

## IPS 3M Multicoda Multilivello Multiservizio

*Pierpaolo Palazzoli, Brescia, Italy*

*Fabio Mostarda, Brescia, Italy*

*Claudia Ghelfi, Brescia, Italy*

L'acronimo IPS fa riferimento ad un Intrusion Prevention System, ovvero ad un dispositivo per la sicurezza di rete. Tale sistema monitora il flusso di traffico in transito sulla rete per bloccare, in tempo reale, eventuali attacchi di origine dolosa o involontaria.

La tecnologia IPS nasce come estensione dell'Intrusion Detection System (IDS) e ne è tuttora strettamente legata.

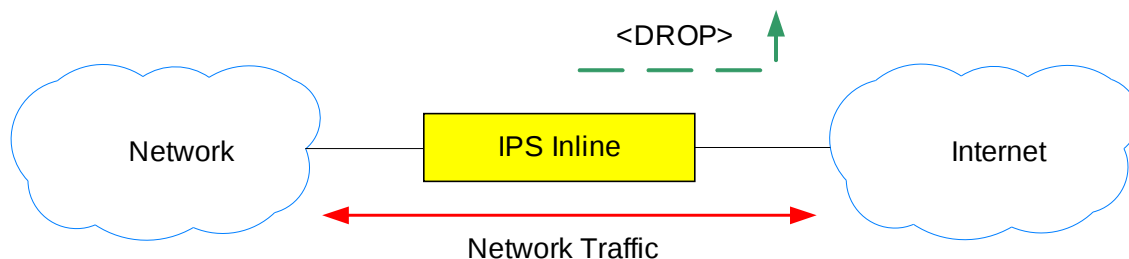
Mentre il sistema di intrusion detection si limita ad ascoltare il flusso in transito, l'IPS ha il vantaggio di poter bloccare in tempo reale il traffico ritenuto pericoloso, secondo determinate regole implementabili dall'amministratore di sistema.

L'IPS può essere implementato mediante una macchina sulla quale il software Snort svolge le funzionalità di IPS, mentre Iptables si occupa di adempiere ai compiti specifici di un firewall e di creare, appoggiandosi ad appositi moduli, le code dei pacchetti da analizzare.

### SNORT

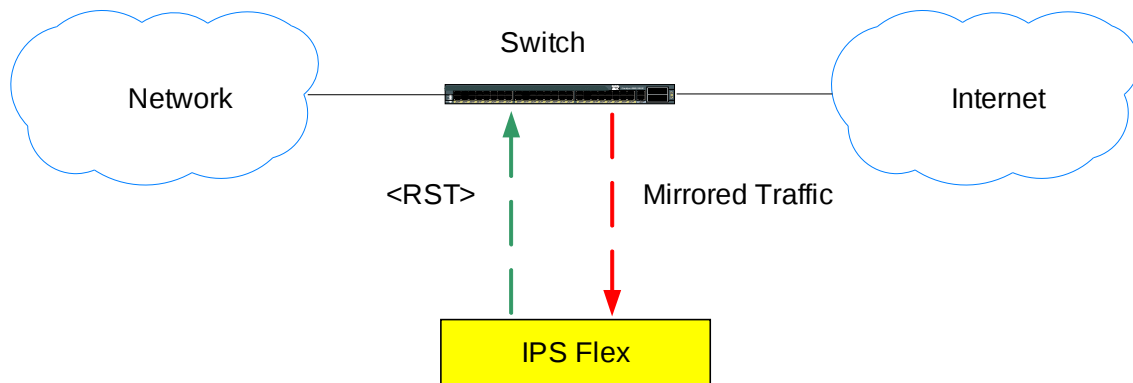
Snort è per definizione un sistema di rilevamento e prevenzione delle intrusioni (IDPS) e può operare in due diverse modalità.

- o *Inline*. In questa modalità Snort funziona come un bridge Ethernet, ovvero, per poter monitorare il traffico in un segmento di rete, va inserito in maniera trasparente tramite due schede in bridge.



Così facendo il pacchetto attraversa indisturbato il bridge da una scheda all'altra, a meno che non rispetti le regole di drop. In tale caso lo switch si apre e blocca il passaggio del pacchetto.

- o *Flex Response*. In questa seconda modalità tutto il traffico in ingresso viene replicato dallo switch verso la “mirror port”. Su tale porta viene inserito l’ips che analizza il flusso dei pacchetti e, nel caso in cui esso rispetti le regole di drop, resettare la porta corrispondente.



In entrambe le modalità l’IPS risulta trasparente e quindi invisibile sulla rete.

## IPTABLES

Delle due modalità di funzionamento di Snort sopra elencate quella con le migliori prestazioni è quella inline perché permette di bloccare anche attacchi formati da un numero limitato di piccoli pacchetti. Al contrario la modalità flex response impiega più tempo ad identificare un’intrusione e quindi a resettare la corrispondente porta.

L’utilizzo di un IPS inline non risolve tutte le problematiche di sicurezza, ma contribuisce alla strutturazione centralizzata di un sistema di sicurezza dinamico ed efficiente.

Snort inline funziona analizzando le code dei pacchetti in transito, per questo motivo è indispensabile utilizzare, in aggiunta a Snort stesso, un sistema di creazione delle code.

