Network virtualization for firewalls

Hendrik Visage

2025/04/24

Outline

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

Topic

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

List of reasons

- no physical hw available
- unused PCIe slots, space saving
- All your dockers, VMs and containers use it
- All open source
- when you can't do/use/get OEM branded device to do stuff

Topic

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

links:

https://datahacker.blog/industry/technology-menu/networking/a-brief-history-of-linux-networking

First steps (90s)

Dial up routing

- only modems, no "home routers"
- mostly use with modems (SLIP/PPPD)
  - roe-pppd with ADSL on the 2nd interface
- enable IP forwarding
- IPs on interfaces, and ‘route add‘
  - RIP the flavor of the year

First steps (90s)

First firewalls

- Abundant IPs
- Commercialization of Internet Started
- ipfw (still in *BSDs)
- statefull packetfilters
  - CheckPoint FireWall-1 (Typically Solaris, but later Windows NT)
  - SunScreen SPF100 (Dedicated, but just a SunSparc 5)

2000s

Major news items:

- IPv4 shortage prophesied
- No more 10base2
  - Switching standard

Linux 2.2-2.6

- ipchains - stateless
  - NAT become a thing - MASQUERADE
- followed by statefull iptables
  - connection tracking
  - SNAT/DNAT
  - port NAT
  - and other inflight packet butchering
- 2.2 introduced bridge
  - includes STP
  - ‘ebtables‘ - like MAC separations
  - packet sniffing with tcpdump
    - no need for a switch with port mirrors
    - Laptop with 2x NICs and you become a network god telling the client what is happening
  - multiple devices on ADSL accounts
  - fileserver with couple home PCs sharing a upstream "home" router.

VLANs

- Published 802.1Q-1998 - IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks
- Various initial OEM names
  - trunking
  - ether channel
  - ISL (Cisco)
- 802.1q in Linux 2.2.13/2.3.99
  - names like vlan0005, vlan5, eth0.0005, eth0.5

Virtual Interfaces (TUN/TAP)

- Universal TUN/TAP driver
  - Copyright © 1999-2000 Maxim Krasnyansky & Maksim Yevmenkin
- TUN
  - Point-to-point "tunnel" (IP packets without Ethernet headers)
  - typical VPN routed
  - IP assigned for local and remote.
  - like where pppd connects/provide IP
- TAP (Ethernet frames)
  - comparable to local ethernet plugged into to remote ethernet
  - "remote" appears on the local network as a "real" device.
  - can do a DHCP to get IP on the remote
- commonly used in VPN software

Possibilities in 2000s

- Everything to be a full blown router and/or switch
  - zebra for BGP/OSPF/RIP/etc.
  - kernel routing
  - bridging
  - filtering on L3/4 with iptables
  - filtering on L2 with ebtables
- VPN concentrators
  - road-warriors connecting to local server as if "on site"
  - DSL services
- name a network function, and we can now do it.

Limitations

- port density
  - at best dual 100Mbps in PC with PCI
  - Sun SBUS with quad options
  - Even today the standard PCIe form is space consuming
    - from the 1980s IBM PC era, meant for Desktops and server that needs cooling for the components on the adapters
- Single CORE CPUs
  - Multi sockets shared bus
- I/O typically on a South bridge chipset so "slow" moving packets.
- Well geared for stuff that needs processing (like VPNs/firewalls) together with some network switching

Virtualization (on x86)

- VMWare Desktop launched 1999
- Xen (started 1999) opensource release 2003
- QEMU 0.1.6 in 2003
- Linux KVM 2006 (used by QEMU/libvirt)
- "BHyVe" FreeBSD ~2009
- Various containers/zones/jails tech since 2001

Using guests as the collective for VMs and containers

Networking in virtualized environments

All guests needs network of some sort

- And they assume it will be Ethernet

Question just remains where/how to connect the network endpoints typically either:

- bridge setup with LAN interface
- bridge setup with a NAT gateway (iptables) to LAN interface

First introduction ~2013

- COZA DNS advance course
  - each student had his own "network" of servers (guests)
  - each network separate from the other
  - ipfw to NAT to each student's set of DNS servers
  - then a router to connect all those to be able to do DNS slave and queries to other servers.

