VyOS Documentation

Release crux

VyOS maintainers and contributors

Sep 19, 2019

Contents:

1 VyOS History **3**

2 Installation **5**
2.1 Requirements . . . 5
2.2 Getting the software . . . 5
2.3 Preparing software verification . . . 5
2.4 GPG verification . . . 7
2.5 Install . . . 7

3 Command-Line Interface **11**

4 Quick Start Guide **13**
4.1 Configure DHCP Server and DNS . . . 13
4.2 NAT and Firewall . . . 14
4.3 Basic QoS . . . 15
4.4 Security Hardening . . . 16

5 Configuration Overview **17**
5.1 Configuration terminology . . . 18
5.2 Navigating in Configuration Mode . . . 19
5.3 Managing the configuration . . . 20
5.4 Operational info from config mode . . . 22
5.5 Configuration archive . . . 23
5.6 Local archive and revisions . . . 23
5.7 Configuring the archive size . . . 23
5.8 Remote archive . . . 24
5.9 Wipe config and restore default . . . 24

6 Network Interfaces **25**
6.1 Interface Addresses . . . 26
6.2 Dummy Interfaces . . . 28
6.3 Ethernet Interfaces . . . 28
6.4 L2TPv3 Interfaces . . . 29
6.5 PPPoE . . . 31
6.6 Wireless Interfaces . . . 33
6.7 Bridging . . . 34
6.8 Bonding . . . 35

6.9 Tunnel Interfaces . . . 36
6.10 VLAN Sub-Interfaces (802.1Q) . . . 39
6.11 QinQ . . . 40
6.12 VXLAN . . . 41

7 Routing **47**
7.1 Address Resolution Protocol (ARP) . . . 47
7.2 Border Gateway Protocol (BGP) . . . 48
7.3 Open Shortest Path First (OSPF) . . . 50
7.4 Policy-Based Routing (PBR) . . . 52
7.5 Routing Information Protocol (RIP) . . . 54
7.6 Static . . . 54
7.7 TCP-MSS Clamping . . . 55
7.8 Routing-policy . . . 55
7.9 IGMP Proxy . . . 56

8 Firewall **59**
8.1 Zone-based Firewall Policy . . . 59
8.2 Groups . . . 59
8.3 Rule-Sets . . . 60
8.4 Applying a Rule-Set to an Interface . . . 60
8.5 Applying a Rule-Set to a Zone . . . 60
8.6 How VyOS replies when being pinged . . . 60
8.7 Example Partial Config . . . 61

9 NAT **63**
9.1 Source NAT . . . 63
9.2 Destination NAT . . . 65
9.3 1-to-1 NAT . . . 66
9.4 1-to-1 NAT example . . . 67
9.5 NPTv6 (RFC6296) . . . 67
9.6 NAT before VPN . . . 68

10 VPN **71**
10.1 OpenVPN . . . 71
10.2 L2TP over IPsec . . . 77
10.3 Site-to-Site . . . 79
10.4 GRE/IPsec . . . 85
10.5 DMVPN . . . 88
10.6 PPTP-Server . . . 95
10.7 WireGuard VPN Interface . . . 96

11 QoS and Traffic Policy **99**
11.1 Configuration nodes . . . 99
11.2 Traffic policies in VyOS . . . 101
11.3 Ingress shaping . . . 118
11.4 Classful policies and traffic matching . . . 118

12 Services **123**
12.1 Conntrack . . . 123
12.2 DHCP Server . . . 126
12.3 DHCPv6 server . . . 129
12.4 DHCP Relay . . . 133
12.5 DNS Forwarding . . . 136
12.6 Dynamic DNS . . . 137

12.7 LLDP . . . 139
12.8 mDNS Repeater . . . 141
12.9 IPoE server . . . 141
12.10 PPPoE server . . . 143
12.11 SSTP server . . . 146
12.12 UDP broadcast relay . . . 148
12.13 SNMP . . . 149
12.14 SSH . . . 152
12.15 TFTP . . . 155
12.16 Webproxy . . . 155

13 System **159**
13.1 Event Handler . . . 159
13.2 Flow Accounting . . . 160
13.3 Host Information . . . 160
13.4 Serial console . . . 163
13.5 Login . . . 163
13.6 Syslog . . . 165
13.7 Task scheduler . . . 167
13.8 Config Management . . . 168
13.9 Time Zone . . . 169

14 High availability **171**
14.1 Basic setup . . . 171
14.2 IPv6 support . . . 172
14.3 Disabling a VRRP group . . . 172
14.4 Setting VRRP group priority . . . 172
14.5 Preemption . . . 172
14.6 Unicast VRRP . . . 172
14.7 Scripting . . . 173

15 Clustering **175**
15.1 General cluster configuration . . . 175
15.2 Cluster group configuration . . . 176
15.3 Review cluster status . . . 176
15.4 Example . . . 176

16 WAN load balancing **177**
16.1 Balancing Rules . . . 177
16.2 Health checks . . . 179
16.3 Source NAT rules . . . 180
16.4 Sticky Connections . . . 180
16.5 Failover . . . 180
16.6 Script execution . . . 180
16.7 Handling and monitoring . . . 181

17 System Image Management **183**
17.1 Update VyOS Installation . . . 184

18 Command scripting **187**
18.1 Run configuration commands . . . 187
18.2 Run operational commands . . . 187
18.3 Other script language . . . 188
18.4 Executing Configuration Scripts . . . 188
18.5 Postconfig on boot . . . 189

19 Release notes **191**
19.1 1.2 (Crux) . . . 191
19.2 Earlier releases . . . 194

20 Troubleshooting **195**
20.1 Basic Connectivity Verification . . . 195
20.2 Monitoring . . . 196
20.3 Clear Command . . . 199
20.4 Basic System Information . . . 200

21 Configuration Examples **203**
21.1 VyOS DMVPN Hub . . . 203
21.2 Zone-Policy example . . . 205
21.3 VyOS BGP ipv6 unnumbered with extended nexthop . . . 211
21.4 VyOS OSPF unnumbered with ecmp . . . 213
21.5 Route-Based Site-to-Site VPN to Azure (BGP over IKEv2/IPsec) . . . 215
21.6 Route-Based Redundant Site-to-Site VPN to Azure (BGP over IKEv2/IPsec) . . . 217
21.7 VyOS Tunnelbroker.net IPv6 . . . 219

22 Command tree **223**
22.1 Operational mode . . . 223
22.2 Configuration mode . . . 230

23 Running on Bare Metal **239**
23.1 Intel Atom C3000 . . . 239
23.2 PC Engines APU4 . . . 240
23.3 Qotom Q355G4 . . . 253
23.4 Partaker i5 . . . 253
23.5 Acrosser AND-J190N1 . . . 254

24 Migrate from Vyatta Core **257**
24.1 Vyatta release compatiblity . . . 257
24.2 Upgrade procedure . . . 257

25 Building VyOS using Docker **261**

26 Issues and Feature requests **263**
26.1 Bug Report / Issue . . . 263
26.2 Feature Request . . . 264

27 Development **265**
27.1 Submit a patch . . . 265

28 VyOS CLI **269**
28.1 Example XML File . . . 269
28.2 Configuration mode command definitions . . . 271
28.3 Mapping old node.def style to new XML definitions . . . 273

29 Python Coding Guidelines **275**
29.1 Configuration script structure and behaviour . . . 275
29.2 Coding guidelines . . . 277
29.3 Code policy . . . 277

30 Upstream packages **279**
30.1 vyos-netplug . . . 279
30.2 keepalived . . . 279

30.3 strongswan . . . 279
30.4 ppp . . . 280
30.5 mdns-repeater . . . 280
30.6 udp-broadcast-relay . . . 280
30.7 Linux kernel . . . 280
30.8 Linux firmware . . . 281
30.9 Intel drivers . . . 281
30.10 accel-ppp . . . 281
30.11 hvinfo . . . 281
30.12 Per-file modifications . . . 281

VyOS is an open source network operating system based on Debian GNU/Linux.

VyOS provides a free routing platform that competes directly with other commercially available solutions from well known network providers. Because VyOS is run on standard amd64, i586 and ARM systems, it is able to be used as a router and firewall platform for cloud deployments.

CHAPTER 1

VyOS History

VyOS is a Linux-based network operating system that provides software-based network routing, firewall, and VPN functionality.

The VyOS project was started in late 2013 as a community fork of the GPL portions of Vyatta Core 6.6R1 with the goal of maintaining a free and open source network operating system in response to the decision to discontinue the community edition of Vyatta. Here everyone loves learning, older managers and new users.

VyOS is primarily based on Debian GNU/Linux and the Quagga routing engine. Its configuration syntax and *Command-Line Interface* are loosely derived from Juniper JUNOS as modelled by the XORP project (which was the original routing engine Vyatta was based upon).

Vyatta changed to the Quagga routing engine for release 4.0.

VyOS changed the routing engine to FRRouting in version 1.2.0

How it's different from other router distros?

- Unified command line interface in the style of hardware routers.
- Scriptable CLI
- Stateful configuration system: prepare changes and commit at once or discard, view previous revisions or roll-back to them, archive revisions to remote server, execute hooks at commit time…
- Image-based upgrade: keep multiple versions on the same system and revert to previous image if something went wrong.
- Not just firewall and VPN: includes routing protocols such as BGP and OSPF and complex routing policy language.
- Runs on physical and virtual platforms alike: small x86 boards, big servers, KVM, Xen, VMWare, Hyper-V…
- Completely free and open source, with documented internal APIs and build procedures.
- Community driven: patches are welcome, all code, bugs, and nightly builds are public.

CHAPTER 2

Installation

2.1 Requirements

The recommended system requirements are 512 MiB RAM and 2 GiB storage.

2.2 Getting the software

Registered subscribers can download stable release (LTS) images. If you log into https://support.vyos.io/ as a registered subscriber, you will see the "Downloads" link there.

Non-subscribers can get the LTS release by building it from source. Just follow the instructions in https://github.com/vyos/vyos-build

Everyone can dowload VyOS rolling images from https://downloads.vyos.io/

2.3 Preparing software verification

This subsection and the following one applies to dowloaded LTS images, for other cases please jump to *Install*.

LTS images are signed by VyOS lead package-maintainer private key. If you have our public key, you will be able to verify the authenticity of the package.

First you need to install GPG or another OpenPGP implementation. On most GNU+Linux distributions it is installed by default because package managers use it to verify package signatures. On other systems you may need to find and install the package.

Then you need VyOS public key. If you already have it, you can jump to *GPG verification*. If you don't have it yet, you can get it from a key server.

```
gpg --recv-keys FD220285A0FE6D7E
```

Or open your web browser, visit a key server and copy our public key from there: https://pgp.mit.edu/pks/lookup?op=get&search=0xFD220285A0FE6D7E