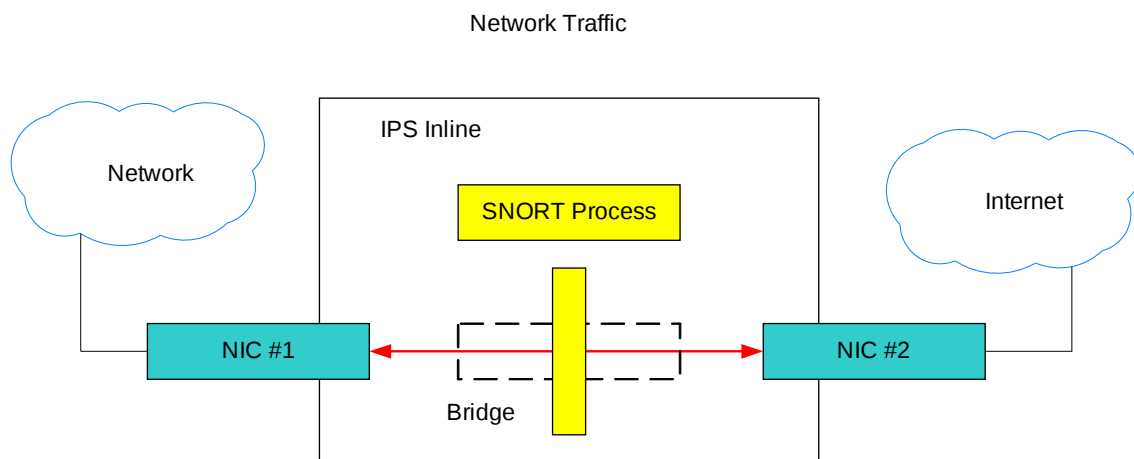
Se lo Snort utilizzato è “Snort in modalità inline” allora è possibile creare un’unica coda, mentre se si impiega “Snort\_inline” le code possibili sono innumerevoli.

Iptables, in aggiunta al suo comportamento da firewall di Linux, svolge la funzione di creazione delle code avvalendosi di moduli aggiuntivi quali ip\_queue o nfnetlink\_queue.

La differenza tra i due moduli è quella che distingue l’IPS mono-coda dall’IPS multi-coda.

## IPS MONOCODA

Nel caso in cui il modulo utilizzato per la creazione delle code sia `ip_queue`, può essere creata un'unica coda sulla quale sono forzati a scorrere tutti i pacchetti in ingresso. Questa coda è analizzata quindi da un unico Snort secondo un'unica classe di regole. Non è possibile differenziare i flussi in ingresso e applicare loro policy di blocco personalizzate.



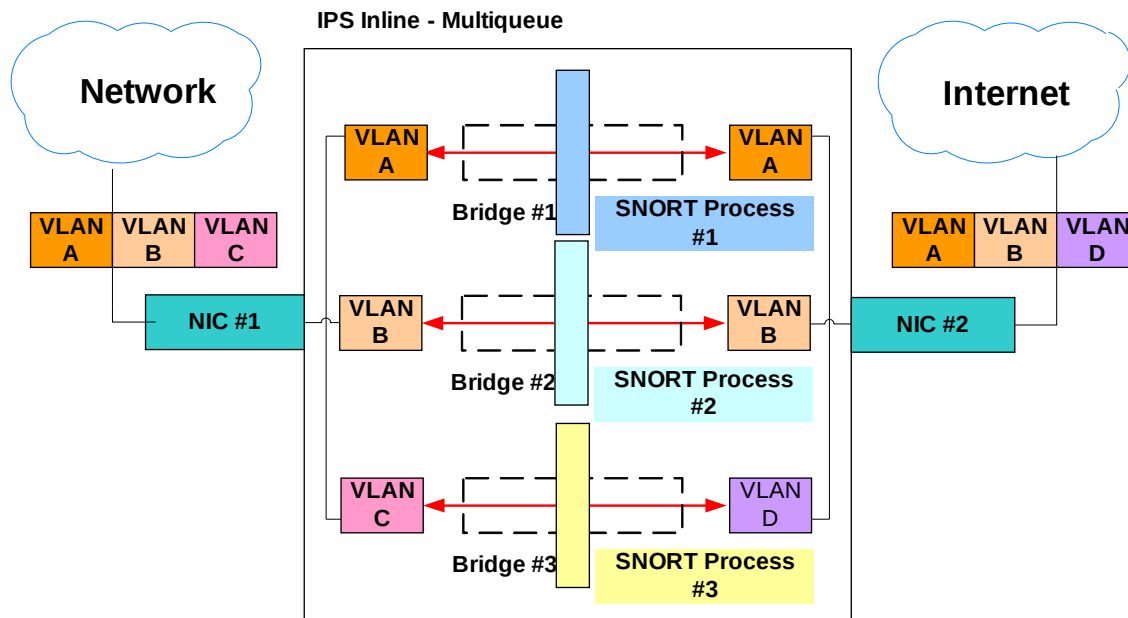
## IPS MULTICODA

Attraverso l'ausilio del modulo `nfnetlink_queue`, invece, possono essere create, in via teorica, fino a 65535 code indipendenti. Grazie a quest'opportunità è possibile differenziare i flussi in ingresso, ad esempio è possibile separare i pacchetti in arrivo da diverse VLAN, creando bridge virtuali dedicati. Sfruttando il TAG all'interno della trama Ethernet, un manageable switch è in grado di riconoscere la VLAN di provenienza del pacchetto e quindi di instradarlo sulla coda corrispondente.