Topic

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

Basic setup

- Each "network" is a bridge interface (br1/br2/br3)
- Each student server, connects to the relevant bridge (ss1a & ss1b to br1, ss2a & ss2b to br2)
- router that connects to all those bridges (br1/br2/br3) and does the needed DHCP/firewall/routing/etc.

All that, hosted inside a single machine

with multiple cores and enough RAM to handle the guests

Introduction to ProxMox

- Initially only Linux bridge (no VLAN support yet)
  - simple:
    - all Guests connects to the same bridge (vmbr0)
    - bridge connect to LAN (eth0)
    - All same broadcast network as the rest of the PCs/Servers on the LAN
  - Firewall upgrade at home
    - Add extra ethernet interface (eth1)
    - add extra bridge (vmbr666)
    - connect eth1 to vmbr666
    - connect virtual firewall to vmbr666 & vmbr0
    - "simple" option Linux VM
    - pfSense had issues as only e100 matched
    - and you want to disable HW offloading on the e100 inside pfSense else DHCP fails strangely

Introduction to ProxMox (continue..)

- Got introduced to OpenVSwitch
  - 802.1q VLANs !!!
  - OVS "Ports" (just a TAP like interface to the OVS switch)
  - Now the eth0 gets connected to VLAN1 on vmbr1
  - eth1 gets connected to VLAN666 on vmbr1
    - Cisco: access tagged 666
  - Guests that users direct access connected to VLAN1 on vmbr1 (so the LAN eth0 also sees them)
  - Guests in the DMZ gets connected to VLAN321 on vmbr1 (not seen by the LAN or WAN interfaces)
  - FireWall now connects on VLAN666 (wan), VLAN321 (dmz) and VLAN1 (LAN) and route, NAT and filter accordingly

Then we have a cluster

- The simple approach on ALL the nodes with all the L2 caveats and emptors:
  - eth0 WAN to VLAN666 on vmbr0
  - eth1 the .1q trunk interface to vmbr0
    - Cisco: trunk port
  - Guests separated in their own VLANs
    - this is trunked between nodes on the eth1 LAN
  - firewall connected using VLAN interfaces to guest vlans & VLAN666
    - theory: firewall can be on any host node and have internet break out and all the guest vlan access

Then we have a cluster (p2)

1st problem in big server hosting environments:

Public IP bound to specific eth0 interface.

Solution:

- Node1: Public eth0 to VLAN667 vmbr0
- Node2: Public eth0 to VLAN668 vmbr0
- eth1 trunked to bridge vmbr0
- Firewall on node2 now can have IP on VLAN667 and exit on Node1 for these VLAN "routes"
  - does work great when firewall is on node1 using VLAN667 or node2 using VLAN668
  - also when public and "back end" is separate switching networks

Then we have a cluster (p2)

2nd problem big hosting provider

ports on shared switching environments (eth0 & eth1 same switching fabric) doesn't like these same macs to enter/ exit (even with different VLAN tags) on different interfaces,
ie. Firewall on node2, using VLAN667 to traverse via trunked eth1 to node1 eth1 then exit on node1 eth0 to the Public/WAN

- problem is the destination-source MACs are before the .1q tags
- switch looks at MAC forwarding tables, notice the mac has 2x exit destinations
- one port gets locked, and you then have to get network admin to unlock the port and debug the issue for a month or more

Cluster have Public internet on the vRack backend

- Also now provide a LACP BOND for redundancy on the .1q-in-q interface
- No more need to have the PublicIP shenanigans
- Different bridge options:
  1. Linux bridge (since it now do support VLANs)
     - create the BOND0 interface with the needed LACP settings
     - attach the BOND0 to the bridge vmbr0
  2. OpenVSwitch (Personal preference)
     - do NOT use the Linux BOND interface!
     - create the bond interface as part of the openVSwitch configuration for vmbr1
     - do the needed native VLAN and tagging stuff

Cluster have Public internet on the vRack backend (Continue)

- native (untagged) is public internet router created on vRack
- still have .1q tags for guest separation across nodes
- firewall "WAN" interface now bound to the VLAN666 that is untagged on the BONDed interface

