Branch VPN Solution with OpenBSD

EuroBSDCon, Paris, September 24, 2017

Remi Locherer <remi.locherer@relo.ch>

Branch VPN Solution with OpenBSD

In 2016 we started deploying a new VPN solution for connecting the branch offices of Netcetera.

You might be interested in this presentation if you are a

System Engineer:	Learn something about networking.
Network Engineer:	See what cool stuff is possible with OpenBSD!
OpenBSD Developer:	See how the software you write is beeing used.

I expect to learn from the audience all the things we did wrong ;-).

Who am I?

System and network engineer with Netcetera since 2009.

First OpenBSD release I used: 3.0

First exposure to networking:
10BASE2 Ethernet connecting Apple Quadras in an architects office.

The setup I'm going to present is a joint development with Daniel Stocker.

Situation before

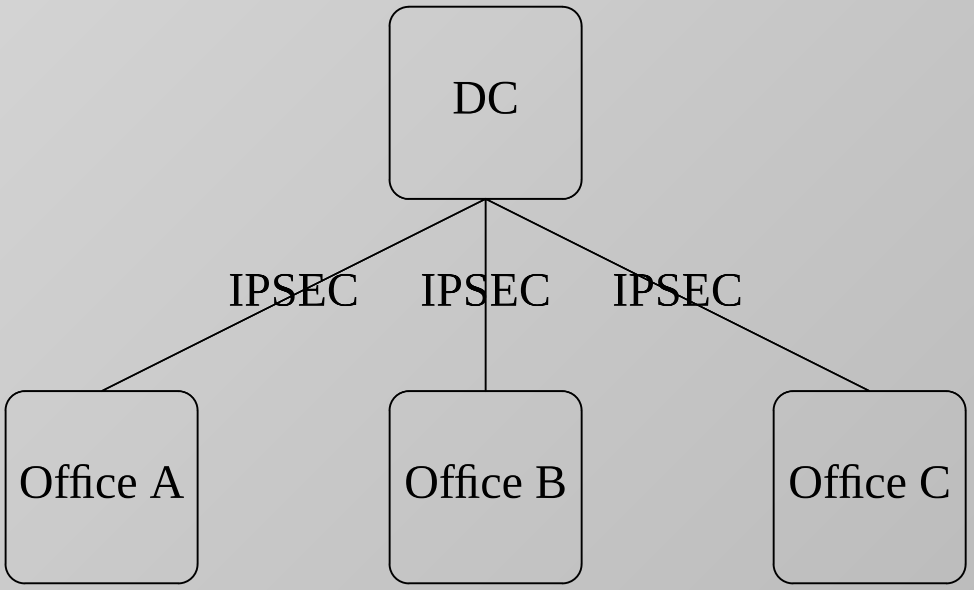
- 4 branch offices connected to HQ
- 4 different platforms
  - Cluster with OpenBSD isakmpd and OpenVPN on HP servers
  - CentOS with StrongSwan on DELL server
  - CentOS with OpenVPN on HP home NAS
  - SnapGear appliances with OpenVPN
- Making new networks available to all locations was manual and error prone task
- SnapGears to slow for available bandwidth (and EOL)

Goals for a new setup

- Less manual work
- Less different technologies to manage
- Enough encryption performance for all branch offices (fastest link 100Mbps)