In questo modo l'analisi dei flussi può essere svolta in modo indipendente, ovvero è possibile impiegare versioni diverse di Snort così come si possono implementare regole di analisi del traffico personalizzate per ogni coda, secondo le esigenze delle VLAN.

Secondo lo standard IEEE 802.1q le VLAN distinguibili sono 1024/2048, perciò i bridge possibili sono 512/1024.

All'interno di ogni coda si possono poi distinguere delle sotto-code differenziate in base al servizio e contraddistinte dal numero della porta utilizzata. Grazie a questa ulteriore suddivisione le regole di Snort possono essere applicate al singolo flusso interessato migliorando così l'efficienza del sistema.



## APPLICAZIONI

La creazione di un IPS segue i seguenti passi:

### 1. Adozione del sistema operativo e delle librerie

Sono richieste le librerie libnfnetlink-0.0.39 e libnetfilter\_queue-0.0.16. Se si sceglie di installare come sistema operativo Debian-4.0rc5 oppure Fedora-10 queste librerie possono essere agevolmente aggiungere in un secondo momento.

L'installazione dei sistemi operativi può essere tranquillamente effettuata nelle loro versioni minimali. In questo caso si suppone di star utilizzando Debian-4.0rc5.

Terminata l'installazione si modifica il file sources.list, nel modo seguente, così da poter scaricare le librerie corrette in seguito:

```
vi /etc/apt/sources.list
```

Si disabilitano tutte le istruzioni facendole precedere da # e si aggiunge:

```
deb http://ftp.it.debian.org/debian/ unstable main contrib non-free
```

si installa ssh per la connessione da remoto:

```
apt-get install ssh
```

infine si rimuovono i pacchetti non necessari, ovvero tutti tranne ssh:

```
netstat -taupen
```

(visualizza l'elenco delle applicazioni che fanno girare un demone su di una specifica porta)

```
apt-get remove portmap exim4 exim4-config openbsd-inetd
```

(rimuove i pacchetti elencati, che non sono necessari)

A questo punto si aggiornano i pacchetti installati e le loro dipendenze:

```
apt-get update
```

```
apt-get upgrade
```

ed ora è possibile installare le librerie funzionali al multicoda tramite i comandi:

```
apt-get install libnfnetlink0
apt-get install libnfnetlink-dev
apt-get install libnetfilter-queue1
apt-get install libnetfilter-queue-dev
```

Per controllare la versione delle librerie che è stata installata si può sfruttare il comando `dpkg -l [libreria]`.

## 2. Installazione di un nuovo kernel

Per operare correttamente con le librerie libnetfilter è necessario che la versione kernel sia superiore alla 2.6.25. In questo caso viene installata la seguente:

```
apt-get install linux-image-2.6-686
apt-get install linux-image-2.6-686-bigmem
```

(per RAM superiore a 3GB)

```
reboot
```

(la macchina viene riavviata)

```
uname -a
```

(attraverso questo comando è possibile controllare che il kernel nuovo sia effettivamente attivo)

## 3. Installazione dei pacchetti aggiuntivi

In aggiunta a queste librerie sono necessari i seguenti pacchetti, per il corretto funzionamento del sistema:

```
apt-get install mlocate
```

(per l'utilizzo di funzioni di ricerca quali updatedb, locate, ...)

```
apt-get install ntpdate
```

(protocollo per la sincronizzazione dell'orologio del pc)

```
apt-get install vim
```

(per rendere più agevole l'uso di vi)

```
apt-get install psmisc
```

(pacchetto contenente killall)

```
apt-get install iptraf
```

(generatore statistico di traffico per il monitoraggio di reti IP)

```
apt-get install bridge-utils
```

(pacchetto contenente utilities per la configurazione di bridge ethernet con Linux)

```
apt-get install gcc
```

(compilatore GNU)

```
apt-get install libpcap0.8
```

(librerie necessarie per catturare i pacchetti e monitorare il traffico di rete)

```
apt-get install libpcap0.8-dev
```

(sorgenti relative alle librerie precedenti che è necessario includere)

```
apt-get install libpcr3
```

(le librerie PCRE (Perl Compatible Regular Expression) contengono una serie di funzioni che implementano i comandi usando la sintassi e la semantica tipiche di Perl)

```
apt-get install libpcre3-dev
```

(sorgenti relative alle librerie pcre)

```
apt-get install libmysqlclient15-dev
```

(librerie necessarie per accedere al database mysql e scrivervi)

```
apt-get install iptables-dev
```

(sorgenti di iptables necessarie ad includere le funzionalità di accodamento dei pacchetti)

```
apt-get install automake autoconf make
```

(pacchetti necessari per la compilazione)

```
apt-get install libtool
```

(librerie necessarie all'installazione di Snort)

```
apt-get install libclamav-dev clamav-freshclam
```

(servono per permettere a Snort di andare a leggere nelle signatures del Clamav (antivirus OpenSource))

```
apt-get install libnet1-dev
```