  Been stable use for more than four years n 2x clusters
- Caveat: Not every hosting provider has bigIron OEM hardware to provide a stable L2 VNI across DCs

Rinse repeat on cheap OEM switches locally

- Same basic LACP BOND as before
- MLAG to have switch redundancy
- had to enable the vlan tags on ports (didn't look into q-in-q at the time)
  - Issue when you also need to use some VLANs for other networks/clients/users in same switching fabric.
  - ie. you are sharing the .1q VLAN space so cluster admin and network admins needs to agree on who use which VLAN tags

Rinse repeat on cheap OEM switches locally (continue)

- It worked(TM)
  - Single stream L2 fun: VM1@node1 -> VM2@node2
    - VM1.Node1 - exit enp1 to Switch1/eth1
    - Switch1 sent via MLAG to Switch2
    - Switch2/eth2 sent to Node2.enp2 I've observed return traffic doing similar but from node2.enp2
  - Noticed other L2 challenges when deployed similar across DCs with L2 vlans extended via same switches

Topic

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

Not using OEM hardware switches

So how about NOT using L2 OEM switches between cluster nodes?

- Could use Linux bridges (Yes, use this mechanism)
  - Linux kernel bridge has STP support
- Or switch to L3 ?

VxLAN and FRR

- First step is that you have an IGP that routes loopback IPs
  - Personal preference openfabric based on IS-IS
  - OpenFabric an expired draft, and IS-IS to inherit the "settings"
  - Issue with OpenFabric and reason for move to IS-IS:
    - OpenFabric assumes PtP links ONLY

VxLAN and FRR (cont)

- VxLAN between nodes connected to the Loopback IPs
- Linux bridge is then connected to a VxLAN VNI
  - Note: OpenVSwitch at this stage isn't (yet) doing EVPN for VxLAN without an VTEP emulator
  - Also note you'll see ALL the 4096 .1q VLAN tags in the L2 with local MACs
- Could have more than one bridge, each with different VNIs, for eg. ceph or another client with their own .1q
- issues:
  - Linux ifupdown VxLAN was not supporting IPv6 endpoints when I first started looking into this
    - manual settings needed

Topic

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

Todays 'cheap' Hardware and VPP

- PCIe quad port 10G SPF+ HH adapters (x710)
- AIOM/SIOM compact profiles with quad SPF28 ports (MCX4)
- PCIe dual SPF28/SPF+ HH "default"
- 2U quad node with AIOM + 2x HH PCIe3 8x with options like:
  - 4x SFP+ (AIOM) + 2x4x SFP+ (2xHH PCIe)
  - 4x SFP28 (AIOM) + 2x4x SFP+ (2xHH PCIe)
  - 4x SFP+ (AIOM) + 2x4xSFP28 (2xHH PCIe)

ie. possible to have 4x router/switches with 12ports each in a 2U space

Yes, yes, it's not the 2x48 SFP+/SFP28 ports, but do they also include 6x2TB SSDs in EACH of the 4nodes?

Topic

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

Throwing out OEM routers and Switches (real life)

- We have ALL the building blocks now to have:
  1. a switched network between hosts to start
  2. a mesh network with L3 redudancies (yes, FRR on each node/server)
  3. with VxLAN we now can put redundant VLANs across hypervisor/server nodes
     - across DCs if need to
  4. My only "switch" is the OOB/IPMI/iDrac RJ45 switch connected to a separate OOB router device.

Throwing out OEM routers and Switches (real life) continued

For single rack server hosting setup, you could do redundant upstream and peering connections all inside and with your ProxMox server stack.

- Inside a 2U space with 4x routers with 8-12 ports per "router" node
- Exercise for the reader: how to mesh effectively

My Future

- VPP to be configured
- grow client base

Topic

1. Why use virtualized networking?
2. A Brief History of Linux Networking
3. How to virtualize Firewalls
4. Where to Next?
5. Future endeavours
6. Conclusion
7. Q&A

Questions?

Don't ask the difficult long answer questions now :)

- hvisage@hevis.co.za
- `https://t.me/hvisage`