High Performance Firewall

Imagine this, you have more than two networks separated by Virtual Lans protocols (IEEE 802.1q) or VLANs, carried to you by an intelligent/manageable switch on one troncal line 10/100/1000 MB HD/FD (naturally the best is 1000 MB FD).

This page introduce how to create High Performance Firewall / Nat with iptables and VLANs and iproute2. Then you can share internet to a really BIG numbers of hosts, and maintain a good performance.

Contents

VLAN support

First, create sub-network as in page **VLAN**.

The round robin NAT

Let's suppose we have a one ip: 200.aaa.bbb.6 and our gateway is 200.aaa.bbb.1. we can safely put these parameters by default in our configuration. It will not get participation at all in our firewall.

I say I have 3 groups of 10 IPs each to play...... we'll define the NEXT in our firewall script:

```
Gr1='200.AAA.CCC.10-200.AAA.CCC.20'
Gr2='200.AAA.DDD.10-200.AAA.DDD.20'
Gr3='200.AAA.EEE.10-200.AAA.EEE.20'
```

And the next important line is:

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/21  -j SNAT --to $Gr1 #ACCESS VLAN 10
iptables -t nat -A POSTROUTING -s 192.168.8.0/21  -j SNAT --to $Gr2 #ACCESS VLAN 20
iptables -t nat -A POSTROUTING -s 192.168.15.0/21  -j SNAT --to $Gr1 #ACCESS VLAN 30
.... etc
```

You can repeat the groups for access, subdivide the networks ETC, iptables make a round robin over the Gr1, Gr2 and Gr3 by default, no modification is needed.

It's not necessary to create a virtual card (alias) to every IP in the group.

It's important that every real router knows every group and publishes its via BGP (or similar) to the neighbours.

tips

To accelerate some ports you can put this in the top of FORWARD chain

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p icmp -o eth0 -j ACCEPT
iptables -A FORWARD -p tcp -m multiport --dports 80,443,110,53 -j ACCEPT  # FAST FAST FAST
iptables -A FORWARD -p udp  --dport 53 -j ACCEPT
```

This mean:

- the packets incoming will pass only 1 rule if it is an establish connection
- the packet incoming will pass 2 rules if is a ping or similar
- the packet will pass 3 rules if is http, mail or similar
- and the DNS request will pass 3 o 4 rules until go out

The outgoing virus will KILL our machine, and we not need to share "windows" conversations so, kill them!!!!

```
 #VIRUS
iptables -A FORWARD -p tcp --dport 135:139 -j DROP
iptables -A FORWARD -p tcp --dport 445 -j DROP
iptables -A FORWARD -p udp --dport 135:139 -j DROP
iptables -A FORWARD -p udp --dport 445 -j DROP
```

If you can, before they reach our machine.

The High Performance

We get to the real important part of this howto.

In our run to get a really big number of hosts running through our machine we miss some things

1. We forget that is just one NICs to potentially more than 8000 Mac Addresses. The card shared memory is not prepare for this!!!!!
2. By default iptables is not prepared to make this number of connections simultaneously !!!!!!

So...

To the first issue... I get some error messages in the logs relative to this, I'm really sorry, I lost these logs and do not remember what they said. But the answer is this, increase the threshold memory to the neighbours. Type this and read:

```
# cat /proc/sys/net/ipv4/neigh/default/gc_thresh1
128
# cat /proc/sys/net/ipv4/neigh/default/gc_thresh2
```

```
512
# cat /proc/sys/net/ipv4/neigh/default/gc_thresh3
1024
```

Next you can put this in the /etc/sysctrl.conf

```
net.ipv4.neigh.default.gc_thresh1 = 512
net.ipv4.neigh.default.gc_thresh2 = 1024
net.ipv4.neigh.default.gc_thresh3 = 2048
```

and make *sysctl -p* to increase to the double!!! (no reboot needed) with this I get no errors!!!!!

The next part will need some comprehension about buckets and conntracks and hashsize (the way how iptables manage the nat connections). There is a very good document about this at **here (https://github.com/jeffmurphy/NetPass/blob/master/doc/netfilter_conntrack_perf.txt)**. Read it!!!! Some thing are change since IPtables is know as Netfiler.

In resume!!! Put this in your modules section:

```
MODULES=(8021q 'nf_conntrack hashsize=1048576' nf_conntrack_ftp
                               ...and other nf_stuff .......)
```

The last ones is just to avoid some problems that we have with ftp connections (I thing this is not necessary anymore). The '**nf_conntrack hashsize=1048576'** increase the numbers of the hashsize (increase the kernel memory designated to NAT connections) (need reboot or **reload module** :-) see with *dmesg* | *grep conntrack*)

And the next is put some similar to the `/etc/sysctl.d/99-sysctl.conf` : file

```
...
net.netfilter.nf_conntrack_max = 1048576
...
```

And do the *sysctl --system* command

In my case is the same number, that means that I have 1 connection for bucket!!!! I do not need more!!!! by default NetFilter put rate of 1:8. I.E. 8 conections per bucket!! (I think, not remember well)..

In our case we get about 600.000 simultaneous connections in 2 1Giga NICs cards, You can see this with the next command

```
# cat /proc/sys/net/netfilter/nf_conntrack_count
```

And put this in a snmpd agent to get and graph it in a MRTG/cacti server ..... uuuuuuu homework

The iproute2

We have 3 big access to Internet!!! This is because we manage 3 class C groups of IPs (some restrictions of BGP) in this firewall. So, we have 3 incoming traffics that we can manage, but only one outgoing!!! Our default gateway. This can easily fill our outgoing quote, so we have to spare it.

First we have to put some new tables to */etc/iproute2/rt_tables* file

```
# echo 200 PRO_1 >> /etc/iproute2/rt_tables
# echo 205 PRO_2 >> /etc/iproute2/rt_tables
# echo 210 PRO_3 >> /etc/iproute2/rt_tables
```

Can be more, can be less, depends on traffic

Second we have to give a default gateway to this tables

```
# ip route add default via 200.aaa.bbb.2 table PRO_1
# ip route add default via 200.aaa.bbb.3 table PRO_2
# ip route add default via 200.aaa.bbb.4 table PRO_3
```

It's recommended but not necessary put the local interfaces to each table. If you do not put the next few lines you will get not answer of ping in the local network, but you will be able to pass trough.

```
# ip route add 192.168.0.0/21 via 192.168.0.1 table PRO_1
# ip route add 192.168.8.0/21 via 192.168.8.1 table PRO_1
# ip route add 192.168.15.0/21 via 192.168.15.1 table PRO_1
.....
same PRO_2, same PRO_3
```

The last thing is to give the order to the incoming packages

```
# ip rule add from 192.168.0.0/21 table PRO_1
....
....
```

Again, you can play with the PRO_X and even you can play with the mask and submask For example we want to give only a one class C to outgoing to PRO_3

```
# ip rule add from 192.168.1.0/24 table PRO_3
```

Put this before the <NET>/21

Retrieved from "https://wiki.archlinux.org/index.php?title=High_Performance_Firewall&oldid=499951"