(queste sorgenti permettono al programmatore di costruire e inserire pacchetti di rete)

```
apt-get install subversion
```

(sistema di controllo versione, utilizza un'architettura client-server: un server immagazzina la versione corrente di un progetto e la sua storia, ed il client si connette al server per verificare l'ultima versione disponibile del software ed utilizzare quest'ultima)

```
apt-get install ntop
```

(strumento di monitoraggio del traffico)

```
apt-get install apache2
```

(piattaforma server Web modulare)

```
apt-get install rrdtool
```

(sistema di memorizzazione e schematizzazione grafica dei dati)

```
apt-get install librrdp-perl librrds-perl
```

(contengono delle interfacce perl a RRDs)

#### 4. Installazione di Snort\_inline

```
svn co https://snort-inline.svn.sourceforge.net/svnroot/snort-inline/trunk
```

(scarica le sorgenti del nuovo snort\_inline )

```
cd /trunk
```

```
sh autojunk.sh
```

(prepara il pacchetto alla compilazione)

```
./configure --enable-pthread --enable-memory-cleanup --enable-stream4udp --enable-inline-init-failopen --enable-nfnlink --enable-clamav --enable-linux-smp-stats --with-mysql
```

(controlla se ci sono tutti i prerequisiti per l'installazione)

*make*  
*make install*  
(installa snort\_inline)

## 5. Ottenimento delle regole di Snort

Nel caso in cui non si possieda un backup con le regole, esse possono essere prelevate dal sito, nel seguente modo:

- o Si sceglie secondo quale modalità ottenere le regole:
  - **Subscribers:** si ricevono gli aggiornamenti delle regole, nel momento in cui sono disponibili. Per essere utenti di questo tipo si paga una quota annua che varia in base al numero di sensori:  
[http://www.snort.org/vrt/why\\_subscribe.html](http://www.snort.org/vrt/why_subscribe.html).
  - **Registered:** rientrano in questa categoria gli utenti che si registrano nel portale di snort.org. Essi ottengono le regole 30 giorni dopo la loro emissione, ma non pagano la quota annua.
  - **Unregistered:** sono quegli utenti che non vogliono registrarsi nel portale snort.org e che quindi hanno a disposizione le regole solo nel momento in cui esce la nuova release del programma.
- o Nel caso in cui si decida di divenire utenti subscribers, la sottoscrizione darà accesso ad un codice (oinkcode) che servirà come chiave per l'utilizzo di oinkmaster.
- o A questo punto si installa Oinkmaster, un programma che mantiene aggiornato l'archivio di regole in modo automatico aggiungendo, per esempio, al crontab un comando del tipo:  

```
30 2 * * * oinkmaster.pl -o /etc/snort/rules/ -t oinkmaster
```

Ogni modifica viene comunicata all'utente. Oinkmaster si può scaricare dall'indirizzo <http://oinkmaster.sourceforge.net/download.shtml>  
A queste regole vanno sommate quelle presenti in: <http://www.emergingthreats.net/>

## 6. Configurazione di rc.local

Il file rc.local si trova generalmente fra gli script di avvio (in /etc/) e viene eseguito alla fine della procedura di startup. Contiene i comandi che si vogliono eseguire dopo che tutti i servizi sono partiti.

Prima di compilare questo file, è indispensabile avere chiara l'architettura che si vuole implementare, in particolare il numero di bridge e quindi code da creare, sia fisiche, sia logiche.

Di seguito vengono proposti quattro esempi di base che operano, rispettivamente, sui livelli 1, 2, 3 e 4 dell'architettura OSI. È poi possibile, sulla base di questi esempi, creare situazioni più complesse che coinvolgano contemporaneamente più livelli, fornendo un servizio multilivello oltre che multicoda e quindi multiservizio.

physical  
Media, Signal  
and Binary Transmission

### Primo caso: due code a livello 1 (fisico)

```
/sbin/ifconfig eth4 0.0.0.0 promisc up
/sbin/ifconfig eth5 0.0.0.0 promisc up
/sbin/ifconfig eth0 0.0.0.0 promisc up
/sbin/ifconfig eth1 0.0.0.0 promisc up
brctl addbr br0
brctl addbr br1
```

(vengono definiti i due bridge br0 e br1, in questo caso fisici)

```
brctl addif br0 eth4
brctl addif br0 eth5
```

(si instaura il primo bridge tra le interfacce eth4 e eth5)

```
brctl addif br1 eth0
brctl addif br1 eth1
```

(si instaura il secondo bridge tra le interfacce eth0 e eth1)

```
ifconfig br0 up promisc
ifconfig br1 up promisc
```

Con questi comandi si accoppiano a livello due le schede di rete creando i due bridge cosicchè l'ips risulti trasparente al traffico.