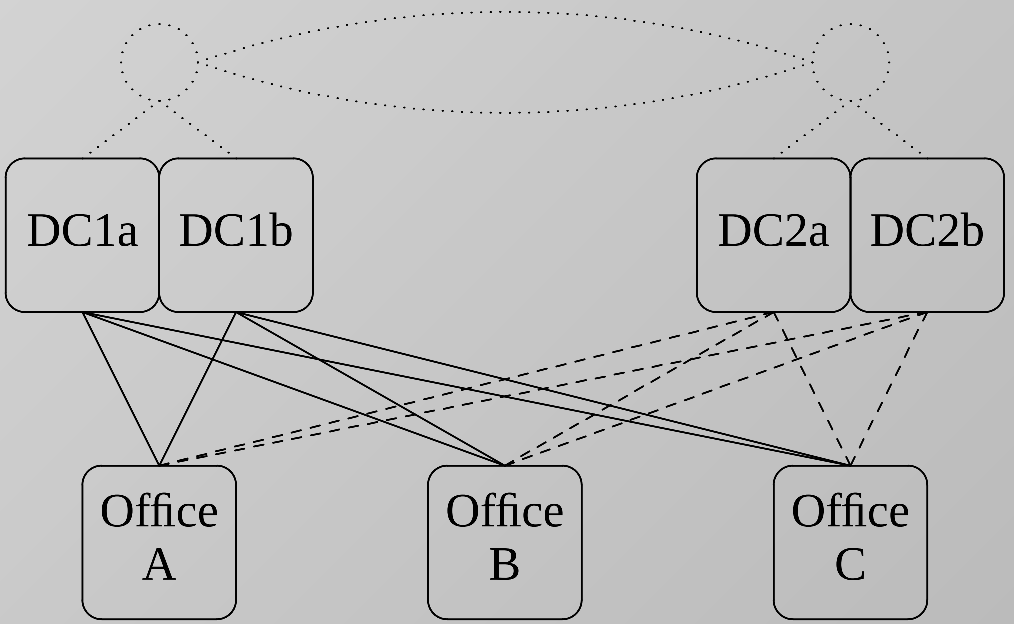
Network setup

- Connect all branch routers to the data center over public Internet.
- IPSEC for confidentiality and integrity



Network setup - Redundancy

- Clustered hubs
  - First hop redundancy with CARP
  - Redundancy based on routing protocol towards other routers
- Hub in 2nd data center



Network setup - Routed IPSEC

- Old setup: more than 200 flows for one branch office!
- Routing to move traffic into the VPN
  - Scales better compared to a flow based setup (rekeying)
  - Easier to debug
  - Faster to make new networks available on all sites
  - Routing for redundancy

IPSEC transport mode in combination with tunnel interfaces.

Network setup - tunneling

gre(4):	no need for keepalives we only need support for IP transport additional GRE header reduces MTU
etherip(4)	we do not want to transport Ethernet frames • MTU • additional L2 traffic
gif(4)	IP in IP same encapsulation as IPSEC tunnel mode provides an interface that we can use for routing!

Network setup - Routing protocol

Symmetric traffic flow is a requirement: The routers are actually stateful firewalls!

BGP

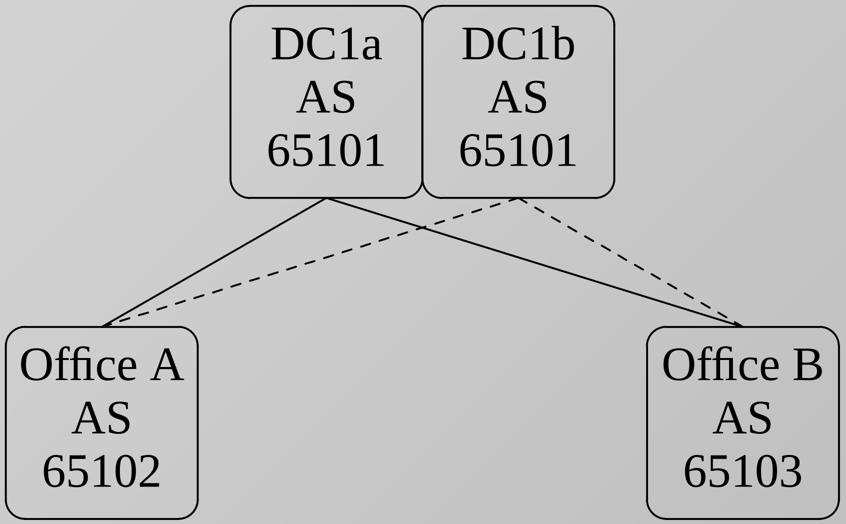
- Allows us to implement policies.
- Scales to very large network sizes.

OSPF

- Fast convergence.
- In use in the backbone.
- Downsides:
    - Every time we loose a link to a branch router all routers recalculate their routes.
    - No route filtering: configuration error on a branch router could bring down the hole company!

Network setup - BGP

- Private AS number per site.
- AS path prepending (dashed line) on link to backup hub router.



Network setup - BGP config branch router

- network (inet|inet6) connected
- 2 neighbors (hub routers)
- announce self

Network setup - BGP config hub router

- network (inet|inet6) rtlabel "fromOSPF"
- neighbor template
- CARP backup: prepend self (AS path prepending)
- ifstated reloads bgpd with changed config depending on carp state
- redistribute bgp routes into ospfd based on route tags

It gets complicated once the branch routers should also connect to the hub in the 2nd data center.

