



Free firewall software distributed under [GNU General Public License](#)

## Start

- [Log](#)

## About

- [Short description](#)
- [Pronunciation](#)
- [Who's behind eatables?](#)
- [Licence](#)
- [About the code](#)

## Downloads

- [Latest release](#)
- [CVS repository](#)

## Documentation

- [What is eatables?](#)
- [Main features](#)
- [What can it do?](#)
- [What can't it do?](#)
- [Documents](#)
- [Todo](#)
- [Links](#)

## Examples

- [Simple examples](#)
- [Real-life examples](#)

## Contact

- [Mailing lists](#)
- [Webmaster](#)

## Example 1: Linux brotting, MAC snat and MAC dnat all in one

This setup was given by [Enrico Ansaloni](#), who got things working together with [Alessandro Eusebi](#).

### Short description

Two 802.1Q VLANs, a HP 4000 M switch and a Linux bridge with iptables and eatables. The HP switch is used for VLAN switching. The Linux bridge is used for bridging specific frames and for IP routing. Obviously good filtering is required to prevent duplicated traffic.

### Enrico's story

Why I use the Linux bridge: I have to bridge the 2 VLANs to let DEC LAT traffic only go through, while IP traffic has to go under normal routing decisions: this is done with eatables.

The bridge has 2 IP addresses, one for each VLAN so that routing will work, in conjunction with iptables eventually to do some traffic shaping and internal control (allowed tcp ports from inside to outside etc.)

### First try: assign the 2 IP addresses to the bridge device (br0)

The problem with the HP switch is that it doesn't allow the same MAC address in 2 different ports which is what happens when I connect the bridge which has 2 cables - it connects into 2 ports - and has 1 MAC address; this occurs even if the ports belong to different VLANs: to put it in another way, the HP VLAN implementation doesn't fully isolate VLANs but checks for MAC address consistency in the whole switch, regardless of VLAN settings... instead, the Cisco Catalyst series does that right but it costs 3 times more and my customer won't pay that much...

### Second try: use eatables MAC snat to give the two bridge IP addresses different MAC source addresses

so I decided I could try with MAC natting, and I set up eatables like this:

```
eatables -t nat -A POSTROUTING -o $INSIDE_IF_NAME -s $DMZ_IF_MAC -j snat --to-source $INSIDE_IF_MAC
eatables -t nat -A PREROUTING -i $INSIDE_IF_NAME -d $INSIDE_IF_MAC -j dnat --to-destination $DMZ_IF_MAC
```

As you can see, I'm trying to fool the switch into thinking the bridge has a different MAC address for each interface: the rule is correct, as I can see by capturing traffic with ethereal, but there's a problem: eatables works at layer 2 only, thus correctly natting MAC address in the layer 2 ethernet frame; the switch now accepts the natted packets but the ARP packets are at layer 3 and they keep the original MAC address as from the linux kernel network stack, so the client's reply is wrong and never goes through...

### Third try: use brouting + MAC snat + MAC dnat

As stated in the Examples section on the eatables hp, I started with this:

```
*****
Bridge table: broute

Bridge chain: BROUTE
Policy: ACCEPT
nr. of entries: 4
1. -p IPV4 -i eth0 -j DROP , count = 47959
2. -p IPV4 -i eth1 -j DROP , count = 47
3. -p ARP -i eth0 -j DROP , count = 371
4. -p ARP -i eth1 -j DROP , count = 141
*****
```