Per la realizzazione dei bridge è indispensabile il pacchetto bridge-utils, precedentemente installato.

```
modprobe nfnetlink_queue
modprobe nf_conntrack
```

Sono i moduli che devono essere eseguiti per la creazione della multicoda.

```
echo "VALUE" > /proc/sys/net/nf_conntrack_max
```

E' un comando necessario alla creazione della multicoda, VALUE va sostituito dal n°sessioni che il kernel deve essere in grado di sorreggere tramite nf\_conntrack (es 262140).

```
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
echo "3" > /proc/sys/net/ipv4/tcp_fin_timeout
```

Comandi atti a sorreggere le sessioni a livello kernel.

```
sysctl -w net.core.rmem_default='8388608'
sysctl -w net.core.wmem_default='8388608'
sysctl -w net.ipv4.tcp_wmem='1048576 4194304 16777216'
sysctl -w net.ipv4.tcp_rmem='1048576 4194304 16777216'
sysctl -w net.ipv4.route.flush=1
```

Settano i parametri ottimizzati per rete ad 1Gb/s.

```
iptables -A FORWARD -i br0 -j NFQUEUE --queue-num 42
iptables -A FORWARD -i br1 -j NFQUEUE --queue-num 43
```

Generano le code 42 e 43 associate rispettivamente ai bridge fisici br0 e br1.

```
snort_inline -A fast -b -d -D -Q42 -c /etc/snort/snort.conf -l
var/log/snort/42/ --nolock-pidfile
snort_inline -A fast -b -d -D -Q43 -c /etc/snort/snort.conf -l
/var/log/snort/43/ --nolock-pidfile
```

Snort viene avviato su entrambe le code.

**data link**  
Physical Addressing (MAC & LLC)

**physical**  
Media, Signal  
and Binary Transmission

### **Secondo caso: due code a livello 2**

```
/sbin/ifconfig eth2 0.0.0.0 promisc up
/sbin/ifconfig eth3 0.0.0.0 promisc up
brctl addbr br0
brctl addif br0 eth2
brctl addif br0 eth3
ifconfig br0 up promisc
```

Si instaura il bridge fisico tra le interfacce eth2 ed eth3.

```
vconfig add eth2 5
vconfig add eth2 7
vconfig set_flag eth2.5 0
vconfig set_flag eth2.7 0
/sbin/ifconfig eth2.5 0.0.0.0 promisc up
/sbin/ifconfig eth2.7 0.0.0.0 promisc up
```

Si creano le VLAN 5 e 7 sull'interfaccia eth2 e le corrispondenti sotto-interfacce virtuali eth2.5 e eth2.7.

```
vconfig add eth3 6
vconfig add eth3 8
vconfig set_flag eth3.6 0
vconfig set_flag eth3.8 0
/sbin/ifconfig eth3.6 0.0.0.0 promisc up
/sbin/ifconfig eth3.8 0.0.0.0 promisc up
```

Si creano le VLAN 6 e 8 sull'interfaccia eth3 e le corrispondenti sotto-interfacce virtuali eth3.6 e eth3.8.

```
brctl addbr br1
brctl addif br1 eth2.5
brctl addif br1 eth3.6
ifconfig br1 up promisc
brctl addbr br2
brctl addif br2 eth2.7
brctl addif br2 eth3.8
ifconfig br2 up promisc
```

Si instaurano i bridge logici br1 (tra le sotto-interfacce eth2.5 e eth3.6) e br2 (tra le sotto-interfacce eth2.7 e eth3.8).

```
modprobe nfnetlink_queue
modprobe nf_conntrack
```

Sono i moduli che devono essere eseguiti per la creazione della multicoda.

```
echo "VALUE" > /proc/sys/net/nf_conntrack_max
```

E' un comando necessario alla creazione della multicoda, VALUE va sostituito dal n°sessioni che il kernel deve essere in grado di sorreggere tramite nf\_conntrack (es 262140).

```
echo "3" > /proc/sys/net/ipv4/tcp_fin_timeout
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
```

Comandi atti a sorreggere le sessioni a livello kernel.

```
sysctl -w net.core.rmem_default='8388608'
sysctl -w net.core.wmem_default='8388608'
sysctl -w net.ipv4.tcp_wmem='1048576 4194304 16777216'
sysctl -w net.ipv4.tcp_rmem='1048576 4194304 16777216'
sysctl -w net.ipv4.route.flush=1
```

Settano i parametri ottimizzati per rete ad 1Gb/s.

```
iptables -A FORWARD -i br1 -j NFQUEUE --queue-num 42
iptables -A FORWARD -i br2 -j NFQUEUE --queue-num 43
```

Generano le code numero 42 e 43 rispettivamente sui bridge logici br1 e br2.

```
snort_inline -A fast -b -d -D -Q42 -c /etc/snort/snort.conf -
l /var/log/snort/42 --nolock-pidfile
snort_inline -A fast -b -d -D -Q43 -c /etc/snort/snort.conf -
l /var/log/snort/43 --nolock-pidfile
```

Snort viene avviato su entrambe le code.