Network setup - OSPF config branch router

- All routers in area 0.
- Branch routers must be stubs.

```
router-id 192.0.2.90
stub router yes

area 0.0.0.0 {
        interface gif8
        interface gif9
        interface lo1 { passive }
        interface vlan331 { passive }
        interface vlan500 { passive }
}
```

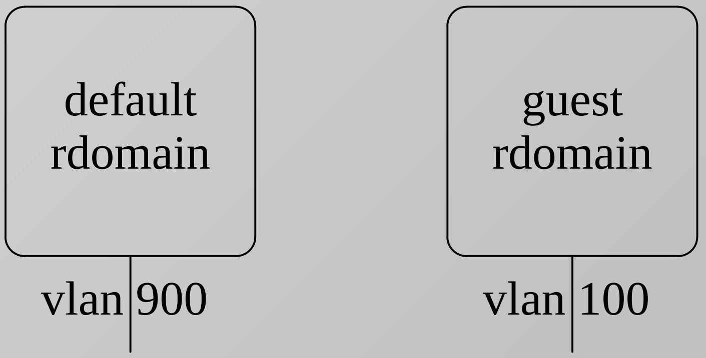
Network setup - OSPF config hub router

```
router-id 192.168.8.85
include "/etc/ospfd.mymetric"
redistribute default set { metric $mymetric type 1 }
redistribute rtlabel toOSPF set metric $mymetric

area 0.0.0.0 {
        interface vlan10 { metric $mymetric }

        interface gif0 { metric $mymetric }
        interface gif2 { metric $mymetric }
        interface gif4 { metric $mymetric }
        [..]

        interface lo1 { passive }

        interface carp800 { passive }
        interface carp801 { passive }
        interface carp870 { passive }
        interface carp900 { passive }
        interface carp901 { passive }
}
```

Network setup - Guest network

- Guests are provided with Internet access.
- Guests must not have access to company network.

OpenBSD rdomains provide this separation.

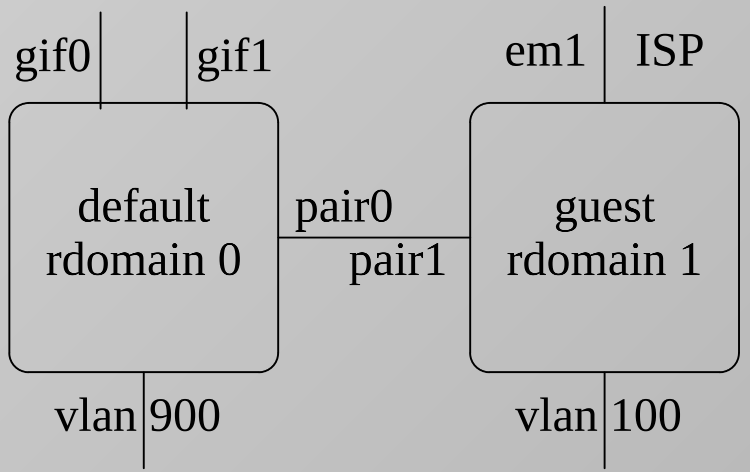


Where should we connect the ISP router?

Network setup - Default route

Local exit for Internet traffic. (Routing all traffic over IPSEC should also be possible.)

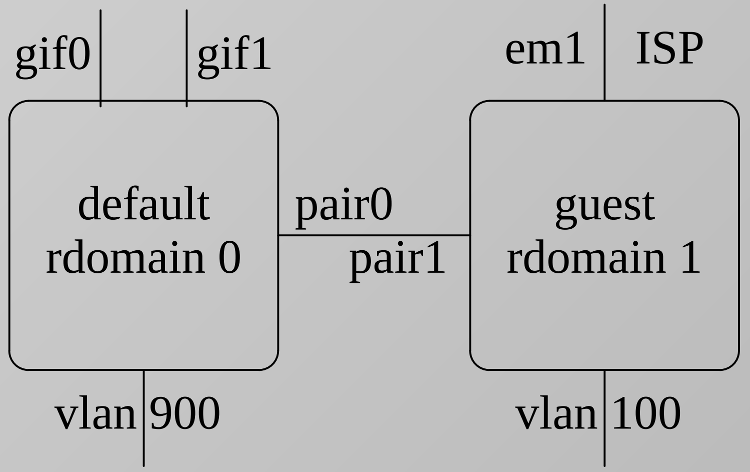
- Default route towards ISP for guest rdomain.
- Default route via pair0 for default rdomain.
- NAT on interfaces pair0 and em1
- gif interface: tunnel in guest rdomain!




```
gif1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1420
        index 17 priority 0 llprio 3
        groups: gif
        tunnel: inet 212.3.192.94 -> 194.106.45.42 rdomain 1
        inet 192.0.2.20 --> 192.0.2.19 netmask 0xffffffff
```