Or from this block below.


```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.12 (GNU/Linux)

mQINBFXKsiIBEACyid9PR/v56pSRG8VgQyRwvzoI7rLErZ8BCQA2WFxA6+zNy+6G
+0E/6XAOzE+VHli+wtJpiVJwAh+wWuqzOmv9css2fdJxpMW87pJAS2i3EVVVf6ab
wU848JYLGzc9y7gZrnT1m2fNh4MXkZBNDp780WpOZx8roZq5X+j+Y5hk5KcLiBn/
lh9Zoh8yzrWDSXQsz0BGoAbVnLUEWyo0tcRcHuC0eLx6oNG/IHvd/+kxWB1uULHU
SlB/6vcx56lLqgzywkmhP01050ZDyTqrFRIfrvw6gLQaWlgR3lB93txvF/sz87Il
VblV7e6HEyVUQxedDS8ikOyzdb5r9a6Zt/j8ZPSntFNM6OcKAI7U1nDD3FVOhlVn
7lhUiNc+/qjC+pR9CrZjr/BTWE7Zpi6/kzeH4eAkfjyALj18oC5udJDjXE5daTL3
k9difHf74VkZm29Cy9M3zPckOZpsGiBl8YQsf+RXSBMDVYRKZ1BNNLDofm4ZGijK
mriXcaY+VIeVB26J8m8y0zN4/ZdioJXRcy72c1KusRt8e/TsqtC9UFK05YpzRm5R
/nwxDFYb7EdY/vHUFOmfwXLaRvyZtRJ9LwvRUAqgRbbRZg3ET/tn6JZk8hqx3e1M
IxuskOB19t5vWyAo/TLGIFw44SErrq9jnpqgclTSRgFjcjHEm061r4vjoQARAQAB
tDZWeU9TIE1haW50YWluZXJzIChWeU9TIFJlbGVhc2UpIDxtYWludGFpbmVyc0B2
eW9zLm5ldD6JAjgEEwECACIFAlXKsiICGwMGCwkIBwMCBhUIAgkKCwQWAgMBAh4B
AheAAAoJEP0iAoWg/m1+xbgP+QEDYZi5dA4IPY+vU1L95Bavju2m2o35TSUDPg5B
jfAGuhbsNUceU+l/yUlxjpKEmvshyW3GHR5QzUaKGup/ZDBo1CBxZNhpSlFida2E
KAYTx4vHk3MRXcntiAj/hIJwRtzCUp5UQIqHoU8dmHoHOkKEP+zhJuR6E2s+WwDr
nTwE6eRa0g/AHY+chj2Je6flpPm2CKoTfUE7a2yBBU3wPq3rGtsQgVxPAxHRZz7A
w4AjH3NM1Uo3etuiDnGkJAuoKKb1J4X3w2QlbwlR4cODLKhJXHIufwaGtRwEin9S
1l2bL8V3gy2Hv3D2t9TQZuR5NUHsibJRXLSa8WnSCcc6Bij5aqfdpYB+YvKH/rIm
GvYPmLZDfKGkx0JE4/qtfFjiPJ5VE7BxNyliEw/rnQsxWAGPqLlL61SD8w5jGkw3
CinwO3sccTVcPz9b6A1RsbBVhTJJX5lcPn1lkOEVwQ7l8bRhOKCMe0P53qEDcLCd
KcXNnAFbVes9u+kfUQ4oxS0G2JS9ISVNmune+uv+JR7KqSdOuRYlyXA9uTjgWz4y
Cs7RS+CpkJFqrqOtS1rmuDW9Ea4PA8ygGlisM5d/AlVkniHz/2JYtgetiLCj9mfE
MzQpgnldNSPumKqJ3wwmCNisE+lXQ5UXCaoaeqF/qX1ykybQn41LQ+0xT5Uvy7sL
9IwGuQINBFXKsiIBEACg2mP3QYkXdgWTK5JyTGyttE6bDC9uqsK8dc1J66Tjd5Ly
Be0amO+88GHXa0o5Smwk2QNoxsRR41G/D/eAeGsuOEYnePROEr3tcLnDjo4KLgQ+
H69zRPn77sdP3A34Jgp+QIzByJWM7Cnim31quQP3qal2QdpGJcT/jDJWdticN76a
Biaz+HN13LyvZM+DWhUDttbjAJc+TEwF9YzIrU+3AzkTRDWkRh4kNIQxjlpNzvho
9V75riVqg2vtgPwttPEhOLb0oMzy4ADdfezrfVvvMb4M4kY9npu4MlSkNTM97F/I
QKy90JuSUIjE05AO+PDXJF4Fd5dcpmukLV/2nV0WM2LAERpJUuAgkZN6pNUFVISR
+nSfgR7wvqeDY9NigHrJqJbSEgaBUs6RTk5hait2wnNKLJajlu3aQ2/QfRT/kG3h
ClKUz3Ju7NCURmFE6mfsdsVrlIsEjHr/dPbXRswXgC9FLlXpWgAEDYi9Wdxxz8o9
JDWrVYdKRGG+OpLFh8AP6QL3YnZF+p1oxGUQ5ugXauAJ9YS55pbzaUFP8oOO2P1Q
BeYnKRs1GcMI8KWtE/fze9C9gZ7Dqju7ZFEyllM4v3lzjhT8muMSAhw41J22mSx6
VRkQVRIAvPDFES45IbB6EEGhDDg4pD2az8Q7i7Uc6/olEmpVONSOZEEPsQe/2wAR
AQABiQIfBBgBAgAJBQJVyrIiAhsMAAoJEP0iAoWg/m1+niUQAKTxwJ9PTAfB+XDk
3qH3n+T49O2wP3fhBI0EGhJp9Xbx29G7qfEeqcQm69/qSq2/0HQOc+w/g8yy71jA
6rPuozCraoN7Im09rQ2NqIhPK/1w5ZvgNVC0NtcMigX9MiSARePKygAHOPHtrhyO
rJQyu8E3cV3VRT4qhqIqXs8Ydc9vL3ZrJbhcHQuSLdZxM1k+DahCJgwWabDCUizm
sVP3epAP19FP8sNtHi0P1LC0kq6/0qJot+4iBiRwXMervCD5ExdOm2ugvSgghdYN
BikFHvmsCxbZAQjykQ6TMn+vkmcEz4fGAn4L7Nx4paKEtXaAFO8TJmFjOlGUthEm
CtHDKjCTh9WV4pwG2WnXuACjnJcs6LcK377EjWU25H4y1ff+NDIUg/DWfSS85iIc
UgkOlQO6HJy0O96L5uxn7VJpXNYFa20lpfTVZv7uu3BC3RW/FyOYsGtSiUKYq6cb
CMxGTfFxGeynwIlPRlH68BqH6ctR/mVdo+5UIWsChSnNd1GreIEI6p2nBk3mc7jZ
7pTEHpjarwOjs/S/lK+vLW53CSFimmW4lw3MwqiyAkxl0tHAT7QMHH9Rgw2HF/g6
XD76fpFdMT856dsuf+j2uuJFlFe5B1fERBzeU18MxML0VpDmGFEaxxypfACeI/iu
8vzPzaWHhkOkU8/J/Ci7+vNtUOZb
=Ld8S
-----END PGP PUBLIC KEY BLOCK-----
```


Then you can paste that text in a new file, and import the file into GPG:

```
gpg --import file_with_the_public_key
```

You can now check your GPG software has our public key.

```
$ gpg --list-keys
...
pub   rsa4096 2015-08-12 [SC]
    0694A9230F5139BF834BA458FD220285A0FE6D7E
uid           [ unknown] VyOS Maintainers (VyOS Release) <maintainers@vyos.net>
sub   rsa4096 2015-08-12 [E]
```

2.4 GPG verification

As you have our public key, you just need the signature of the software you want to verify.

Note: In order to get the signature, go to your web browser and append .asc to the URL of your dowloaded VyOS image. You will download a small *.asc* file, that's the signature of your image.

So finally you can verify the authenticity of your image.

```
$ gpg2 --verify vyos-1.2.1-amd64.iso.asc  vyos-1.2.1-amd64.iso
gpg: Signature made So 14 Apr 12:58:07 2019 CEST
gpg:                using RSA key FD220285A0FE6D7E
gpg: Good signature from "VyOS Maintainers (VyOS Release) <maintainers@vyos.net>"␣
↪[unknown]
Primary key fingerprint: 0694 A923 0F51 39BF 834B  A458 FD22 0285 A0FE 6D7E
```

2.5 Install

The VyOS ISO is a Live CD and will boot to a functional VyOS image. To login to the system, use the default username `vyos` with password `vyos`.

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
vyos@vyos:~$

vyos@vyos:~$ uname -a
Linux vyos 4.18.11-amd64-vyos #23 SMP Mon Oct 1 17:29:22 CEST 2018 x86_64 GNU/Linux
```

Unlike general purpose Linux distributions, VyOS uses "image installation" that mimics the user experience of traditional hardware routers and allows you to keep multiple VyOS versions on the same machine and switch to a previous version if something breaks after upgrade. Every version is contained in its own squashfs image that is mounted in a union filesystem together with a directory for mutable data (configs etc.).

Note: Older versions used to support non-image installation (*install system* command). Support for this is removed from VyOS 1.2 (crux) and newer releases

This installation method has been deprecated since the time image installation was introduced (long before the fork), and does not provide any version management capabilities. You **should not** use it for new installations even if it's still available in new versions. You should not worry about older systems installed that way though, they can be upgraded with `add system image`.

To install VyOS, run `install image`.

```
vyos@vyos:~$ install image
Welcome to the VyOS install program.  This script
will walk you through the process of installing the
VyOS image to a local hard drive.
Would you like to continue? (Yes/No) [Yes]: Yes
Probing drives: OK
Looking for pre-existing RAID groups...none found.
The VyOS image will require a minimum 2000MB root.
Would you like me to try to partition a drive automatically
or would you rather partition it manually with parted?  If
you have already setup your partitions, you may skip this step

Partition (Auto/Parted/Skip) [Auto]:

I found the following drives on your system:
 sda    4294MB

Install the image on? [sda]:

This will destroy all data on /dev/sda.
Continue? (Yes/No) [No]: Yes

How big of a root partition should I create? (2000MB - 4294MB) [4294]MB:

Creating filesystem on /dev/sda1: OK
Done!
Mounting /dev/sda1...
What would you like to name this image? [1.2.0-rolling+201809210337]:
OK.  This image will be named: 1.2.0-rolling+201809210337
Copying squashfs image...
Copying kernel and initrd images...
Done!
I found the following configuration files:
    /opt/vyatta/etc/config.boot.default
Which one should I copy to sda? [/opt/vyatta/etc/config.boot.default]:

Copying /opt/vyatta/etc/config.boot.default to sda.
Enter password for administrator account
Enter password for user 'vyos':
Retype password for user 'vyos':
I need to install the GRUB boot loader.
I found the following drives on your system:
 sda    4294MB

Which drive should GRUB modify the boot partition on? [sda]:

Setting up grub: OK
Done!
vyos@vyos:~$
```

After the installation is complete, remove the Live CD and reboot the system:

```
vyos@vyos:~$ reboot
Proceed with reboot? (Yes/No) [No] Yes
```

CHAPTER 3

Command-Line Interface

The VyOS CLI comprises an *Operational mode* and a *Configuration mode*.

Operational mode allows for commands to perform operational system tasks and view system and service status, while configuration mode allows for the modification of system configuration. The *command tree page* lists available commands and their functions.

The CLI provides a built-in help system. In the CLI the **[?]** key may be used to display available commands. The **[tab]** key can be used to auto-complete commands and will present the help system upon a conflict or unknown value.

For example typing *sh* followed by the **[tab]** key will complete to *show*. Pressing **[tab]** a second time will display the possible sub-commands of the *show* command.

```
vyos@vyos:~$ s[tab]
set   show
vyos@vyos:~$
```

Example showing possible show commands:

```
vyos@vyos:~$ show [tab]
Possible completions:
  arp           Show Address Resolution Protocol (ARP) information
  bridge        Show bridging information
  cluster       Show clustering information
  configuration Show running configuration
  conntrack     Show conntrack entries in the conntrack table
  conntrack-sync
                Show connection syncing information
  date          Show system date and time
  dhcp          Show Dynamic Host Configuration Protocol (DHCP) information
  dhcpv6        Show status related to DHCPv6
  disk          Show status of disk device
  dns           Show Domain Name Server (DNS) information
  file          Show files for a particular image
  firewall      Show firewall information
  flow-accounting
```

(continues on next page)

(continued from previous page)

```
                Show flow accounting statistics
  hardware      Show system hardware details
  history       show command history
  host          Show host information
  incoming      Show ethernet input-policy information
: q
vyos@vyos:~$
```

You can scroll up with the keys [Shift]+[PageUp] and sroll down with [Shift]+[PageDown].

When the output of a command results in more lines than can be displayed on the terminal screen the output is paginated as indicated by a : prompt.

When viewing in page mode the following commands are available:

- **[q]** key can be used to cancel output
- **[space]** will scroll down one page
- **[b]** will scroll back one page
- **[return]** will scroll down one line
- **[up-arrow]** and **[down-arrow]** will scroll up or down one line at a time respectively
- **[left-arrow]** and **[right-arrow]** can be used to scroll left or right in the event that the output has lines which exceed the terminal size.

To enter configuration mode use the *configure* command:

```
vyos@vyos:~$ configure
[edit]
vyos@vyos:~#
```

Note: Prompt changes from *$* to *#*. To exit configuration mode, type *exit*.

```
vyos@vyos:~# exit
exit
vyos@vyos:~$
```

See the configuration section of this document for more information on configuration mode.

CHAPTER 4

Quick Start Guide

Below is a very basic configuration example that will provide a NAT gateway for a device with two interfaces.

Enter configuration mode:

```
vyos@vyos$ configure
vyos@vyos#
```

Configure network interfaces:

```
set interfaces ethernet eth0 address dhcp
set interfaces ethernet eth0 description 'OUTSIDE'
set interfaces ethernet eth1 address '192.168.0.1/24'
set interfaces ethernet eth1 description 'INSIDE'
```

Enable SSH for remote management:

```
set service ssh port '22'
```

4.1 Configure DHCP Server and DNS

```
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 default-router
↪'192.168.0.1'
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 dns-server '192.
↪168.0.1'
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 domain-name
↪'internal-network'
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 lease '86400'
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 range 0 start␣
↪192.168.0.9
set service dhcp-server shared-network-name LAN subnet 192.168.0.0/24 range 0 stop
↪'192.168.0.254'
```

And a DNS forwarder:

```
set service dns forwarding cache-size '0'
set service dns forwarding listen-address '192.168.0.1'
set service dns forwarding name-server '8.8.8.8'
set service dns forwarding name-server '8.8.4.4'
```

4.2 NAT and Firewall

Configure Source NAT for our "Inside" network.