### Terzo caso: più code a livello 3 (IP)

```
/sbin/ifconfig eth0 0.0.0.0 promisc up
/sbin/ifconfig eth1 0.0.0.0 promisc up
/usr/sbin/brctl addbr br0
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 eth1
/sbin/ifconfig br0 up
```

Si instaura il bridge fisico tra le interfacce eth0 ed eth1.

```
modprobe nfnetlink_queue
modprobe nf_conntrack
```

Sono i moduli che devono essere eseguiti per la creazione della multicoda.

```
echo "VALUE" > /proc/sys/net/nf_conntrack_max
```

E' un comando necessario alla creazione della multicoda, VALUE va sostituito dal n°sessioni che il kernel deve essere in grado di sorreggere tramite nf\_conntrack (es 262140).

```
echo "3" > /proc/sys/net/ipv4/tcp_fin_timeout
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
```

Comandi atti a sorreggere le sessioni a livello kernel.

```
sysctl -w net.core.rmem_default=8388608
sysctl -w net.core.rmem_max=8388608
sysctl -w net.core.wmem_max=8388608
sysctl -w net.ipv4.tcp_rmem='4096 87380 8388608'
sysctl -w net.ipv4.tcp_wmem='4096 65536 8388608'
sysctl -w net.ipv4.tcp_mem='8388608 8388608 8388608'
```

```
sysctl -w net.ipv4.route.flush=1
```

Settano i parametri ottimizzati per rete ad 1Gb/s.

```
/sbin/iptables -A FORWARD -i br0 -d "ip1" -j NFQUEUE --queue-num 43
```

```
/sbin/iptables -A FORWARD -i br0 -s "ip1" -j NFQUEUE --queue-num 43
```

```
/sbin/iptables -A FORWARD -i br0 -d "ip2" -j NFQUEUE --queue-num 44
```

```
/sbin/iptables -A FORWARD -i br0 -s "ip2" -j NFQUEUE --queue-num 44
```

```
/sbin/iptables -A FORWARD -i br0 -j NFQUEUE --queue-num 42
```

Selezionano il traffico proveniente e diretto verso gli "ip1" e "ip2" e lo convogliano rispettivamente nelle code 43 e 44.

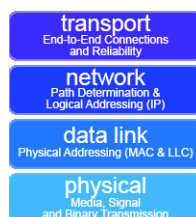
La coda 42 comprende tutto ciò che non è contenuto nelle code 43 e 44.

```
/usr/local/bin/snort_inline -A fast -b -d -D -Q42 -c /etc/snort/snort.conf  
-l /var/log/snort/42/ --nolock-pidfile
```

```
/usr/local/bin/snort_inline -A fast -b -d -D -Q43 -c /etc/snort/snort.conf  
-l /var/log/snort/43/ --nolock-pidfile
```

```
/usr/local/bin/snort_inline -A fast -b -d -D -Q44 -c /etc/snort/snort.conf  
-l /var/log/snort/44/ --nolock-pidfile
```

Snort viene avviato su tutte le code.



#### Quarto caso: due code a livello 4 (TCP/UDP)

```
/sbin/ifconfig eth0 0.0.0.0 promisc up
```

```
/sbin/ifconfig eth1 0.0.0.0 promisc up
```

```
/usr/sbin/brctl addbr br0
```

```
/usr/sbin/brctl addif br0 eth0
```

```
/usr/sbin/brctl addif br0 eth1
```

```
ifconfig br0 up promisc
```

Si instaura il bridge fisico tra le interfacce eth0 ed eth1.

```
modprobe nfnetlink_queue
```

```
modprobe nf_contrack
```

Sono i moduli che devono essere eseguiti per la creazione della multicoda.

```
echo "VALUE" > /proc/sys/net/nf_contrack_max
```

E' un comando necessario alla creazione della multicoda, VALUE va sostituito dal n°sessioni che il kernel deve essere in grado di sorreggere tramite nf\_contrack (es 262140).

```
echo "3" > /proc/sys/net/ipv4/tcp_fin_timeout
```

```
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
```

Comandi atti a sorreggere le sessioni a livello kernel.

```
sysctl -w net.core.rmem_default='8388608'
```

```
sysctl -w net.core.wmem_default='8388608'
sysctl -w net.ipv4.tcp_rmem='4096 87380 8388608'
sysctl -w net.ipv4.tcp_wmem='4096 65536 8388608'
sysctl -w net.ipv4.tcp_mem='8388608 8388608 8388608'
sysctl -w net.ipv4.route.flush=1
```

Settano i parametri ottimizzati per rete ad 1Gb/s.

```
/sbin/iptables -A FORWARD -i br0 -p tcp --dport "port" -j NFQUEUE --queue-num 43
/sbin/iptables -A FORWARD -i br0 -p tcp --sport "port" -j NFQUEUE --queue-num 43
iptables -A FORWARD -i br0 -j NFQUEUE --queue-num 42
```

Viene creata la coda 43 contenente il traffico tcp diretto e proveniente dalla porta "port", mentre nella coda 42 viene convogliato tutto il traffico restante.

```
/usr/local/bin/snort_inline -A fast -b -d -D -Q43 -c /etc/snort/snort.conf
-l /var/log/snort/43/ --nolock-pidfile
/usr/local/bin/snort_inline -A fast -b -d -D -Q42 -c /etc/snort/snort.conf
-l /var/log/snort/42/ --nolock-pidfile
```

Snort viene avviato su entrambe le code.

Si osserva che se per ogni coda venisse dichiarato un diverso file di snort.conf, nell'istruzione con cui si esegue Snort, allora si potrebbero personalizzare le regole applicate.

Al termine della compilazione del file rc.local, si creano le cartelle in cui riporre i log di snort, secondo la struttura imposta al sistema, in questo caso:

```
mkdir /var/log/snort
mkdir /var/log/snort/42
mkdir /var/log/snort/43 ..etc..
```

E' fondamentale riavviare il sistema, giunti a questo punto, cosicché automaticamente, all'avvio, vengano istituiti i bridge e venga lanciato snort.