Network Setup - IPv6

- No RFC1918 addresses
- Own prefixes for default rdomain
- ISP provided prefixes for guest rdomain
- Prefix rewrite on pair0 with pf (nat-to with bitmask)



Current state:
IPv6 activated for guest network where local ISP provides IPv6.

Network Setup - QOS

For some locations available bandwith is very limited.

- Prefer VoIP
- Limit guests
- Limit not critical flows consuming lot of bandwidth.

```
# Upstream
queue em1 on em1 bandwidth {{ upspeed_max }}
queue em1_voice   parent em1 bandwidth 10M, min {{ voice_min }}
queue em1_ipsec   parent em1 bandwidth 10M, min {{ ipsec_min }}
queue em1_guest   parent em1 bandwidth 10M, max {{ guest_max }}
queue em1_default parent em1 bandwidth 10M, max {{ upspeed_max }} default
# Link RD5 to RD0
queue pair5 on pair5 bandwidth {{ downspeed_max }}
queue pair5_apple   parent pair5 bandwidth 10M, max {{ apple_max }}
queue pair5_voice   parent pair5 bandwidth 10M, min {{ voice_min }}
queue pair5_default parent pair5 bandwidth 10M, max {{ downspeed_max }} default
# GuestNetwork
queue vl320 on vlan320 bandwidth {{ downspeed_max }}
queue vl320_default parent vl320 bandwidth 10M, max {{ guest_max }} default
```

Future: use "flow queue" (aka FQ-CoDel)

Network Setup - QOS

```
   1 users Load 2.05 2.05 2.02                         gw-bn75.netcetera.c 13:31:25

QUEUE                   BW SCH   PR   PKTS BYTES DROP_P DROP_B QLEN BORR SUSP P/S  B/S
em1 on em1             25M               0     0      0      0    0             0    0
 em1_voice             10M            209K   75M      0      0    0             0    0
 em1_ipsec             10M               0     0      0      0    0             0    0
 em1_guest             10M             98M   80G  10150  9919K    0            11 3340
 em1_default           10M            299M   92G 113546    99M    0           255  35K
pair5 on pair5         25M               0     0      0      0    0             0    0
 pair5_apple           10M           7079K    9G  41745 60950K    0             0    0
 pair5_voice           10M               0     0      0      0    0             0    0
 pair5_default         10M            138M  167G      0      0    0           325 398K
vl500 on vlan500       25M               0     0      0      0    0             0    0
 vl500_voice           10M               0     0      0      0    0             0    0
 vl500_default         10M            305M  332G     73  29131    0           247 319K
vl720 on vlan720       25M               0     0      0      0    0             0    0
 vl720_voice           10M               0     0      0      0    0             0    0
 vl720_default         10M             57M   61G  24694 34791K    0            91  81K
gif8 on gif8           25M               0     0      0      0    0             0    0
 gif8_voice            10M             31M   12G      0      0    0           0.2   11
 gif8_default          10M            183M   51G 117026   108M    0            65  11K
gif9 on gif9           25M               0     0      0      0    0             0    0
 gif9_voice            10M              22  3344      0      0    0             0    0
 gif9_default          10M            403K   41M      0      0    0             0    0
```

Tooling - vnstat

```
 em1                                                                     22:56
 ^                                r
 |                                r
 |                                r           r
 |                                r           r
 |                                r           r
 |                             r  r  r        r  r
 |                             r  r  r        r  r
 |                             r  r  r  r     r  r
 |                             r  r  r  r  r  r  r     r
 |                          r  r  r  r  r  r  r  r  r  r  r  r
-+---------------------------------------------------------------------------->
 |  23 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22

 h  rx (MiB)   tx (MiB)       h  rx (MiB)   tx (MiB)       h  rx (MiB)   tx (MiB)
23    153.69     105.40      07    360.61      58.23      15    360.35      83.82
00     14.23      17.54      08   1189.40     110.85      16    557.27     108.61
01     14.46      18.21      09   2140.63     181.72      17    217.17      63.57
02     85.25     102.78      10   1154.31     169.26      18    247.94      33.84
03     14.58      18.19      11    836.10     174.68      19    163.70     107.93
04     16.43      19.14      12    562.35     102.52      20     55.45      31.21
05    137.91      95.93      13   1772.91     159.54      21     52.63      30.51
06     52.97      25.06      14   1184.61     106.89      22     46.43      27.34
```