```
set nat source rule 100 outbound-interface 'eth0'
set nat source rule 100 source address '192.168.0.0/24'
set nat source rule 100 translation address masquerade
```

Add a set of firewall policies for our "Outside" interface.

This configuration creates a proper stateful firewall that blocks all traffic:

```
set firewall name OUTSIDE-IN default-action 'drop'
set firewall name OUTSIDE-IN rule 10 action 'accept'
set firewall name OUTSIDE-IN rule 10 state established 'enable'
set firewall name OUTSIDE-IN rule 10 state related 'enable'
set firewall name OUTSIDE-LOCAL default-action 'drop'
set firewall name OUTSIDE-LOCAL rule 10 action 'accept'
set firewall name OUTSIDE-LOCAL rule 10 state established 'enable'
set firewall name OUTSIDE-LOCAL rule 10 state related 'enable'
set firewall name OUTSIDE-LOCAL rule 20 action 'accept'
set firewall name OUTSIDE-LOCAL rule 20 icmp type-name 'echo-request'
set firewall name OUTSIDE-LOCAL rule 20 protocol 'icmp'
set firewall name OUTSIDE-LOCAL rule 20 state new 'enable'
```

If you wanted to enable SSH access to your firewall from the the Internet, you could create some additional rules to allow the traffic.

These rules allow SSH traffic and rate limit it to 4 requests per minute. This blocks brute-forcing attempts:

```
set firewall name OUTSIDE-LOCAL rule 30 action 'drop'
set firewall name OUTSIDE-LOCAL rule 30 destination port '22'
set firewall name OUTSIDE-LOCAL rule 30 protocol 'tcp'
set firewall name OUTSIDE-LOCAL rule 30 recent count '4'
set firewall name OUTSIDE-LOCAL rule 30 recent time '60'
set firewall name OUTSIDE-LOCAL rule 30 state new 'enable'
set firewall name OUTSIDE-LOCAL rule 31 action 'accept'
set firewall name OUTSIDE-LOCAL rule 31 destination port '22'
set firewall name OUTSIDE-LOCAL rule 31 protocol 'tcp'
set firewall name OUTSIDE-LOCAL rule 31 state new 'enable'
```

Apply the firewall policies:

```
set interfaces ethernet eth0 firewall in name 'OUTSIDE-IN'
set interfaces ethernet eth0 firewall local name 'OUTSIDE-LOCAL'
```

Commit changes, save the configuration, and exit configuration mode:

```
vyos@vyos# commit
vyos@vyos# save
Saving configuration to '/config/config.boot'...
Done
vyos@vyos# exit
vyos@vyos$
```

4.3 Basic QoS

The traffic policy subsystem provides an interface to Linux traffic control (tc).

One common use of traffic policy is to limit bandwidth for an interface. In the example below we limit bandwidth for our LAN connection to 200 Mbit download and out WAN connection to 50 Mbit upload:

```
set traffic-policy shaper WAN-OUT bandwidth '50Mbit'
set traffic-policy shaper WAN-OUT default bandwidth '50%'
set traffic-policy shaper WAN-OUT default ceiling '100%'
set traffic-policy shaper WAN-OUT default queue-type 'fair-queue'
set traffic-policy shaper LAN-OUT bandwidth '200Mbit'
set traffic-policy shaper LAN-OUT default bandwidth '50%'
set traffic-policy shaper LAN-OUT default ceiling '100%'
set traffic-policy shaper LAN-OUT default queue-type 'fair-queue'
```

Resulting in the following configuration:

```
traffic-policy {
    shaper WAN-OUT {
        bandwidth 50Mbit
        default {
            bandwidth 50%
            ceiling 100%
            queue-type fair-queue
        }
    }
    shaper LAN-OUT {
        bandwidth 200Mbit
        default {
            bandwidth 50%
            ceiling 100%
            queue-type fair-queue
        }
    }
}
```

Once defined, a traffic policy can be applied to each interface using the interface-level traffic-policy directive:

```
set interfaces ethernet eth0 traffic-policy out 'WAN-OUT'
set interfaces ethernet eth1 traffic-policy out 'LAN-OUT'
```

Note: A traffic policy can also be defined to match specific traffic flows using class statements.

VyOS 1.2 (Crux) also supports HFSC (`set traffic-policy shaper-hfsc`)

See further information in the *QoS and Traffic Policy* chapter.

4.4 Security Hardening

Especially if you are allowing SSH access from the Internet, there are a few additional configuration steps that should be taken.

Create a user to replace the default *vyos* user:

```
set system login user myvyosuser level admin
set system login user myvyosuser authentication plaintext-password mysecurepassword
```

Set up SSH key based authentication. For example, on Linux you'd want to run *ssh-keygen -t rsa*. Then the contents of *id_rsa.pub* would be used below:

```
set system login user myvyosuser authentication public-keys myusername@mydesktop type␣
↪ssh-rsa
set system login user myvyosuser authentication public-keys myusername@mydesktop key␣
↪contents_of_id_rsa.pub
```

Or you can use the *loadkey* command. Commit and save.

Finally, try and ssh into the VyOS install as your new user.

Once you have confirmed that your new user can access your server, without a password, delete the original *vyos* user and disable password authentication into SSH:

```
delete system login user vyos
set service ssh disable-password-authentication
```

Commit and save.

CHAPTER 5

Configuration Overview

VyOS makes use of a unified configuration file for all system configuration: *config.boot*. This allows for easy template creation, backup, and replication of system configuration.

The current active configuration -aka running configuration- can be viewed using the show configuration command.

```
vyos@vyos:~$ show configuration
interfaces {
    ethernet eth0 {
        address dhcp
        hw-id 00:0c:29:44:3b:0f
    }
    loopback lo {
    }
}
service {
    ssh {
        port 22
    }
}
system {
    config-management {
        commit-revisions 20
    }
    console {
        device ttyS0 {
            speed 9600
        }
    }
    login {
        user vyos {
            authentication {
                encrypted-password ****************
            }
            level admin
        }
```

(continues on next page)

(continued from previous page)

```
    }
    ntp {
        server 0.pool.ntp.org {
        }
        server 1.pool.ntp.org {
        }
        server 2.pool.ntp.org {
        }
    }
    syslog {
        global {
            facility all {
                level notice
            }
            facility protocols {
                level debug
            }
        }
    }
}
vyos@vyos:~$
```

By default the configuration is displayed in a hierarchy like the example above, this is only one of the possible ways to display the configuration.

When the configuration is generated and the device is configured, changes are added through a collection of *set* and *delete* commands. You can see that collection of commands by entering *show configuration commands*, which is another way of seeing the running configuration.

```
vyos@vyos:~$ show configuration commands
set interfaces ethernet eth0 address 'dhcp'
set interfaces ethernet eth0 hw-id '00:0c:29:44:3b:0f'
set interfaces loopback 'lo'
set service ssh port '22'
set system config-management commit-revisions '20'
set system console device ttyS0 speed '9600'
set system login user vyos authentication encrypted-password '<removed>'
set system login user vyos level 'admin'
set system ntp server '0.pool.ntp.org'
set system ntp server '1.pool.ntp.org'
set system ntp server '2.pool.ntp.org'
set system syslog global facility all level 'notice'
set system syslog global facility protocols level 'debug'
vyos@vyos:~$
```

Both these commands should be executed when in operational mode, they do not work in configuration mode.

5.1 Configuration terminology

A VyOS system has three major types of configurations:

5.1.1 Active or running configuration

The active or running configuration is the system configuration that is loaded and currently being used by VyOS. Any change in the configuration will have to be committed to belong to the active/running configuration.

5.1.2 Working configuration

The working configuration is the configuration which is currently being modified in configuration mode. Changes made to the working configuration do not go into effect until the changes are committed with the *commit* command. At which time the working configuration will become the active or running configuration.

5.1.3 Saved configuration

A saved configuration is a configuration saved to a file using the *save* command. It allowes you to keep safe a configuration for future uses. There can be multiple configuration files. The default or "boot" configuration is saved and loaded from the file config.boot.

5.2 Navigating in Configuration Mode

When entering the configuration mode you are navigating inside the tree structure exported in the overview above, to enter configuration mode enter the command *configure* when in operational mode

```
vyos@vyos$ configure
[edit]
vyos@vyos#
```

Note: When going into configuration mode, prompt changes from *$* to *#*. To exit configuration mode, type *exit*.

All commands executed here are relative to the configuration level you have entered. You can do everything from the top level, but commands will be quite lengthy when manually typing them.

To change the current hierarchy level use the command: *edit*

```
[edit]
vyos@vyos# edit interfaces ethernet eth0

[edit interfaces ethernet eth0]
vyos@vyos#
```

You are now in a sublevel relative to *interfaces ethernet eth0*, all commands executed from this point on are relative to this sublevel. Use either the *top* or *exit* command to go back to the top of the hierarchy. You can also use the *up* command to move only one level up at a time.

The *show* command within configuration mode will show the working configuration indicating line changes with + for additions, > for replacements and - for deletions.

```
vyos@vyos:~$ configure
[edit]
vyos@vyos# show interfaces
 ethernet eth0 {
     description MY_OLD_DESCRIPTION
```

(continues on next page)

(continued from previous page)

```
     disable
     hw-id 52:54:00:0e:82:d9
 }
 loopback lo {
 }
[edit]
vyos@vyos# set interfaces ethernet eth0 address dhcp
[edit]
vyos@vyos# set interfaces ethernet eth0 description MY_NEW_DESCRIPTION
[edit]
vyos@vyos# delete interfaces ethernet eth0 disable
[edit]
vyos@vyos# show interfaces
 ethernet eth0 {
+    address dhcp
>    description MY_NEW_DESCRIPTION
-    disable
     hw-id 52:54:00:0e:82:d9
 }
 loopback lo {
 }
[edit]
vyos@vyos#
```

It is also possible to display all *set* commands within configuration mode using *show | commands*

```
vyos@vyos# show interfaces ethernet eth0 | commands
set address dhcp
set hw-id 00:0c:29:44:3b:0f
```

These commands are also relative to the level you are inside and only relevant configuration blocks will be displayed when entering a sub-level.

```
[edit interfaces ethernet eth0]
vyos@vyos# show
 address dhcp
 hw-id 00:0c:29:44:3b:0f
```

Exiting from the configuration mode is done via the *exit* command from the top level, executing *exit* from within a sub-level takes you back to the top level.

```
[edit interfaces ethernet eth0]
vyos@vyos# exit
[edit]
vyos@vyos# exit
Warning: configuration changes have not been saved.
vyos@vyos:~$
```

5.3 Managing the configuration

The configuration is managed by the use of *set* and *delete* commands from within configuration mode. Configuration commands are flattend from the tree into 'one-liner' commands shown in *show configuration commands* from operation mode.

These commands are also relative to the level where they are executed and all redundant information from the current level is removed from the command entered.

```
[edit]
vyos@vyos# set interface ethernet eth0 address 203.0.113.6/24

[edit interfaces ethernet eth0]
vyos@vyos# set address 203.0.113.6/24
```

These two commands above are essentially the same, just executed from different levels in the hierarchy.

To delete a configuration entry use the *delete* command, this also deletes all sub-levels under the current level you've specified in the *delete* command. Deleting an entry could also mean to reset it back to its default value if the element is mandatory, in each case it will be removed from the configuration file.

```
[edit interfaces ethernet eth0]
vyos@vyos#  delete address 203.0.113.6/24
```

Any change you do on the configuration, will not take effect until committed using the *commit* command in configuration mode.

```
vyos@vyos# commit
[edit]
vyos@vyos# exit
Warning: configuration changes have not been saved.
vyos@vyos:~$
```

In order to preserve configuration changes upon reboot, the configuration must also be saved once applied. This is done using the *save* command in configuration mode.

```
vyos@vyos# save
Saving configuration to '/config/config.boot'...
Done
[edit]
vyos@vyos#
```

Configuration mode can not be exited while uncommitted changes exist. To exit configuration mode without applying changes, the exit discard command can be used.