## 7. Configurazione di snort

L'ultimo elemento da personalizzare è il file di configurazione di Snort: snort.conf.

Le sezioni che sono da configurare per questa specifica applicazione sono le seguenti:

- Definizione delle variabili di rete:
 

```
var HOME_NET [10.0.0.0/8]
```

Attraverso questa riga di comando si definisce la rete da difendere, in questo caso si considera a titolo esemplificativo la 10.0.0.0/8

```
var EXTERNAL_NET !$HOME_NET
var HONEYNET $HOME_NET
var DNS_SERVERS $HOME_NET
var SMTP_SERVERS $HOME_NET
```

```

var HTTP_SERVERS $HOME_NET
var SQL_SERVERS $HOME_NET
var TELNET_SERVERS $HOME_NET
var SNMP_SERVERS $HOME_NET
var SSH_PORTS 22
var HTTP_PORTS [80,443]
var SHELLCODE_PORTS !80
var ORACLE_PORTS 1521
var AIM_SERVERS
[64.12.24.0/24,64.12.25.0/24,64.12.26.14/24,64.12.28.0/24,64.12.29.0/24,64.12.
161.0/24,64.12.163.0/24,205.188.5.0/24,205.188.9.0/24]

```

Le istruzioni precedenti definiscono le variabili di rete

```
var RULE_PATH /etc/snort/rules
```

Dichiara dove si trovano le regole

#### ➤ Configurazione dei preprocessori:

I preprocessori, introdotti dalla versione 1.5, effettuano un primo controllo del traffico per prepararlo alla successiva analisi da parte delle regole.

Ogni preprocessore tratta una diversa tipologia di protocollo.

Innanzitutto si dichiarano i path dove trovare i preprocessori:

```

config checksum_drop : all
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
dynamicengine /usr/local/lib/snort_dynamicengine/libsfe_engine.so

```

L'istruzione che permette l'introduzione di un preprocessore è la seguente:

```
preprocessor <name>: <options>
```

I preprocessori normalmente impiegati sono:

- Il preprocessore frag3 ricompone il traffico frammentato:

```
preprocessor frag3_global: max_frag 65536
```

l'opzione impostata (max\_frag) riguarda il massimo numero di frammenti allineabili (di default questo valore è 8192)

```
preprocessor frag3_engine: policy first detect_anomalies ttl_limit 255
```

le opzioni impostate riguardano la modalità di deframmentazione (policy first), l'identificazione di frammenti anomali (detect\_anomalies) e il massimo delta ttl accettabile per pacchetti basati sul primo pacchetto del frammento (ttl\_limit 255)

- I seguenti preprocessori sono complementari. Essi si occupano dei protocolli TCP e UDP e vengono configurati nel seguente modo:

```
preprocessor flow: memcap 83886080, rows 8198, stats_interval 0 hash 2
```

Memcap definisce il numero di bytes da allocare, rows il numero di righe nella tabella di indirizzamento e hash il tipo di indirizzamento.

```
preprocessor stream4: disable_evasion_alerts, enable_udp_sessions,
norm_window, max_sessions 32768, memcap 1000000000
```

Disable\_evasion\_alerts disabilita le segnalazioni per eventi quali l'overlap del TCP, enable\_udp\_sessions abilita il tracking delle sessioni udp, infine

max\_sessions e memcap definiscono rispettivamente il massimo numero di sessioni e il numero di bytes da allocare

```
preprocessor clamav: ports 25, toserveronly, action-drop,
dbdir /var/lib/clamav, dbreload-time 3600
```

Il preprocessore clamav esegue un'azione di antivirus, in questo caso sulla porta 25 per il solo flusso diretto al server. I pacchetti che risultano affetti da virus vengono droppati e le regole per questa selezione sono contenute nella directory /var/lib/clamav. Clamav funziona solo con stream4 e di conseguenza con flow.

- Il preprocessore stream5 è impiegato in alternativa al precedente gruppo di preprocessori, poichè anch'esso si occupa dei protocolli TCP/UDP. In questo caso si è scelto di utilizzare stream5. Una configurazione di stream5 per questa applicazione è la seguente:

```
preprocessor stream5_global: max_tcp 131070, track_tcp yes, track_udp
yes, memcap 256000000
```

max\_tcp identifica il massimo numero di sessioni tcp contemporanee, le due opzioni successive permettono di analizzare i protocolli tcp e udp, memcap definisce il numero di bytes da allocare

```
preprocessor stream5_tcp: policy first, use_static_footprint_sizes
```

questa riga di comando configura stream5 nel trattamento del protocollo tcp

```
preprocessor stream5_udp: ignore_any_rules
```

questa riga di comando infine configura stream5 nel trattamento del protocollo udp

- Il preprocessore perfmonitor misura in tempo reale le performance di Snort:

```
preprocessor perfmonitor: time 60 file /var/log/snort/perfmon.txt pktcnt 500
```

 si definisce l'intervallo tra le misure e il file in cui vengono riportate le statistiche.

