The Double NAT MINI-HOWTO

Yan-Fa Li <yanfali@best.com>, mailing list `netfilter@lists.samba.org` $Revision: 515 $

Contents

1 Introduction

Like many small companies we are typical in our use of

RFC1918 `http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1918.html` . addressing. Specifically the 192.168.0.0/16 range. Because we needed some untrusted network access, as a security precaution I made the dubious decision to use overlapping address ranges that deliberately clashed with our own corporate network use to make it especially hard for someone to hack our main network if the untrusted network were compromised.

Unfortunately those untrusted networks have become increasingly more and more important and developers at our company needed use them. This became a problem because access was limited to a few physical locations. Now you know the reasons why I went through all this rigamarole, here's how I accomplished it.

This mini-howto is for 2.4 kernels and ip netfilter. All the examples were tested on a 2.4.18 kernel.

2 The Situation

```
ASCII Art 1: The situation


        Network 1
        192.168.150.0 (Corporate)

       --------------/-------------------------



        Network 2
        192.168.150.0 (Untrusted Network)
```

How to hook them up ? As you can see, both subnets have overlapping network address ranges.

3 First Attempt: Single NAT

On my first try I simply used a third intermediate network which was routed to network 1 and connected the NAT box in-between.

```
ASCII Art 2: Attempt 1

          Network 1
        192.168.150.0 (Corp)
              |
            Router
              |
           Network 2
        192.168.180.0 (Intermediate)
                   |
          NAT BOX  |
          eth0 192.168.180.180
          eth1 192.168.150.180
                   |
        Network 3  |
        192.168.150.0 (Untrusted Network)
```

Using a combination of Destination NAT mappings and a Source NAT remapping, this was a partial success. I could access all the stuff on Network 3 from any network that WASN'T network 1. Why ? Localhost routing policy on the NAT box.

The local routing policy on a normal linux system is usually:

- Directly Connected Interfaces
- Static Routes
- Default Routes

In that order. So since I was directly connected to 192.168.150.0 there was no way for the NAT box to talk to Network 1 since it was technically already directly connected to it, or at least it's doppleganger. I'm sure there's some way to overcome this using Alexey's ip util but I did not look into this further.

4 Second Attempt: Double NAT

Double NAT is one of those unholy things that makes network administrators cringe. Their bones crackle and sinews twist as they think of the nasty things that a) NAT does and b) Double NAT does doubly. It's a hack, but when you have a situation where you have two subnets that are the same addressing that wish to communicate, it's a useful hack because it allows you to de-couple them into separate address space.

```
ASCII Art 3: Attempt 2

        Network 1
        192.168.150.0 (Corp)
                         |
        Network 2        |
        192.168.180.0 (Intermediate)
                                |
                NAT BOX 1       |
                eth0 192.168.180.180
                eth1 10.15.15.1
                            |
                NAT BOX 2   |
                eth0 10.15.15.2
                eth1 192.168.150.252
                                |
        Newtwork 3              |
        192.168.150.0 (Untrusted Network)
```

Yeah, like I said it's pretty satanic. So first step, host preparation.

5 Rusty's Really Quick Guide To Packet Filtering Copied:

I will assume you've used Rusty's really quick guide to packet filtering and you already have 2 systems prepared for NAT. Check the Packet Filtering FAQ for more details.

```
'iptables -vL' will probably look something like this:

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in     out     source               destination
 2434  219K block       all  --  any    any     anywhere             anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
11657 5137K block      all  --  any    any     anywhere             anywhere
```

```
Chain OUTPUT (policy ACCEPT 2514 packets, 217214 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       icmp --  any    any     anywhere             anywhere           state INVALID

Chain block (2 references)
 pkts bytes target     prot opt in     out     source               destination
13579 5319K ACCEPT     all  --  any    any     anywhere             anywhere           state RELATED,ESTABLISH
  512 37065 ACCEPT     all  --  !eth1  any     anywhere             anywhere           state NEW
    0     0 DROP       all  --  any    any     anywhere             anywhere
```

translated it will look something like this:

```
iptables -N block
iptables -A INPUT -j block
iptables -A FORWARD -j block
iptables -A OUTPUT -p icmp -m state --state INVALID -j DROP
iptables -A block -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A block -i ! eth1 -m state --state NEW -j ACCEPT
iptables -A block -j DROP
```

Don't forget to turn IP forwarding on, otherwise this probably won't work. On a redhat 7.[2|3] system after entering this you can simply type:

```
iptables-save > /etc/sysconfig/iptables
```

This will create your iptables config file which will load automagically at every boot up.

6 DOUBLE NAT

6.1 Assign alias IP addresses to the eth0 interfaces of NAT BOX 1 and 2

The key to making this work is to remap the IP addresses you are interested in accessing to a different subnet. For our example, let's say we interested in communicating with address range 192.168.150.10 through 12 on Network 3.

On NAT BOX 1, create 3 new alias interfaces on eth0, e.g.

```
ifconfig eth0:0 192.168.180.181 netmask 255.255.255.0
ifconfig eth0:1 192.168.180.182 netmask 255.255.255.0
ifconfig eth0:2 192.168.180.183 netmask 255.255.255.0
```

On NAT BOX 2, create 3 new alias interfaces on eth0, e.g.

```
ifconfig eth0:0 10.15.15.181 netmask 255.255.255.0
ifconfig eth0:1 10.15.15.182 netmask 255.255.255.0
ifconfig eth0:2 10.15.15.183 netmask 255.255.255.0
```

6.2 Create Static NAT Mappings on NAT BOX 1

On NAT BOX 1, create 3 new mappings on eth0:

```
iptables -t nat -A PREROUTING -d 192.168.180.181 -i eth0 \
        -j DNAT --to-destination 10.15.15.181
iptables -t nat -A PREROUTING -d 192.168.181.182 -i eth0 \
        -j DNAT --to-destination 10.15.15.182
iptables -t nat -A PREROUTING -d 192.168.181.183 -i eth0 \
        -j DNAT --to-destination 10.15.15.183
```

and 1 SOURCE NAT map:

```
iptables -A POSTROUTING -s 192.168.150.0/255.255.255.0 \
        -d 10.15.15.0/255.255.255.0 -j SNAT -o eth1 \
        --to-source 10.15.15.1
```

This takes care of NAT BOX 1, and allows you to talk to NAT BOX 2.

You can test it at this point and make sure that NAT box 1 is set up. Typically ssh'ing to one of the alias addresses will allow you to login to NAT box 2 and make sure you haven't done anything silly. If this works then you are good to go.

6.3 Create Static NAT Mappings on NAT BOX 2

On NAT BOX 2, create 3 new mappings on eth0:

```
iptables -t nat -A PREROUTING -d 10.15.15.181 -i eth0 \
        -j DNAT --to-destination 192.168.150.10
iptables -t nat -A PREROUTING -d 10.15.15.182 -i eth0 \
        -j DNAT --to-destination 192.168.150.11
iptables -t nat -A PREROUTING -d 10.15.15.183 -i eth0 \
        -j DNAT --to-destination 192.168.150.12
```

and 1 SOURCE NAT map to our target destinations:

```
iptables -A POSTROUTING -s 10.15.15.0/24 \
        -d 192.168.150.0/24 -j SNAT -o eth1 \
        --to-source 192.168.150.252
```

You should now be done. Now it is possible to access 192.168.150.150 through 152 on network 3 via the addresses 192.168.180.181-183 on network 2 from network 1. I told you it was evil.

7 Thanks

Thanks to the entire netfilter team. You guys rock.