is used in an IPS setup, it can increase the overall performance.



Download PFRING

```
cd /usr/src
```

```
svn co https://svn.ntop.org/svn/ntop/trunk/PF_RING/ last version of pf_ring
```

Check Kernel version:

```
uname -a
```

Modify the downloaded script in order to generate the desired kernel patch; in our example, we'll use the proper version for our kernel:

```
cd /usr/src/PF_RING/
```



```
vi mkpatch.sh
edit the right kernel
VERSION=${VERSION:-2}
PATCHLEVEL=${PATCHLEVEL:-6}
SUBLEVEL=${SUBLEVEL:-26}
Run the script to patch the kernel
sh ./mkpatch.sh
cd workspace/
cd linux-2.6.26-1-686-smp-PF_RING
cp /boot/config-2.6.26-1-686-bigmem .config
```

Enable pfring, in order to include it in the new kernel  
*make menuconfig*

select *Networking -> Networking Options* and make sure PF\_RING socket is enabled.  
Save the changes and exit the menu.

Now it's possible to generate and install the package with the new kernel:

```
make-kpkg clean
fakeroot make-kpkg -initrd -revision=pfring.1.0 linux-image
cd ../
dpkg -i linux-image-2.6.26_pfring.1.0_i386.deb
```

In order to use this new kernel at every boot, you have to properly edit this file:  
*vi /boot/grub/menu.lst*

*Create the file :*

```
vi /etc/modprobe.d/options
```

adding the line : `options ring num_slots=32740 transparent_mode=0`

65535 is the buffer ring memory. (just choose it according to system resources, if it's higher you'll have to change in grub the proper value; i.e. `vmalloc=256M` for 256M)

Reload the server and check if the correct kernel version is running:  
*reboot*

Copy the ring source file in the linux directory:

```
cp /usr/src/PF_RING/workspace/linux-2.6.26-1-686-smp-
PF_RING/include/linux/ring.h /usr/include/linux/
cd /usr/src/PF_RING/userland
make
```



**Install the Libpfring libraries:**

```
cd lib
gcc -shared -Wl,-soname -Wl,libpfring.so.0.9.7 -o libpfring.so.0.9.7 *.o -lc
cp libpfring.a libpfring.so.0.9.7 /usr/local/lib
cp pfring.h /usr/local/include
ln -s /usr/local/lib/libpfring.so.0.9.7 /usr/local/lib/libpfring.so
ldconfig
```

```
ldconfig -v |grep pfring
```

This last command checks if the installed libraries are correct; the desired input is therefore:

```
libpfring.so.0.9.7 -> libpfring.so.0.9.7
```

**Last, install Libpcap-ring libraries:**

```
cd /usr/src/PF_RING/userland
wget http://www.tcpdump.org/release/libpcap-0.9.7.tar.gz
tar -zxvf libpcap-0.9.7.tar.gz
cd libpcap-0.9.7
mv pcap-int.h pcap-int.h.orig
mv pcap-linux.c pcap-linux.c.orig
cp ../libpcap-0.9.7-ring/pcap* .
./configure CPPFLAGS="-I/usr/local/include" LDFLAGS="-L/usr/local/lib"
CFLAGS="-D_LARGEFILE_SOURCE -D_LARGEFILE64_SOURCE
-D_FILE_OFFSET_BITS=64"
make && gcc -shared -Wl,-soname -Wl,libpcap.so.`cat VERSION` -o
libpcap.so.`cat VERSION` *.o -lc
make install && cp libpcap.so.0.9.7 /usr/local/lib
ldconfig -v |grep pcap
```

This last command checks the installed libraries version; the desired input is therefore:

```
libpcap.so.0.9.7 -> libpcap.so.0.9.7
```

**3. Install Libdnet libraries:**

```
wget http://prdownloads.sourceforge.net/libdnet/libdnet-1.11.tar.gz?download
tar zxvf libdnet-1.11.tar.gz
cd libdnet-1.11
./configure
make
make install
ldconfig (rearranges libraries)
```

**4. Install Snort:**

Run:

```
wget http://www.snort.org/dl/snort-2.8.3.2.tar.gz
```

```
tar xzvf snort-2.8.3.1.tar.gz
cd snort-2.8.3.1
./configure --enable-flexresp2 --enable-memory-cleanup --enable-linux-smp-
stats --enable-pthread
vi src/Makefile
Modify the Makefile in modo order to have:
LDLFLAGS = -L/usr/lib -lpcrc -L/usr/local/lib -ldnet -lpfring -lpcap
CPPFLAGS = -DENABLE_RESPONSE2 -I/usr/local/include -fno-strict-
aliasing
If everything went fine, just run:
make
make install
cd ..
mkdir /var/log/snort
```

#### 5. Define Snort rules:

The rules according to which Snort decide sto reset network connections or not, are found in /etc/snort/rules/, arranged in files

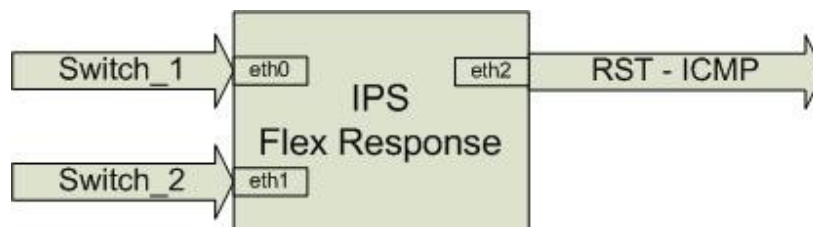
At the end of every rule, it's compulsory to have: *resp: reset\_both,icmp\_all*; this references Snort Flexible Response mode.

A rule example is:

```
alert tcp $EXTERNAL_NET 6112 -> $HOME_NET any (msg:"ET GAMES
Battle.net 'emote' message"; flow:established,from_server; content:"|FF 0F|";
depth:2; content:"|17 00 00 00|"; offset:4; depth:4; classtype: policy-violation;
sid:2002152; rev:2; resp: reset_both,icmp_all;)
```

#### 6. Run Snort:

Run a Snort instance, with special rules, analyzing traffic from every mirror port. In this example, there are two network flows coming from two switches (Switch\_1 and Switch\_2).



Snort has to be run twice, one for each switch:

```
snort -A fast -i eth0 -b -d -D -c /etc/snort/snort_1.conf
snort -A fast -i eth1 -b -d -D -c /etc/snort/snort_2.conf
```