But this is not enough... With this rule, IP routing and ARP stuff is perfectly working for both networks (VLANs) and I can control IP stuff with iptables, but bridge isn't working yet... that's because of the duplicate MAC in different VLANs problem of the HP switch: when I use bridging, the same MAC address of the bridged client appears on two ports, one for each VLAN, and the switch automatically deactivates the first port! But I solved this issue with eatables MAC NAT, like this:

```
*****
Bridge table: nat

Bridge chain: PREROUTING
Policy: ACCEPT
nr. of entries: 2
1. -d 10:50:da:e7:18:51 -i eth1 -j dnat --to-dst 0:50:da:e7:18:51 --dnat-target
ACCEPT, count = 1260
2. -d 10:10:a4:9b:30:d -i eth0 -j dnat --to-dst 0:10:a4:9b:30:d --dnat-target
ACCEPT, count = 1252

Bridge chain: OUTPUT
Policy: ACCEPT
nr. of entries: 0

Bridge chain: POSTROUTING
Policy: ACCEPT
nr. of entries: 2
1. -s 0:50:da:e7:18:51 -o eth1 -j snat --to-src 10:50:da:e7:18:51 --snat-target
ACCEPT, count = 1362
*****
```

```
2. -s 0:10:a4:9b:30:d -o eth0 -j snat --to-src 10:10:a4:9b:30:d --snat-target
ACCEPT, count = 1346
*****
```

The MAC addresses you see are two network client, one for each VLAN. When a client passes the bridge, I have to change his MAC (i change the first 00: with 10:) in order to make the switch happy. I did a small script also, so you can specify a list of MAC addresses for each VLAN. Here's the script:

```
*****
#!/bin/bash
#####
# EBTables test script
#####
# Binaries

EBTABLES=/usr/local/sbin/ebtables
#####
# Interface names

INSIDE_IF_NAME=eth0
DMZ_IF_NAME=eth1
BRIDGE_IF_NAME=br0
#####
# Bridge mac address list

INSIDE_IF_MAC="00:04:76:14:74:99"
DMZ_IF_MAC="00:01:03:e2:e9:4c"
#####
# Client mac address list

LAN_CLIENT_MACS="00:50:DA:E7:18:51 00:50:DA:E7:F1:A0 00:10:A4:9B:E8:21"
DMZ_CLIENT_MACS="00:10:A4:9B:30:0D 00:01:03:E2:12:9C 00:50:DA:E7:11:2B"
NEW_PREFIX="10:"
#####
# Set default policy
#
$EBTABLES -P INPUT ACCEPT
$EBTABLES -P OUTPUT ACCEPT
$EBTABLES -P FORWARD ACCEPT
# clear existing tables
$EBTABLES -F
$EBTABLES -t nat -F
$EBTABLES -t broute -F
```

```
#####
# BRoute

$EBTABLES -t broute -A BROUTE -p ipv4 -i $INSIDE_IF_NAME -j DROP
$EBTABLES -t broute -A BROUTE -p ipv4 -i $DMZ_IF_NAME -j DROP
$EBTABLES -t broute -A BROUTE -p arp -i $INSIDE_IF_NAME -j DROP
$EBTABLES -t broute -A BROUTE -p arp -i $DMZ_IF_NAME -j DROP

#####
# Bridged clients

for MAC in $LAN_CLIENT_MACS; do
    NEW_MAC="${NEW_PREFIX}`echo ${MAC} | cut -f2- -d':'`"
    $EBTABLES -t nat -A POSTROUTING -o $DMZ_IF_NAME -s $MAC -j snat --to-source
$NEW_MAC
    $EBTABLES -t nat -A PREROUTING -i $DMZ_IF_NAME -d $NEW_MAC -j dnat --to-
destination $MAC
done

for MAC in $DMZ_CLIENT_MACS; do
    NEW_MAC="${NEW_PREFIX}`echo ${MAC} | cut -f2- -d':'`"
    $EBTABLES -t nat -A POSTROUTING -o $INSIDE_IF_NAME -s $MAC -j snat --to-
source $NEW_MAC
    $EBTABLES -t nat -A PREROUTING -i $INSIDE_IF_NAME -d $NEW_MAC -j dnat --to-
destination $MAC
done

#####
# END
#####
```

I hope this can be useful to you or someone else... if you're so unlucky to have to deal with HP 4000 switches and their VLAN implementation :)