Automation

Branch routers are completely configured by Ansible:

```
---
ansible_host:                    "213.13.2.90"
dhcrelay:                        "172.28.74.10"

downspeed_max:                   "25M"
upspeed_max:                     "25M"
ipsec_min:                       "15M"
guest_max:                       "10M"
voice_min:                       "10M"

if_lo1_ip:                       "172.18.0.13/32"
if_em1_ip:                       "213.13.2.90/29"
if_vlan320_ip:                   "172.19.48.1/24"
if_vlan331_ip:                   "172.19.62.1/24"
if_vlan500_ip:                   "172.19.94.1/24"

name_servers:
   - '10.11.11.11'
   - '10.22.22.22'
```

Ansible - Network interfaces

- Ansible uses a template and creates /etc/hostname.if.
- Ansible executes "sh /etc/netstart if".

Benefits:	reboot safe config easy to write Ansible tasks and templates
Drawback:	revealed bugs (carp) ans shortcomings (ospfd)

Ansible - Tunnel configuration

```
# IP addresses for tunnels 192.168.1.0 - 192.168.3.255
# -> max. 1024 IP’s, max. 512 Tunnels
tun4:                 "192.168."
tun6:                 "2001:db8:fff::"

# VPN Concentrator (HQ) must be mentioned first !
magictunnel:
  0:
    peer1: rock
    peer2: gw-br47
    key: "{{ vault_ipsec_key_0 }}"
  1:
    peer1: roll
    peer2: gw-br47
    key: "{{ vault_ipsec_key_1 }}"
```

```
- name: configure gif for SPOKE
  template:
    src=hostname.gif.spoke.j2
    dest=/etc/hostname.gif{{ item.key }}
    owner=root
    group=wheel
    mode=0640
  with_dict: ’{{ magictunnel }}’
  when: "{{ inventory_hostname_short == item.value.peer2|lower }}"
  register: gif_task_spoke
  notify: activate interface config spoke
```

OS Upgrade

- Upgrade with bsd.rd not an option (no remote access to console).

- Ansible playbook for upgrading
  - Delete old binpatches
  - Copy install sets to target
  - Copy script which performs the actual upgrade
  - Copy script for cleanups after the upgrade

- Login and execute upgrade script (does reboot the box).
- Wait 2 min and login again. Execute cleanup script.

Local Originating UDP Traffic

UDP traffic originating local on the VPN gateway might be sent out on the wrong interface after a route change.

Example:

- Internet connection down
- OSPF adjacency down, right via tunnel gone
- DHCP requests now forwarding using the default route.
- Route via tunnel learned again: traffic does not shift back to tunnel.

Affected services:

- syslog
- ntp
- netflow
- dhcp relay

Workarounds

Syslog

Use TCP instead of UDP.

Other services

A script monitors the routing socket and act upon route changes (simplified):

```
route -n monitor | while read -r l ; do
        if [[ $l == *RTM_ADD* ]] ; then
                rcctl restart ntpd
                rcctl restart $(rcctl ls on | grep dhcrelay)
                sh /etc/netstart pflow0
        fi
done
```

Could we use an tool similar to ifstated but for route messages?

Filesystem check

OpenBSD does "fsck -p" during boot if needed.

But we have no physical access to remote routers!

```
- name: fix fsck at reboot
  replace:
     dest=/etc/rc
     regexp='fsck -p'
     replace='fsck -y
```

Hardware

PC Engines APU2

Gives us a bit more than 100Mb/s with AES128-GCM (measured with OpenBSD 5.9)

HP DL360

Used for OpenBSD in the data center when not a VM. Because it’s the company standard. OpenBSD runs fine on that hardware.

Conclusion

- This setup is now in production for 8 branch offices.
- It’s not rocket sience - just a combination of available toold.
- We are now fast with rolling out a new office.
- So far we are happy with it!

Questions?