```
vyos@vyos# exit
Cannot exit: configuration modified.
Use 'exit discard' to discard the changes and exit.
[edit]
vyos@vyos# exit discard
exit
vyos@vyos:~$
```

VyOS automatically maintains backups of previous configurations. To compare configuration revisions in configuration mode, use the compare command:

```
vyos@vyos# compare [tab]
Possible completions:
  <Enter>     Compare working & active configurations
  saved               Compare working & saved configurations
  <N>         Compare working with revision N
  <N> <M>     Compare revision N with M
  Revisions:
    0    2013-12-17 20:01:37 root by boot-config-loader
```

(continues on next page)

(continued from previous page)

```
    1    2013-12-13 15:59:31 root by boot-config-loader
    2    2013-12-12 21:56:22 vyos by cli
    3    2013-12-12 21:55:11 vyos by cli
    4    2013-12-12 21:27:54 vyos by cli
    5    2013-12-12 21:23:29 vyos by cli
    6    2013-12-12 21:13:59 root by boot-config-loader
    7    2013-12-12 16:25:19 vyos by cli
    8    2013-12-12 15:44:36 vyos by cli
    9    2013-12-12 15:42:07 root by boot-config-loader
    10   2013-12-12 15:42:06 root by init

[edit]
vyos@vyos#
```

You can rollback configuration using the rollback command. This command will apply the selected revision and trigger a system reboot.

```
vyos@vyos# compare 1
[edit system]
>host-name vyos-1
[edit]
vyos@vyos# rollback 1
Proceed with reboot? [confirm][y]
Broadcast message from root@vyos-1 (pts/0) (Tue Dec 17 21:07:45 2013):
The system is going down for reboot NOW!
[edit]
vyos@vyos#
```

VyOS also supports saving and loading configuration remotely using SCP, FTP, or TFTP.

```
vyos@vyos# save [tab]
Possible completions:
  <Enter>       Save to system config file
  <file>        Save to file on local machine
  scp://<user>:<passwd>@<host>/<file> Save to file on remote machine
  ftp://<user>:<passwd>@<host>/<file> Save to file on remote machine
  tftp://<host>/<file>      Save to file on remote machine
vyos@vyos# save tftp://192.168.0.100/vyos-test.config.boot
Saving configuration to 'tftp://192.168.0.100/vyos-test.config.boot'...
##################################################################### 100.0%
Done
```

5.4 Operational info from config mode

When inside configuration mode you are not directly able to execute operational commands.

Access to these commands are possible through the use of the *run [command]* command. From this command you will have access to everything accessible from operational mode. Command completion and syntax help with *?* and *[tab]* will also work.

```
[edit]
vyos@vyos# run show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
```

(continues on next page)

(continued from previous page)

```
---------        ----------                        ---  -----------
eth0             0.0.0.0/0                         u/u
```

5.5 Configuration archive

VyOS has built-in config archiving and versioning that renders tools like rancid, largely unnecessary.

This feature was available in Vyatta Core since 6.3

5.6 Local archive and revisions

Revisions are stored on disk. You can view them, compare them, and rollback to previous revisions if anything goes wrong.

To view existing revisions, use *show system commit* operational mode command.

```
vyos@vyos-test-2# run show system commit
0   2015-03-30 08:53:03 by vyos via cli
1   2015-03-30 08:52:20 by vyos via cli
2   2015-03-26 21:26:01 by root via boot-config-loader
3   2015-03-26 20:43:18 by root via boot-config-loader
4   2015-03-25 11:06:14 by root via boot-config-loader
5   2015-03-25 01:04:28 by root via boot-config-loader
6   2015-03-25 00:16:47 by vyos via cli
7   2015-03-24 23:43:45 by root via boot-config-loader
```

You can compare revisions with *compare X Y* command, where X and Y are revision numbers. The output will describe how the configuration X is when comparted to Y, indicating with a plus sign (**+**) the additional parts X has when compared to y, and indicating with a minus sign (**-**) the lacking parts x misses when compared to y.

```
vyos@vyos-test-2# compare 0 6
[edit interfaces]
+dummy dum1 {
+    address 10.189.0.1/31
+}
[edit interfaces ethernet eth0]
+vif 99 {
+    address 10.199.0.1/31
+}
-vif 900 {
-    address 192.0.2.4/24
-}
```

You can rollback to a previous revision with *rollback X*, where X is a revision number. Your system will reboot and load the config from the archive.

5.7 Configuring the archive size

You can specify the number of revisions stored on disk with *set system config-management commit-revisions X*, where X is a number between 0 and 65535. When the number of revisions exceeds that number, the oldest revision is removed.

5.8 Remote archive

VyOS can copy the config to a remote location after each commit. TFTP, FTP, and SFTP servers are supported.

You can specify the location with:

- *set system config-management commit-archive location URL*

For example, *set system config-management commit-archive location tftp://10.0.0.1/vyos*.

You can specify the location with *set system config-management commit-archive location URL* command, e.g. *set system config-management commit-archive location tftp://10.0.0.1/vyos*.

5.9 Wipe config and restore default

In the case you want to completely delete your configuration and restore the default one, you can enter the following command in configuration mode:

```
load /opt/vyatta/etc/config.boot.default
```

Note: If you are remotely connected, you will lose your connection. You may want to copy first the config, edit it to ensure connectivity, and load the edited config.

CHAPTER 6

Network Interfaces

Configured interfaces on a VyOS system can be displayed using the *show interfaces* command.

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             172.16.51.129/24                  u/u  OUTSIDE
eth1             192.168.0.1/24                    u/u  INSIDE
lo               127.0.0.1/8                       u/u
                 ::1/128
vyos@vyos:~$
```

A specific interface can be shown using the *show interfaces* <*type*> <*name*> command.

```
vyos@vyos:~$ show interfaces ethernet eth0
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:44:3b:0f brd ff:ff:ff:ff:ff:ff
    inet 172.16.51.129/24 brd 172.16.51.255 scope global eth0
    inet6 fe80::20c:29ff:fe44:3b0f/64 scope link
       valid_lft forever preferred_lft forever
    Description: OUTSIDE

    RX:  bytes    packets     errors    dropped    overrun      mcast
        274397       3064          0          0          0          0
    TX:  bytes    packets     errors    dropped    carrier collisions
        257276       1890          0          0          0          0
vyos@vyos:~$
```

Different network interfaces provide type-specific configuration. Ethernet interfaces, for example, allow the configuration of speed and duplex.

Many services, such as network routing, firewall, and traffic policy also maintain interface-specific configuration. These will be covered in their respective sections.

6.1 Interface Addresses

Each interface can be configured with a description and address. Interface addresses might be:

- Static IPv4 *address 172.16.51.129/24*
- Static IPv6 *address 2001:db8:1::ffff/64*
- DHCP IPv4 *address dhcp*
- DHCP IPv6 *address dhcpv6*

An interface description is assigned using the following command:

```
set interfaces ethernet eth0 description 'OUTSIDE'
```

6.1.1 IPv4

Static Address

This method is supported on all interfaces, apart from OpenVPN that uses different syntax and wireless modems that are always autoconfigured through PPP.

The command is *set interfaces $type $name address $address*. Examples:

```
set interfaces ethernet eth0 address 192.0.2.1/24
set interfaces tunnel tun0 address 10.0.0.1/30
set interfaces bridge br0 address 203.0.113.45/26
set interfaces ethernet eth0 vif 30 address 192.0.30.254/24
```

DHCP

This method is supported on all physical interfaces, and those that are directly connected to a physical interface (ethernet, VLAN, bridge, bond, pseudo-ethernet, wireless).

The command is *set interfaces $type $name address dhcp*. Examples:

```
set interfaces ethernet eth0 vif 90 address dhcp
set interfaces bridge br0 address dhcp
```

6.1.2 IPv6

Static Address

This method is supported on all interfaces, apart from OpenVPN that uses different syntax and wireless modems that are always autoconfigured through PPP. Static IPv6 addresses are supported on all interfaces except *Tunnel Interfaces*.

The command is *set interfaces $type $name address $address*. Examples:

```
set interfaces ethernet eth0 address 2001:db8:100::ffff/64
set interfaces tunnel tun0 address 2001:db8::1/64
set interfaces bridge br0 address  2001:db8:200::1/64
set interfaces ethernet eth0 vif 30 address 2001:db8:3::ffff/64
```

DHCP

This method is supported on all physical interfaces, and those that are directly connected to a physical interface (ethernet, VLAN, bridge, bond, pseudo-ethernet, wireless).

The command is *set interfaces $type $name address dhcpv6*. Examples:

```
set interfaces bonding bond1 address dhcpv6
set interfaces bridge br0 vif 56 address dhcpv6
```

Autoconfiguration (SLAAC)

SLAAC is specified in RFC4862. This method is supported on all physical interfaces, and those that are directly connected to a physical interface (ethernet, VLAN, bridge, bond, pseudo-ethernet, wireless).

The command is *set interfaces $type $name ipv6 address autoconf*. Examples:

```
set interfaces ethernet eth0 vif 90 ipv6 address autoconf
set interfaces bridge br0 ipv6 address autoconf
```

Note: This method automatically disables IPv6 traffic forwarding on the interface in question.

EUI-64

EUI-64 (64-Bit Extended Unique Identifier) as specified in RFC4291. IPv6 addresses in /64 networks can be automatically generated from the prefix and MAC address, if you specify the prefix.

The command is *set interfaces $type $name ipv6 address eui64 $prefix*. Examples:

```
set interfaces bridge br0 ipv6 address eui64 2001:db8:beef::/64
set interfaces pseudo-ethernet peth0 ipv6 address eui64 2001:db8:aa::/64
```

Router Advertisements

Router advertisements are described in RFC4861 section 4.2. They are part of what is known as SLAAC (Stateless Address Autoconfiguration).

To enable or disable, use:

```
set interfaces <interface> ipv6 router-advert send-advert <true or false>
```

To set the options described in "Router Advertisement Message Format":

```
vyos@vyos#  set interfaces <interface> ipv6 router-advert
Possible completions:
  cur-hop-limit         Value to be placed in the "Current Hop Limit" field in RAs
  default-lifetime      Value to be placed in "Router Lifetime" field in RAs
  default-preference    Default router preference
  link-mtu              Value of link MTU to place in RAs
  managed-flag          Value for "managed address configuration" flag in RAs
  max-interval          Maximum interval between unsolicited multicast RAs
  min-interval          Minimum interval between unsolicited multicast RAs
```

(continues on next page)

(continued from previous page)

```
+  name-server          IPv6 address of a Recursive DNS Server
  other-config-flag     Value to be placed in the "other configuration" flag in RAs
+> prefix                IPv6 prefix to be advertised in Router Advertisements (RAs)
  reachable-time        Value to be placed in "Reachable Time" field in RAs
  retrans-timer         Value to place in "Retrans Timer" field in RAs.
  send-advert           Enable/disable sending RAs
```

Prefix Information

Prefix information is described in RFC4861 section 4.6.2

```
vyos@vyos# set interfaces <interface> ipv6 router-advert prefix <h:h:h:h:h:h:h:h/x>
Possible completions:
  autonomous-flag       Whether prefix can be used for address auto-configuration
  on-link-flag          Flag that prefix can be used for on-link determination
  preferred-lifetime    Time in seconds that the prefix will remain preferred
  valid-lifetime        Time in seconds that the prefix will remain valid
```

Receiving Router Advertisements

To receive and accept RAs on an interface, you need to enable it with the following configuration command

```
vyos@vyos# set system sysctl custom net.ipv6.conf.<interface name>.accept_ra value 2
```

6.2 Dummy Interfaces

Dummy interfaces — much like the loopback, except you can have as many as you want. Dummy interfaces can be used as interfaces that always stay up (in the same fashion to loopbacks in IOS), or for testing purposes.