- Il preprocessore HttpInspect si occupa di decodificare il traffico HTTP e di identificare attacchi a livello applicativo che sfruttano eventuali vulnerabilità del protocollo http. La configurazione di questo preprocessore è divisa in due parti, una globale e una per i server:

```
preprocessor http_inspect: global iis_unicode_map unicode.map 1252
```

iis\_unicode\_map unicode.map 1252 deve essere sempre specificato in quanto contiene la global IIS unicode map

```
preprocessor http_inspect_server: server default \
```

```
profile all ports { 80 8080 8180 } oversize_dir_length 500 flow_depth 0
```

- Il preprocessore dcerpc analizza i protocolli SMB, DCE/RPC:

```
preprocessor dcerpc: \
```

```
ports smb { 139 445 } \
```

```
ports dcerpc { 135 } \
```

```
max_frag_size 3000 \
```

```
memcap 150000 \
```

```
reassemble_increment 0
```

vengono definite le porte su cui monitorare il traffico smb e dce/rpc, la massima dimensione del frammento e il numero di bytes da allocare.

- Il preprocessore smtp si occupa del flusso di posta ed è così configurato:

```
preprocessor smtp: \
  ports { 25 } \
  inspection_type stateful \
  normalize_cmds \
  normalize_cmds { EXPN VRFY RCPT } \
  alt_max_command_line_len 260 { MAIL } \
  alt_max_command_line_len 300 { RCPT } \
  alt_max_command_line_len 500 { HELP HELO ETRN } \
  alt_max_command_line_len 255 { EXPN VRFY }
```

vengono definite la porta su cui fare l'analisi del protocollo e la modalità operativa stateful. Infine si impostano le lunghezze delle linee di comando smtp che devono generare un alert per determinati servizi.

➤ Personalizzazione del set di regole:

In questa ultima sezione del file di configurazione si dichiarano le regole da implementare. In questo esempio:

```
include $RULE_PATH/classification.config
include $RULE_PATH/reference.config
```

Il comando attraverso il quale aggiungere le regole è:

```
include $RULE_PATH
```

seguito dal nome della regola. L'elenco completo delle regole è qui sotto riportato.

In base alle esigenze poi si selezionano le regole appropriate da applicare.

```
attack-responses.rules
backdoor.rules
bad-traffic.rules
chat.rules
community-bot.rules
community-deleted.rules
community-dos.rules
community-exploit.rules
community-ftp.rules
community-game.rules
community-icmp.rules
community-imap.rules
community-inappropriate.rules
community-mail-client.rules
community-misc.rules
community-nntp.rules
community-oracle.rules
community-policy.rules
community-sip.rules
```

*community-smtp.rules*  
*community-sql-injection.rules*  
*community-virus.rules*  
*community-web-attacks.rules*  
*community-web-cgi.rules*  
*community-web-client.rules*  
*community-web-dos.rules*  
*community-web-iis.rules*  
*community-web-misc.rules*  
*community-web-php.rules*  
*content-replace.rules*  
*ddos.rules*  
*dns.rules*  
*dos.rules*  
*emerging-attack\_response.rules*  
*emerging-botcc-BLOCK.rules*  
*emerging-botcc.rules*  
*emerging-compromised-BLOCK.rules*  
*emerging-compromised.rules*  
*emerging-dos.rules*  
*emerging-drop-BLOCK.rules*  
*emerging-drop.rules*  
*emerging-dshield-BLOCK.rules*  
*emerging-dshield.rules*  
*emerging-exploit.rules*  
*emerging-game.rules*  
*emerging-inappropriate.rules*  
*emerging-malware.rules*  
*emerging-p2p.rules*  
*emerging-policy.rules*  
*emerging-rbn-BLOCK.rules*  
*emerging-rbn.rules*  
*emerging.rules*  
*emerging-scan.rules*  
*emerging-virus.rules*  
*emerging-voip.rules*  
*emerging-web.rules*  
*emerging-web\_sql\_injection.rules*  
*experimental.rules*  
*exploit.rules*  
*finger.rules*  
*ftp.rules*  
*icmp-info.rules*  
*icmp.rules*

*imap.rules*  
*info.rules*  
*misc.rules*  
*multimedia.rules*  
*mysql.rules*  
*netbios.rules*  
*nntp.rules*  
*oracle.rules*  
*other-ids.rules*  
*p2p.rules*  
*policy.rules*  
*pop2.rules*  
*pop3.rules*  
*porn.rules*  
*rpc.rules*  
*rservices.rules*  
*scada.rules*  
*scan.rules*  
*shellcode.rules*  
*smtp.rules*  
*snmp.rules*  
*specific-threats.rules*  
*spyware-put.rules*  
*sql.rules*  
*telnet.rules*  
*tftp.rules*  
*virus.rules*  
*voip.rules*  
*web-activex.rules*  
*web-attacks.rules*  
*web-cgi.rules*  
*web-client.rules*  
*web-coldfusion.rules*  
*web-frontpage.rules*  
*web-iis.rules*  
*web-misc.rules*  
*web-php.rules*  
*x11.rules*

Per la descrizione dettagliata di tutti i parametri di configurazione di Snort si rimanda a:  
[http://www.snort.org/docs/snort\\_htmanuals/htmanual\\_283/](http://www.snort.org/docs/snort_htmanuals/htmanual_283/)