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 being 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 lose 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.

![Diagram showing network setup with BGP]
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) rlabel "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.

```conf
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 rd domains 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 bandwidth 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

<table>
<thead>
<tr>
<th>QUEUE</th>
<th>BW</th>
<th>SCH</th>
<th>PR</th>
<th>PKTS</th>
<th>BYTES</th>
<th>DROP_P</th>
<th>DROP_B</th>
<th>QLEN</th>
<th>BORR</th>
<th>SUSP</th>
<th>P/S</th>
<th>B/S</th>
</tr>
</thead>
<tbody>
<tr>
<td>eml on eml</td>
<td>25M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>eml_voice</td>
<td>10M</td>
<td>209K</td>
<td>75M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>eml_ipsec</td>
<td>10M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>eml_guest</td>
<td>10M</td>
<td>98M</td>
<td>80G</td>
<td>10150</td>
<td>9919K</td>
<td>0</td>
<td>11</td>
<td>3340</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eml_default</td>
<td>10M</td>
<td>299M</td>
<td>92G</td>
<td>113546</td>
<td>99M</td>
<td>0</td>
<td>255</td>
<td>35K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pair5 on pair5</td>
<td>25M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pair5_apple</td>
<td>10M</td>
<td>7079K</td>
<td>9G</td>
<td>41745</td>
<td>60950K</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pair5_voice</td>
<td>10M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pair5_default</td>
<td>10M</td>
<td>138M</td>
<td>167G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>325</td>
<td>398K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vl500 on vlan500</td>
<td>25M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vl500_voice</td>
<td>10M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vl500_default</td>
<td>10M</td>
<td>305M</td>
<td>332G</td>
<td>73</td>
<td>29131</td>
<td>0</td>
<td>247</td>
<td>319K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vl720 on vlan720</td>
<td>25M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vl720_voice</td>
<td>10M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vl720_default</td>
<td>10M</td>
<td>57M</td>
<td>61G</td>
<td>24694</td>
<td>34791K</td>
<td>0</td>
<td>91</td>
<td>81K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gif8 on gif8</td>
<td>25M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gif8_voice</td>
<td>10M</td>
<td>31M</td>
<td>12G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gif8_default</td>
<td>10M</td>
<td>183M</td>
<td>51G</td>
<td>117026</td>
<td>108M</td>
<td>0</td>
<td>65</td>
<td>11K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gif9 on gif9</td>
<td>25M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gif9_voice</td>
<td>10M</td>
<td>22</td>
<td>3344</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gif9_default</td>
<td>10M</td>
<td>403K</td>
<td>41M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Tooling - vnstat

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

```yaml
---
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!
magic_tunnel:
  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: '{{ magic_tunnel }}'
  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):

```bash
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?
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 science - just a combination of available tool.
• We are now fast with rolling out a new office.
• So far we are happy with it!
Questions?