Configuration commands:

```
interfaces
    dummy <dum[0-999]>
    + address           IP address
    description         Description
    disable             Disable interface
    > ip                IPv4 routing parameters
    > ipv6              IPv6 routing parameters
    redirect            Incoming packet redirection destination
    > traffic-policy    Traffic-policy for interface
```

6.3 Ethernet Interfaces

Ethernet interfaces allow for the configuration of speed, duplex, and hw-id (MAC address). Below is an example configuration:

```
set interfaces ethernet eth1 address '192.168.0.1/24'
set interfaces ethernet eth1 address '2001:db8:1::ffff/64'
set interfaces ethernet eth1 description 'INSIDE'
set interfaces ethernet eth1 duplex 'auto'
set interfaces ethernet eth1 speed 'auto'
```

Resulting in:

```
ethernet eth1 {
    address 192.168.0.1/24
    address 2001:db8:1::ffff/64
    description INSIDE
    duplex auto
    hw-id 00:0c:29:44:3b:19
    smp_affinity auto
    speed auto
}
```

In addition, Ethernet interfaces provide the extended operational commands:

- *show interfaces ethernet <name> physical*
- *show interfaces ethernet <name> statistics*

Statistics available are driver dependent.

```
vyos@vyos:~$ show interfaces ethernet eth0 physical
Settings for eth0:
        Supported ports: [ TP ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Speed: 1000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: on
        MDI-X: Unknown
        Supports Wake-on: d
        Wake-on: d
        Current message level: 0x00000007 (7)
        Link detected: yes
driver: e1000
version: 7.3.21-k8-NAPI
firmware-version:
bus-info: 0000:02:01.0

vyos@vyos:~$ show interfaces ethernet eth0 statistics
NIC statistics:
     rx_packets: 3530
     tx_packets: 2179
[...]
```

6.4 L2TPv3 Interfaces

L2TPv3 is a pseudowire protocol, you can read more about here Wikipedia L2TPv3 or RFC3921

L2TPv3 can transport any traffic including ethernet frames. L2TPv2 is limited to PPP.

6.4.1 L2TPv3 over IP

```
# show interfaces l2tpv3
l2tpv3 l2tpeth10 {
    address 192.168.37.1/27
    encapsulation ip
    local-ip 192.0.2.1
    peer-session-id 100
    peer-tunnel-id 200
    remote-ip 203.0.113.24
    session-id 100
    tunnel-id 200
}
```

Inverse configuration has to be applied to the remote side.

6.4.2 L2TPv3 over UDP

UDP mode works better with NAT:

- Set local-ip to your local IP (LAN).
- Add a forwarding rule matching UDP port on your internet router.

```
# show interfaces l2tpv3
l2tpv3 l2tpeth10 {
    address 192.168.37.1/27
    destination-port 9001
    encapsulation udp
    local-ip 192.0.2.1
    peer-session-id 100
    peer-tunnel-id 200
    remote-ip 203.0.113.24
    session-id 100
    source-port 9000
    tunnel-id 200
}
```

To create more than one tunnel, use distinct UDP ports.

6.4.3 L2TPv3 over IPSec, L2 VPN (bridge)

This is the LAN extension use case. The eth0 port of the distant VPN peers will be directly connected like if there was a switch between them.

IPSec:

```
set vpn ipsec ipsec-interfaces <VPN-interface>
set vpn ipsec esp-group test-ESP-1 compression 'disable'
set vpn ipsec esp-group test-ESP-1 lifetime '3600'
set vpn ipsec esp-group test-ESP-1 mode 'transport'
set vpn ipsec esp-group test-ESP-1 pfs 'enable'
set vpn ipsec esp-group test-ESP-1 proposal 1 encryption 'aes128'
set vpn ipsec esp-group test-ESP-1 proposal 1 hash 'sha1'
set vpn ipsec ike-group test-IKE-1 ikev2-reauth 'no'
set vpn ipsec ike-group test-IKE-1 key-exchange 'ikev1'
```

(continues on next page)

(continued from previous page)

```
set vpn ipsec ike-group test-IKE-1 lifetime '3600'
set vpn ipsec ike-group test-IKE-1 proposal 1 dh-group '5'
set vpn ipsec ike-group test-IKE-1 proposal 1 encryption 'aes128'
set vpn ipsec ike-group test-IKE-1 proposal 1 hash 'sha1'
set vpn ipsec site-to-site peer <peer-ip> authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer <peer-ip> authentication pre-shared-secret <pre-
↪shared-key>
set vpn ipsec site-to-site peer <peer-ip> connection-type 'initiate'
set vpn ipsec site-to-site peer <peer-ip> ike-group 'test-IKE-1'
set vpn ipsec site-to-site peer <peer-ip> ikev2-reauth 'inherit'
set vpn ipsec site-to-site peer <peer-ip> local-address <local-ip>
set vpn ipsec site-to-site peer <peer-ip> tunnel 1 allow-nat-networks 'disable'
set vpn ipsec site-to-site peer <peer-ip> tunnel 1 allow-public-networks 'disable'
set vpn ipsec site-to-site peer <peer-ip> tunnel 1 esp-group 'test-ESP-1'
set vpn ipsec site-to-site peer <peer-ip> tunnel 1 protocol 'l2tp'
```

Bridge:

```
set interfaces bridge br0 description 'L2 VPN Bridge'
# remote side in this example:
# set interfaces bridge br0 address '172.16.30.18/30'
set interfaces bridge br0 address '172.16.30.17/30'
set interfaces ethernet eth0 bridge-group bridge 'br0'
set interfaces ethernet eth0 description 'L2 VPN Physical port'
```

L2TPv3:

```
set interfaces l2tpv3 l2tpeth0 bridge-group bridge 'br0'
set interfaces l2tpv3 l2tpeth0 description 'L2 VPN Tunnel'
set interfaces l2tpv3 l2tpeth0 destination-port '5000'
set interfaces l2tpv3 l2tpeth0 encapsulation 'ip'
set interfaces l2tpv3 l2tpeth0 local-ip <local-ip>
set interfaces l2tpv3 l2tpeth0 mtu '1500'
set interfaces l2tpv3 l2tpeth0 peer-session-id '110'
set interfaces l2tpv3 l2tpeth0 peer-tunnel-id '10'
set interfaces l2tpv3 l2tpeth0 remote-ip <peer-ip>
set interfaces l2tpv3 l2tpeth0 session-id '110'
set interfaces l2tpv3 l2tpeth0 source-port '5000'
set interfaces l2tpv3 l2tpeth0 tunnel-id '10'
```

6.5 PPPoE

There are two main ways to setup VyOS to connect over a PPPoE internet connection. This is due to most ISPs (Internet Service Providers) providing a DSL modem that is also a wireless router.

First Method: (Common for Homes)

In this method, the DSL Modem/Router connects to the ISP for you with your credentials preprogrammed into the device. This gives you an RFC1918 address, such as 192.168.1.0/24 by default.

For a simple home network using just the ISP's equipment, this is usually desirable. But if you want to run VyOS as your firewall and router, this will result in having a double NAT and firewall setup. This results in a few extra layers of complexity, particularly if you use some NAT or tunnel features.

Second Method: (Common for Businesses)

In order to have full control and make use of multiple static public IP addresses, your VyOS will have to initiate the PPPoE connection and control it. In order for this method to work, you will have to figure out how to make your DSL Modem/Router switch into a Bridged Mode so it only acts as a DSL Transceiver device to connect between the Ethernet link of your VyOS and the phone cable. Once your DSL Transceiver is in Bridge Mode, you should get no IP address from it. Please make sure you connect to the Ethernet Port 1 if your DSL Transeiver has a switch, as some of them only work this way. Once you have an Ethernet device connected, i.e. eth0, then you can configure it to open the PPPoE session for you and your DSL Transceiver (Modem/Router) just acts to translate your messages in a way that vDSL/aDSL understands.

Here is an example configuration:

```
set interface ethernet eth0 description "DSL Modem"
set interface ethernet eth0 duplex auto
set interface ethernet eth0 smp_affinity auto
set interface ethernet eth0 speed auto
set interface ethernet eth0 pppoe 0 default-route auto
set interface ethernet eth0 pppoe 0 mtu 1492
set interface ethernet eth0 pppoe 0 name-server auto
set interface ethernet eth0 pppoe 0 user-id <PPPoE Username>
set interface ethernet eth0 pppoe 0 password <PPPoE Password>
```

- You should add a firewall to your configuration above as well by assigning it to the pppoe0 itself as shown here:

```
set interface ethernet eth0 pppoe 0 firewall in name NET-IN
set interface ethernet eth0 pppoe 0 firewall local name NET-LOCAL
set interface ethernet eth0 pppoe 0 firewall out name NET-OUT
```

- You need your PPPoE credentials from your DSL ISP in order to configure this. The usual username is in the form of name@host.net but may vary depending on ISP.
- The largest MTU size you can use with DSL is 1492 due to PPPoE overhead. If you are switching from a DHCP based ISP like cable then be aware that things like VPN links may need to have their MTU sizes adjusted to work within this limit.
- With the `default-route` option set to `auto`, VyOS will only add the Default Gateway you receive from your DSL ISP to the routing table if you have no other WAN connections. If you wish to use a Dual WAN connection, change the `default-route` option to `force`.

6.5.1 Handling and troubleshooting

You can test connecting and disconnecting with the below commands:

```
disconnect interface 0
connect interface 0
```

You can check the PPPoE connection logs with the following:

This command shows the current statistics, status and some of the settings (i.e. MTU) for the current connection on pppoe0.

```
show interfaces pppoe 0
```

This command shows the entire log for the PPPoE connection starting with the oldest data. Scroll down with the <space> key to reach the end where the current data is.

```
show interfaces pppoe 0 log
```

This command shows the same log as without the 'tail' option but only starts with the last few lines and continues to show added lines until you exit with `Ctrl + x`

```
show interfaces pppoe 0 log tail
```

6.6 Wireless Interfaces

Wireless, for example WiFi 802.11 b/g/n, interfaces allow for connection to WiFi networks or act as an access-point. If your device is configurable it will appear as *wlan* in *show interfaces*.

To be able to use the wireless interfaces you will first need to set a regulatory domain with the country code of your locaion.

```
set system wifi-regulatory-domain SE
```

An example on how to set it up as an access point:

```
set interfaces wireless wlan0 address '192.168.99.1/24'
set interfaces wireless wlan0 type access-point
set interfaces wireless wlan0 channel 1
set interfaces wireless wlan0 ssid '<your ssid>'
set interfaces wireless wlan0 security wpa mode wpa2
set interfaces wireless wlan0 security wpa cipher CCMP
set interfaces wireless wlan0 security wpa passphrase '<your passphrase>'
```

Resulting in

```
interfaces {
  [...]
  wireless wlan0 {
        address 192.168.99.1/24
        channel 1
        mode g
        security {
            wpa {
                cipher CCMP
                mode wpa2
                passphrase "<your passphrase>"
            }
        }
        ssid "<your ssid>"
        type access-point
    }
}
system {
  [...]
  wifi-regulatory-domain SE
}
```

To get it to work as a access point with this configuration you will need to set up a DHCP server to work with that network.

6.7 Bridging

Interfaces in VyOS can be bridged together to provide software switching of Layer-2 traffic.

A bridge is created when a bridge interface is defined. In the example below we will be creating a bridge for VLAN 100 and assigning a VIF to the bridge.

```
set interfaces bridge 'br100'
set interfaces ethernet eth1 vif 100 bridge-group bridge br100
```

Interfaces assigned to a bridge-group do not have address configuration. An IP address can be assigned to the bridge interface itself, however, like any normal interface.

```
set interfaces bridge br100 address '192.168.100.1/24'
set interfaces bridge br100 address '2001:db8:100::1/64'
```

Example Result:

```
bridge br100 {
    address 192.168.100.1/24
    address 2001:db8:100::1/64
}
[...]
ethernet eth1 {
[...]
    vif 100 {
        bridge-group {
            bridge br100
        }
    }
}
```

In addition to normal IP interface configuration, bridge interfaces support Spanning-Tree Protocol. STP is disabled by default.

Note: Please use caution when introducing spanning-tree protocol on a network as it may result in topology changes.

To enable spanning-tree use the *set interfaces bridge <name> stp true* command:

```
set interfaces bridge br100 stp true
```

STP *priority*, *forwarding-delay*, *hello-time*, and *max-age* can be configured for the bridge-group. The MAC aging time can also be configured using the *aging* directive.

For member interfaces, the bridge-group *priority* and *cost* can be configured.

The *show bridge* operational command can be used to display configured bridges:

```
vyos@vyos:~$ show bridge
bridge name     bridge id               STP enabled     interfaces
br100           0000.000c29443b19       yes             eth1.100
```

If spanning-tree is enabled, the *show bridge <name> spanning-tree* command can be used to show STP configuration:

```
vyos@vyos:~$ show bridge br100 spanning-tree
br100
```

(continues on next page)

(continued from previous page)

```
 bridge id              0000.000c29443b19
 designated root        0000.000c29443b19
 root port                 0                    path cost                  0
 max age                  20.00                 bridge max age            20.00
 hello time                2.00                 bridge hello time          2.00
 forward delay            15.00                 bridge forward delay      15.00
 ageing time             300.00
 hello timer               0.47                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                  64.63
 flags

eth1.100 (1)
 port id                8001                    state                forwarding
 designated root        0000.000c29443b19       path cost                  4
 designated bridge      0000.000c29443b19       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags
```

The MAC address-table for a bridge can be displayed using the *show bridge <name> macs* command:

```
vyos@vyos:~$ show bridge br100 macs
port no mac addr                is local?       ageing timer
  1     00:0c:29:44:3b:19       yes                0.00
```

6.8 Bonding

You can combine (aggregate) 2 or more physical interfaces into a single logical one. It's called bonding, or LAG, or ether-channel, or port-channel.

Create interface bondX, where X is just a number:

```
set interfaces bonding bond0 description 'my-sw1 int 23 and 24'
```

You are able to choose a hash policy:

```
vyos@vyos# set interfaces bonding bond0 hash-policy
Possible completions:
  layer2       use MAC addresses to generate the hash (802.3ad)
  layer2+3     combine MAC address and IP address to make hash
  layer3+4     combine IP address and port to make hash
```

For example:

```
set interfaces bonding bond0 hash-policy 'layer2'
```

You may want to set IEEE 802.3ad Dynamic link aggregation (802.3ad) AKA LACP (don't forget to setup it on the other end of these links):

```
set interfaces bonding bond0 mode '802.3ad'
```

or some other modes:

```
vyos@vyos# set interfaces bonding bond0 mode
Possible completions:
  802.3ad      IEEE 802.3ad Dynamic link aggregation (Default)
  active-backup
               Fault tolerant: only one slave in the bond is active
  broadcast    Fault tolerant: transmits everything on all slave interfaces
  round-robin  Load balance: transmit packets in sequential order
  transmit-load-balance
               Load balance: adapts based on transmit load and speed
  adaptive-load-balance
               Load balance: adapts based on transmit and receive plus ARP
  xor-hash     Load balance: distribute based on MAC address
```

Now bond some physical interfaces into bond0:

```
set interfaces ethernet eth0 bond-group 'bond0'
set interfaces ethernet eth0 description 'member of bond0'
set interfaces ethernet eth1 bond-group 'bond0'
set interfaces ethernet eth1 description 'member of bond0'
```

After a commit you may treat bond0 as almost a physical interface (you can't change its' duplex, for example) and assign IPs or VIFs on it.

You may check the result:

```
vyos@vyos# run sh interfaces bonding
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
bond0            -                                 u/u  my-sw1 int 23 and 24
bond0.10         192.168.0.1/24                    u/u  office-net
bond0.100        10.10.10.1/24                     u/u  management-net
```

6.9 Tunnel Interfaces

This article touches on 'classic' IP tunneling protocols.

GRE is often seen as a one size fits all solution when it comes to classic IP tunneling protocols, and for a good reason. However, there are more specialized options, and many of them are supported by VyOS. There are also rather obscure GRE options that can be useful.

All those protocols are grouped under 'interfaces tunnel' in VyOS. Let's take a closer look at the protocols and options currently supported by VyOS.

6.9.1 IPIP

This is one of the simplest types of tunnels, as defined by RFC2003. It takes an IPv4 packet and sends it as a payload of another IPv4 packet. For this reason, there are no other configuration options for this kind of tunnel.

An example:

```
set interfaces tunnel tun0 encapsulation ipip
set interfaces tunnel tun0 local-ip 192.0.2.10
set interfaces tunnel tun0 remote-ip 203.0.113.20
set interfaces tunnel tun0 address 192.168.100.200
```

6.9.2 IP6IP6

This is the IPv6 counterpart of IPIP. I'm not aware of an RFC that defines this encapsulation specifically, but it's a natural specific case of IPv6 encapsulation mechanisms described in RFC2473.

It's not likely that anyone will need it any time soon, but it does exist.

An example:

```
set interfaces tunnel tun0 encapsulation ipip
set interfaces tunnel tun0 local-ip 2001:db8:aa::1/64
set interfaces tunnel tun0 remote-ip 2001:db8:aa::2/64
set interfaces tunnel tun0 address 2001:db8:bb::1/64
```

6.9.3 IPIP6

In the future this is expected to be a very useful protocol (though there are other proposals).

As the name implies, it's IPv4 encapsulated in IPv6, as simple as that.

An example:

```
set interfaces tunnel tun0 encapsulation ipip6
set interfaces tunnel tun0 local-ip 2001:db8:aa::1/64
set interfaces tunnel tun0 remote-ip 2001:db8:aa::2/64
set interfaces tunnel tun0 address 192.168.70.80
```

6.9.4 6in4 (SIT)

6in4 uses tunneling to encapsulate IPv6 traffic over IPv4 links as defined in RFC4213. The 6in4 traffic is sent over IPv4 inside IPv4 packets whose IP headers have the IP protocol number set to 41. This protocol number is specifically designated for IPv6 encapsulation, the IPv4 packet header is immediately followed by the IPv6 packet being carried. The encapsulation overhead is the size of the IPv4 header of 20 bytes, therefore with an MTU of 1500 bytes, IPv6 packets of 1480 bytes can be sent without fragmentation. This tunneling technique is frequently used by IPv6 tunnel brokers like Hurricane Electric.

An example:

```
set interfaces tunnel tun0 encapsulation sit
set interfaces tunnel tun0 local-ip 192.0.2.10
set interfaces tunnel tun0 remote-ip 192.0.2.20
set interfaces tunnel tun0 address 2001:db8:bb::1/64
```

A full example of a Tunnelbroker.net config can be found at *here*.

6.9.5 Generic Routing Encapsulation (GRE)

A GRE tunnel operates at layer 3 of the OSI model and is repsented by IP protocol 47. The main benefit of a GRE tunnel is that you are able to route traffic across disparate networks. GRE also supports multicast traffic and supports routing protocols that leverage multicast to form neighbor adjacencies.

Configuration

A basic configuration requires a tunnel source (local-ip), a tunnel destination (remote-ip), an encapsulation type (gre), and an address (ipv4/ipv6). Below is a configuration example taken from a VyOS router and a Cisco IOS router. The main difference between these two configurations is that VyOS requires you explicitly configure the encapsulation type. The Cisco router defaults to 'gre ip' otherwise it would have to be configured as well.

VyOS Router:

```
set interfaces tunnel tun100 address '10.0.0.1/30'
set interfaces tunnel tun100 encapsulation 'gre'
set interfaces tunnel tun100 local-ip '198.51.100.2'
set interfaces tunnel tun100 remote-ip '203.0.113.10'
```

Cisco IOS Router:

```
interface Tunnel100
ip address 10.0.0.2 255.255.255.252
tunnel source 203.0.113.10
tunnel destination 198.51.100.2
```

Troubleshooting

GRE is a well defined standard that is common in most networks. While not inherently difficult to configure there are a couple of things to keep in mind to make sure the configuration performs as expected. A common cause for GRE tunnels to fail to come up correctly include ACL or Firewall configurations that are discarding IP protocol 47 or blocking your source/desintation traffic.

1. Confirm IP connectivity between tunnel local-ip and remote-ip:

```
vyos@vyos:~$ ping 203.0.113.10 interface 198.51.100.2 count 4
PING 203.0.113.10 (203.0.113.10) from 198.51.100.2 : 56(84) bytes of data.
64 bytes from 203.0.113.10: icmp_seq=1 ttl=254 time=0.807 ms
64 bytes from 203.0.113.10: icmp_seq=2 ttl=254 time=1.50 ms
64 bytes from 203.0.113.10: icmp_seq=3 ttl=254 time=0.624 ms
64 bytes from 203.0.113.10: icmp_seq=4 ttl=254 time=1.41 ms

--- 203.0.113.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 0.624/1.087/1.509/0.381 ms
```

2. Confirm the link type has been set to GRE:

```
vyos@vyos:~$ show interfaces tunnel tun100
tun100@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue state UNKNOWN␣
↪group default qlen 1000
  link/gre 198.51.100.2 peer 203.0.113.10
  inet 10.0.0.1/30 brd 10.0.0.3 scope global tun100
     valid_lft forever preferred_lft forever
  inet6 fe80::5efe:c612:2/64 scope link
     valid_lft forever preferred_lft forever

  RX:  bytes    packets     errors    dropped    overrun      mcast
        2183         27          0          0          0          0
  TX:  bytes    packets     errors    dropped    carrier collisions
         836          9          0          0          0          0
```

3. Confirm IP connectivity across the tunnel:

```
vyos@vyos:~$ ping 10.0.0.2 interface 10.0.0.1 count 4
PING 10.0.0.2 (10.0.0.2) from 10.0.0.1 : 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=255 time=1.05 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=255 time=1.88 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=255 time=1.98 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=255 time=1.98 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3008ms
rtt min/avg/max/mdev = 1.055/1.729/1.989/0.395 ms
```

6.9.6 Virtual Tunnel Interface (VTI)

Set Virtual Tunnel Interface

```
set interfaces vti vti0 address 192.168.2.249/30
set interfaces vti vti0 address 2001:db8:2::249/64
```

Results in:

```
vyos@vyos# show interfaces vti
vti vti0 {
    address 192.168.2.249/30
    address 2001:db8:2::249/64
    description "Description"
}
```

6.10 VLAN Sub-Interfaces (802.1Q)

802.1Q VLAN interfaces are represented as virtual sub-interfaces in VyOS. The term used for this is *vif*. Configuration of a tagged sub-interface is accomplished using the configuration command *set interfaces ethernet <name> vif <vlan-id>*.

```
set interfaces ethernet eth1 vif 100 description 'VLAN 100'
set interfaces ethernet eth1 vif 100 address '192.168.100.1/24'
set interfaces ethernet eth1 vif 100 address '2001:db8:100::1/64'
```

Resulting in:

```
ethernet eth1 {
    address 192.168.100.1/24
    address 2001:db8:100::1/64
    description INSIDE
    duplex auto
    hw-id 00:0c:29:44:3b:19
    smp_affinity auto
    speed auto
    vif 100 {
        address 192.168.100.1/24
        description "VLAN 100"
    }
}
```

VLAN interfaces are shown as *<name>.<vlan-id>*, e.g. *eth1.100*:

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             172.16.51.129/24                  u/u  OUTSIDE
eth1             192.168.0.1/24                    u/u  INSIDE
eth1.100         192.168.100.1/24                  u/u  VLAN 100
lo               127.0.0.1/8                       u/u
                 ::1/128
```

6.11 QinQ

QinQ (802.1ad) — allows multiple VLAN tags to be inserted into a single frame.

QinQ can be used to tunnel vlans in a vlan.

vif-s and **vif-c** stand for the ethertype tags that get set:

The inner tag is the tag which is closest to the payload portion of the frame; it is officially called C-TAG (Customer tag, with ethertype 0x8100). The outer tag is the one closer/closest to the Ethernet header; its name is S-TAG (Service tag, ethertype 0x88a8).

Configuration commands:

```
interfaces
    ethernet <eth[0-999]>
        address <ipv4>
        address <ipv6>
        description <txt>
        disable
        ip
            <usual IP options>
        ipv6
            <usual IPv6 options>
        vif-s <[0-4096]>
            address <ipv4>
            address <ipv6>
            description <txt>
            disable
            ip
                <usual IP options>
            ipv6
                <usual IPv6 options>
            vif-c <[0-4096]>
                address <ipv4>
                address <ipv6>
                description <txt>
                disable
                ip
                    <usual IP options>
                ipv6
                    <usual IPv6 options>
```

Example:

```
set interfaces ethernet eth0 vif-s 333
set interfaces ethernet eth0 vif-s 333 address 192.0.2.10/32
set interfaces ethernet eth0 vif-s 333 vif-c 777
set interfaces ethernet eth0 vif-s 333 vif-c 777 address 10.10.10.10/24
```

6.12 VXLAN

VXLAN is an overlaying Ethernet over IP protocol. It is described in RFC7348[1].

If configuring VXLAN in a VyOS virtual machine, ensure that MAC spoofing (Hyper-V) or Forged Transmits (ESX) are permitted, otherwise forwarded frames may be blocked by the hypervisor.

6.12.1 Multicast VXLAN

Example Topology:

PC4 - Leaf2 - Spine1 - Leaf3 - PC5

PC4 has IP 10.0.0.4/24 and PC5 has IP 10.0.0.5/24, so they believe they are in the same broadcast domain.

Let's assume PC4 on Leaf2 wants to ping PC5 on Leaf3. Instead of setting Leaf3 as our remote end manually, Leaf2 encapsulates the packet into a UDP-packet and sends it to its designated multicast-address via Spine1. When Spine1 receives this packet it forwards it to all other Leafs who has joined the same multicast-group, in this case Leaf3. When Leaf3 receives the packet it forwards it, while at the same time learning that PC4 is reachable behind Leaf2, because the encapsulated packet had Leaf2's IP-address set as source IP.

PC5 receives the ping echo, responds with an echo reply that Leaf3 receives and this time forwards to Leaf2's unicast address directly because it learned the location of PC4 above. When Leaf2 receives the echo reply from PC5 it sees that it came from Leaf3 and so remembers that PC5 is reachable via Leaf3.

Thanks to this discovery, any subsequent traffic between PC4 and PC5 will not be using the multicast-address between the Leafs as they both know behind which Leaf the PCs are connected. This saves traffic as less multicast packets sent reduces the load on the network, which improves scalability when more Leafs are added.

For optimal scalability Multicast shouldn't be used at all, but instead use BGP to signal all connected devices between leafs. Unfortunately, VyOS does not yet support this.

6.12.2 Configuration commands

```
interfaces
  vxlan <vxlan[0-16777215]>
    address           # IP address of the VXLAN interface
    bridge-group      # Configure a L2 bridge-group
    description       # Description
    group <ipv4>      # IPv4 Multicast group address (required)
    ip                # IPv4 routing options
    ipv6              # IPv6 routing options
    link <dev>        # IP interface for underlay of this vxlan overlay (optional)
    mtu               # MTU
    policy            # Policy routing options
```

(continues on next page)

[1] https://datatracker.ietf.org/doc/rfc7348/

(continued from previous page)

```
    remote           # Remote address of the VXLAN tunnel, used for PTP instead of␣
↪multicast
    vni <1-16777215> # Virtual Network Identifier (required)
```

6.12.3 Configuration Example

The setup is this:

Leaf2 - Spine1 - Leaf3

Spine1 is a Cisco IOS router running version 15.4, Leaf2 and Leaf3 is each a VyOS router running 1.2.

This topology was built using GNS3.

Topology:

```
Spine1:
fa0/2 towards Leaf2, IP-address: 10.1.2.1/24
fa0/3 towards Leaf3, IP-address: 10.1.3.1/24

Leaf2:
Eth0 towards Spine1, IP-address: 10.1.2.2/24
Eth1 towards a vlan-aware switch

Leaf3:
Eth0 towards Spine1, IP-address 10.1.3.3/24
Eth1 towards a vlan-aware switch
```

Spine1 Configuration:

```
conf t
ip multicast-routing
!
interface fastethernet0/2
 ip address 10.1.2.1 255.255.255.0
 ip pim sparse-dense-mode
!
interface fastethernet0/3
 ip address 10.1.3.1 255.255.255.0
 ip pim sparse-dense-mode
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
```

Multicast-routing is required for the leafs to forward traffic between each other in a more scalable way. This also requires PIM to be enabled towards the Leafs so that the Spine can learn what multicast groups each Leaf expect traffic from.

Leaf2 configuration:

```
set interfaces ethernet eth0 address '10.1.2.2/24'
set protocols ospf area 0 network '10.0.0.0/8'

! Our first vxlan interface
set interfaces bridge br241 address '172.16.241.1/24'
set interfaces ethernet eth1 vif 241 bridge-group bridge 'br241'
```

(continues on next page)

(continued from previous page)

```
set interfaces vxlan vxlan241 bridge-group bridge 'br241'
set interfaces vxlan vxlan241 group '239.0.0.241'
set interfaces vxlan vxlan241 link 'eth0'
set interfaces vxlan vxlan241 vni '241'

! Our seconds vxlan interface
set interfaces bridge br242 address '172.16.242.1/24'
set interfaces ethernet eth1 vif 242 bridge-group bridge 'br242'
set interfaces vxlan vxlan242 bridge-group bridge 'br242'
set interfaces vxlan vxlan242 group '239.0.0.242'
set interfaces vxlan vxlan242 link 'eth0'
set interfaces vxlan vxlan242 vni '242'
```

Leaf3 configuration:

```
set interfaces ethernet eth0 address '10.1.3.3/24'
set protocols ospf area 0 network '10.0.0.0/8'

! Our first vxlan interface
set interfaces bridge br241 address '172.16.241.1/24'
set interfaces ethernet eth1 vif 241 bridge-group bridge 'br241'
set interfaces vxlan vxlan241 bridge-group bridge 'br241'
set interfaces vxlan vxlan241 group '239.0.0.241'
set interfaces vxlan vxlan241 link 'eth0'
set interfaces vxlan vxlan241 vni '241'

! Our seconds vxlan interface
set interfaces bridge br242 address '172.16.242.1/24'
set interfaces ethernet eth1 vif 242 bridge-group bridge 'br242'
set interfaces vxlan vxlan242 bridge-group bridge 'br242'
set interfaces vxlan vxlan242 group '239.0.0.242'
set interfaces vxlan vxlan242 link 'eth0'
set interfaces vxlan vxlan242 vni '242'
```

As you can see, Leaf2 and Leaf3 configuration is almost identical. There are lots of commands above, I'll try to into more detail below, command descriptions are placed under the command boxes:

```
set interfaces bridge br241 address '172.16.241.1/24'
```

This commands creates a bridge that is used to bind traffic on eth1 vlan 241 with the vxlan241-interface. The IP-address is not required. It may however be used as a default gateway for each Leaf which allows devices on the vlan to reach other subnets. This requires that the subnets are redistributed by OSPF so that the Spine will learn how to reach it. To do this you need to change the OSPF network from '10.0.0.0/8' to '0.0.0.0/0' to allow 172.16/12-networks to be advertised.

```
set interfaces ethernet eth1 vif 241 bridge-group bridge 'br241'
set interfaces vxlan vxlan241 bridge-group bridge 'br241'
```

Binds eth1 vif 241 and vxlan241 to each other by putting them in the same bridge-group. Internal VyOS requirement.

```
set interfaces vxlan vxlan241 group '239.0.0.241'
```

The multicast-group used by all Leafs for this vlan extension. Has to be the same on all Leafs that has this interface.

```
set interfaces vxlan vxlan241 link 'eth0'
```

Sets the interface to listen for multicast packets on. Could be a loopback, not yet tested.

```
set interfaces vxlan vxlan241 vni '241'
```

Sets the unique id for this vxlan-interface. Not sure how it correlates with multicast-address.

```
set interfaces vxlan vxlan241 remote-port 12345
```

The destination port used for creating a VXLAN interface in Linux defaults to its pre-standard value of 8472 to preserve backwards compatibility. A configuration directive to support a user-specified destination port to override that behavior is available using the above command.

6.12.4 Older Examples

Example for bridging normal L2 segment and vxlan overlay network, and using a vxlan interface as routing interface.

```
interfaces {
     bridge br0 {
     }
     ethernet eth0 {
         address dhcp
     }
     loopback lo {
     }
     vxlan vxlan0 {
         bridge-group {
             bridge br0
         }
         group 239.0.0.1
         vni 0
     }
     vxlan vxlan1 {
         address 192.168.0.1/24
         link eth0
         group 239.0.0.1
         vni 1
     }
}
```

Here is a working configuration that creates a VXLAN between two routers. Each router has a VLAN interface (26) facing the client devices and a VLAN interface (30) that connects it to the other routers. With this configuration, traffic can flow between both routers' VLAN 26, but can't escape since there is no L3 gateway. You can add an IP to a bridge-group to create a gateway.

```
interfaces {
     bridge br0 {
     }
     ethernet eth0 {
         duplex auto
         smp-affinity auto
         speed auto
         vif 26 {
             bridge-group {
                 bridge br0
             }
         }
         vif 30 {
```

(continues on next page)

(continued from previous page)

```
            address 10.7.50.6/24
        }
    }
    loopback lo {
    }
    vxlan vxlan0 {
        bridge-group {
            bridge br0
        }
        group 239.0.0.241
        vni 241
    }
}
```

6.12.5 Unicast VXLAN

Alternative to multicast, the remote IPv4 address of the VXLAN tunnel can set directly. Let's change the Multicast example from above:

```
# leaf2 and leaf3
delete interfaces vxlan vxlan241 group '239.0.0.241'
delete interfaces vxlan vxlan241 link 'eth0'

# leaf2
set interface vxlan vxlan241 remote 10.1.3.3

# leaf3
set interface vxlan vxlan241 remote 10.1.2.2
```

The default port udp is set to 8472. It can be changed with `set interface vxlan <vxlanN> remote-port <port>`

CHAPTER 7

Routing

VyOS is a "router first" network operating system. It supports static routing, policy routing, and dynamic routing using standard protocols (RIP, OSPF, and BGP).

7.1 Address Resolution Protocol (ARP)

To manipulate or display ARP table entries, the following commands are implemented.

7.1.1 adding a static arp entry

```
set protocols static arp 10.1.1.100 hwaddr 08:00:27:de:23:aa
commit
```

7.1.2 display arp table entries

```
show protocols static arp

Address                  HWtype  HWaddress           Flags Mask            Iface
10.1.1.1                 ether   08:00:27:de:23:2e   C                     eth1
10.1.1.100               ether   08:00:27:de:23:aa   CM                    eth1
```

```
show protocols static arp interface eth1
Address                  HWtype  HWaddress           Flags Mask            Iface
10.1.1.1                 ether   08:00:27:de:23:2e   C                     eth1
10.1.1.100               ether   08:00:27:de:23:aa   CM                    eth1
```

7.2 Border Gateway Protocol (BGP)

7.2.1 IPv4

A simple eBGP configuration:

Node 1:

```
set protocols bgp 65534 neighbor 192.168.0.2 ebgp-multihop '2'
set protocols bgp 65534 neighbor 192.168.0.2 remote-as '65535'
set protocols bgp 65534 neighbor 192.168.0.2 update-source '192.168.0.1'
set protocols bgp 65534 address-family ipv4-unicast network '172.16.0.0/16'
set protocols bgp 65534 parameters router-id '192.168.0.1'
```

Node 2:

```
set protocols bgp 65535 neighbor 192.168.0.1 ebgp-multihop '2'
set protocols bgp 65535 neighbor 192.168.0.1 remote-as '65534'
set protocols bgp 65535 neighbor 192.168.0.1 update-source '192.168.0.2'
set protocols bgp 65535 address-family ipv4-unicast network '172.17.0.0/16'
set protocols bgp 65535 parameters router-id '192.168.0.2'
```

Don't forget, the CIDR declared in the network statement MUST **exist in your routing table (dynamic or static), the best way to make sure that is true is creating a static route:**

Node 1:

```
set protocols static route 172.16.0.0/16 blackhole distance '254'
```

Node 2:

```
set protocols static route 172.17.0.0/16 blackhole distance '254'
```

7.2.2 IPv6

A simple BGP configuration via IPv6.

Node 1:

```
set protocols bgp 65534 neighbor 2001:db8::2 ebgp-multihop '2'
set protocols bgp 65534 neighbor 2001:db8::2 remote-as '65535'
set protocols bgp 65534 neighbor 2001:db8::2 update-source '2001:db8::1'
set protocols bgp 65534 neighbor 2001:db8::2 address-family ipv6-unicast
set protocols bgp 65534 address-family ipv6-unicast network '2001:db8:1::/48'
set protocols bgp 65534 parameters router-id '10.1.1.1'
```

Node 2:

```
set protocols bgp 65535 neighbor 2001:db8::1 ebgp-multihop '2'
set protocols bgp 65535 neighbor 2001:db8::1 remote-as '65534'
set protocols bgp 65535 neighbor 2001:db8::1 update-source '2001:db8::2'
set protocols bgp 65535 neighbor 2001:db8::1 address-family ipv6-unicast
set protocols bgp 65535 address-family ipv6-unicast network '2001:db8:2::/48'
set protocols bgp 65535 parameters router-id '10.1.1.2'
```

Don't forget, the CIDR declared in the network statement **MUST exist in your routing table (dynamic or static), the best way to make sure that is true is creating a static route:**

Node 1:

```
set protocols static route6 2001:db8:1::/48 blackhole distance '254'
```

Node 2:

```
set protocols static route6 2001:db8:2::/48 blackhole distance '254'
```

7.2.3 Route Filter

Route filter can be applied using a route-map:

Node1:

```
set policy prefix-list AS65535-IN rule 10 action 'permit'
set policy prefix-list AS65535-IN rule 10 prefix '172.16.0.0/16'
set policy prefix-list AS65535-OUT rule 10 action 'deny'
set policy prefix-list AS65535-OUT rule 10 prefix '172.16.0.0/16'
set policy prefix-list6 AS65535-IN rule 10 action 'permit'
set policy prefix-list6 AS65535-IN rule 10 prefix '2001:db8:2::/48'
set policy prefix-list6 AS65535-OUT rule 10 action 'deny'
set policy prefix-list6 AS65535-OUT rule 10 prefix '2001:db8:2::/48'
set policy route-map AS65535-IN rule 10 action 'permit'
set policy route-map AS65535-IN rule 10 match ip address prefix-list 'AS65535-IN'
set policy route-map AS65535-IN rule 10 match ipv6 address prefix-list 'AS65535-IN'
set policy route-map AS65535-IN rule 20 action 'deny'
set policy route-map AS65535-OUT rule 10 action 'deny'
set policy route-map AS65535-OUT rule 10 match ip address prefix-list 'AS65535-OUT'
set policy route-map AS65535-OUT rule 10 match ipv6 address prefix-list 'AS65535-OUT'
set policy route-map AS65535-OUT rule 20 action 'permit'
set protocols bgp 65534 neighbor 2001:db8::2 route-map export 'AS65535-OUT'
set protocols bgp 65534 neighbor 2001:db8::2 route-map import 'AS65535-IN'
```

Node2:

```
set policy prefix-list AS65534-IN rule 10 action 'permit'
set policy prefix-list AS65534-IN rule 10 prefix '172.17.0.0/16'
set policy prefix-list AS65534-OUT rule 10 action 'deny'
set policy prefix-list AS65534-OUT rule 10 prefix '172.17.0.0/16'
set policy prefix-list6 AS65534-IN rule 10 action 'permit'
set policy prefix-list6 AS65534-IN rule 10 prefix '2001:db8:1::/48'
set policy prefix-list6 AS65534-OUT rule 10 action 'deny'
set policy prefix-list6 AS65534-OUT rule 10 prefix '2001:db8:1::/48'
set policy route-map AS65534-IN rule 10 action 'permit'
set policy route-map AS65534-IN rule 10 match ip address prefix-list 'AS65534-IN'
set policy route-map AS65534-IN rule 10 match ipv6 address prefix-list 'AS65534-IN'
set policy route-map AS65534-IN rule 20 action 'deny'
set policy route-map AS65534-OUT rule 10 action 'deny'
set policy route-map AS65534-OUT rule 10 match ip address prefix-list 'AS65534-OUT'
set policy route-map AS65534-OUT rule 10 match ipv6 address prefix-list 'AS65534-OUT'
set policy route-map AS65534-OUT rule 20 action 'permit'
set protocols bgp 65535 neighbor 2001:db8::1 route-map export 'AS65534-OUT'
set protocols bgp 65535 neighbor 2001:db8::1 route-map import 'AS65534-IN'
```

We could expand on this and also deny link local and multicast in the rule 20 action deny.

7.3 Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing (LSR) algorithm and falls into the group of interior gateway protocols (IGPs), operating within a single autonomous system (AS). It is defined as OSPF Version 2 in RFC2328 (1998) for IPv4. Updates for IPv6 are specified as OSPF Version 3 in RFC5340 (2008). OSPF supports the Classless Inter-Domain Routing (CIDR) addressing model.

OSPF is a widely used IGP in large enterprise networks.

7.3.1 OSPFv2 (IPv4)

In order to have a VyOS system exchanging routes with OSPF neighbors, you will at least need to configure the area and a network,

```
set protocols ospf area 0 network 192.168.0.0/24
```

as well as the router ID.

```
set protocols ospf parameters router-id 10.1.1.1
```

That is the minimum configuration you will need.

Below you can see a typical configuration using 2 nodes, redistribute loopback address and the node 1 sending the default route:

Node 1

```
set interfaces loopback lo address 10.1.1.1/32
set protocols ospf area 0 network 192.168.0.0/24
set protocols ospf default-information originate always
set protocols ospf default-information originate metric 10
set protocols ospf default-information originate metric-type 2
set protocols ospf log-adjacency-changes
set protocols ospf parameters router-id 10.1.1.1
set protocols ospf redistribute connected metric-type 2
set protocols ospf redistribute connected route-map CONNECT

set policy route-map CONNECT rule 10 action permit
set policy route-map CONNECT rule 10 match interface lo
```

Node 2

```
set interfaces loopback lo address 10.2.2.2/32
set protocols ospf area 0 network 192.168.0.0/24
set protocols ospf log-adjacency-changes
set protocols ospf parameters router-id 10.2.2.2
set protocols ospf redistribute connected metric-type 2
set protocols ospf redistribute connected route-map CONNECT

set policy route-map CONNECT rule 10 action permit
set policy route-map CONNECT rule 10 match interface lo
```

7.3.2 OSPFv3 (IPv6)

A typical configuration using 2 nodes.

Node 1:

```
set protocols ospfv3 area 0.0.0.0 interface eth1
set protocols ospfv3 area 0.0.0.0 range 2001:db8:1::/64
set protocols ospfv3 parameters router-id 192.168.1.1
set protocols ospfv3 redistribute connected
```

Node 2:

```
set protocols ospfv3 area 0.0.0.0 interface eth1
set protocols ospfv3 area 0.0.0.0 range 2001:db8:2::/64
set protocols ospfv3 parameters router-id 192.168.2.1
set protocols ospfv3 redistribute connected
```

Note: You can not easily redistribute IPv6 routes via OSPFv3 on a WireGuard interface link. This requires you to configure link-local addresses manually on the WireGuard interfaces, see Phabricator task T1483.

Example configuration for WireGuard interfaces:

Node 1

```
set interfaces wireguard wg01 address 'fe80::216:3eff:fe51:fd8c/64'
set interfaces wireguard wg01 address '192.168.0.1/24'
set interfaces wireguard wg01 peer ospf02 allowed-ips '::/0'
set interfaces wireguard wg01 peer ospf02 allowed-ips '0.0.0.0/0'
set interfaces wireguard wg01 peer ospf02 endpoint '10.1.1.101:12345'
set interfaces wireguard wg01 peer ospf02 pubkey 'ie3...='
set interfaces wireguard wg01 port '12345'
set protocols ospfv3 parameters router-id 192.168.1.1
set protocols ospfv3 area 0.0.0.0 interface 'wg01'
set protocols ospfv3 area 0.0.0.0 interface 'lo'
```

Node 2

```
set interfaces wireguard wg01 address 'fe80::216:3eff:fe0a:7ada/64'
set interfaces wireguard wg01 address '192.168.0.2/24'
set interfaces wireguard wg01 peer ospf01 allowed-ips '::/0'
set interfaces wireguard wg01 peer ospf01 allowed-ips '0.0.0.0/0'
set interfaces wireguard wg01 peer ospf01 endpoint '10.1.1.100:12345'
set interfaces wireguard wg01 peer ospf01 pubkey 'NHI...='
set interfaces wireguard wg01 port '12345'
set protocols ospfv3 parameters router-id 192.168.1.2
set protocols ospfv3 area 0.0.0.0 interface 'wg01'
set protocols ospfv3 area 0.0.0.0 interface 'lo'
```

Status

```
vyos@ospf01:~$ sh ipv6 ospfv3 neighbor
Neighbor ID     Pri    DeadTime    State/IfState         Duration I/F[State]
192.168.0.2       1    00:00:37     Full/PointToPoint    00:18:03 wg01[PointToPoint]

vyos@ospf02# run sh ipv6 ospfv3 neighbor
Neighbor ID     Pri    DeadTime    State/IfState         Duration I/F[State]
192.168.0.1       1    00:00:39     Full/PointToPoint    00:19:44 wg01[PointToPoint]
```

7.4 Policy-Based Routing (PBR)

VyOS supports Policy Routing, allowing traffic to be assigned to a different routing table. Traffic can be matched using standard 5-tuple matching (source address, destination address, protocol, source port, destination port).

7.4.1 Transparent Proxy

The following example will show how VyOS can be used to redirect web traffic to an external transparent proxy:

```
set policy route FILTER-WEB rule 1000 destination port 80
set policy route FILTER-WEB rule 1000 protocol tcp
set policy route FILTER-WEB rule 1000 set table 100
```

This creates a route policy called FILTER-WEB with one rule to set the routing table for matching traffic (TCP port 80) to table ID 100 instead of the default routing table.

To create routing table 100 and add a new default gateway to be used by traffic matching our route policy:

```
set protocols static table 100 route 0.0.0.0/0 next-hop 10.255.0.2
```

This can be confirmed using the show ip route table 100 operational command.

Finally, to apply the policy route to ingress traffic on our LAN interface, we use:

```
set interfaces ethernet eth1 policy route FILTER-WEB
```

7.4.2 Multiple Uplinks

VyOS Policy-Based Routing (PBR) works by matching source IP address ranges and forwarding the traffic using different routing tables.

Routing tables that will be used in this example are:

- `table 10` Routing tabled used for VLAN 10 (192.168.188.0/24)
- `table 11` Routing tabled used for VLAN 11 (192.168.189.0/24)
- `main` Routing table used by VyOS and other interfaces not paritipating in PBR

Add default routes for routing `table 10` and `table 11`

```
set protocols static table 10 route 0.0.0.0/0 next-hop 192.0.1.1
set protocols static table 11 route 0.0.0.0/0 next-hop 192.0.2.2
```

Add policy route matching VLAN source addresses

```
set policy route PBR rule 20 set table '10'
set policy route PBR rule 20 description 'Route VLAN10 traffic to table 10'
set policy route PBR rule 20 source address '192.168.188.0/24'

set policy route PBR rule 30 set table '11'
set policy route PBR rule 30 description 'Route VLAN11 traffic to table 11'
set policy route PBR rule 30 source address '192.168.189.0/24'
```

Apply routing policy to **inbound** direction of out VLAN interfaces

Fig. 1: Policy-Based Routing with multiple ISP uplinks (source ./draw.io/pbr_example_1.drawio)

```
set interfaces ethernet eth0 vif 10 policy route 'PBR'
set interfaces ethernet eth0 vif 11 policy route 'PBR'
```

OPTIONAL: Exclude Inter-VLAN traffic (between VLAN10 and VLAN11) from PBR

```
set policy route PBR rule 10 description 'VLAN10 <-> VLAN11 shortcut'
set policy route PBR rule 10 destination address '192.168.188.0/24'
set policy route PBR rule 10 destination address '192.168.189.0/24'
set policy route PBR rule 10 set table 'main'
```

Note: Allows the VLAN10 and VLAN20 hosts to communicate with each other using the main routing table.

7.5 Routing Information Protocol (RIP)

Simple RIP configuration using 2 nodes and redistributing connected interfaces.

Node 1:

```
set interfaces loopback address 10.1.1.1/32
set protocols rip network 192.168.0.0/24
set protocols rip redistribute connected
```

Node 2:

```
set interfaces loopback address 10.2.2.2/32
set protocols rip network 192.168.0.0/24
set protocols rip redistribute connected
```

7.6 Static

Static routes are manually configured network routes.

A typical use for a static route is a static default route for systems that do not make use of DHCP or dynamic routing protocols:

```
set protocols static route 0.0.0.0/0 next-hop 10.1.1.1 distance '1'
```

Another common use of static routes is to blackhole (drop) traffic. In the example below, RFC1918 networks are set as blackhole routes.

This prevents these networks leaking out public interfaces, but it does not prevent them from being used as the most specific route has the highest priority.

```
set protocols static route 10.0.0.0/8 blackhole distance '254'
set protocols static route 172.16.0.0/12 blackhole distance '254'
set protocols static route 192.168.0.0/16 blackhole distance '254'
```

Note: Routes with a distance of 255 are effectively disabled and not installed into the kernel.

7.7 TCP-MSS Clamping

As Internet wide PMTU discovery rarely works we sometimes need to clamp our TCP MSS value to a specific value. Starting with VyOS 1.2 there is a firewall option to clamp your TCP MSS value for IPv4 and IPv6.

Clamping can be disabled per interface using the *disable* keywork:

```
set firewall options interface pppoe0 disable
```

7.7.1 IPv4

Clamp outgoing MSS value in a TCP SYN packet to *1452* for *pppoe0* and *1372* for your WireGuard *wg02* tunnel.

```
set firewall options interface pppoe0 adjust-mss '1452'
set firewall options interface wg02 adjust-mss '1372'
```

7.7.2 IPv6

Clamp outgoing MSS value in a TCP SYN packet to *1280* for both *pppoe0* and *wg02* interface.

To achieve the same for IPv6 please use:

```
set firewall options interface pppoe0 adjust-mss6 '1280'
set firewall options interface wg02 adjust-mss6 '1280'
```

Note: MSS value = MTU - 20 (IP header) - 20 (TCP header), resulting in 1452 bytes on a 1492 byte MTU.

7.8 Routing-policy

Routing Policies could be used to tell the router (self or neighbors) what routes and their attributes needs to be put into the routing table.

There could be a wide range of routing policies. Some examples are below:

- Set some metric to routes learned from a particular neighbor
- Set some attributes (like AS PATH or Community value) to advertised routes to neighbors
- Prefer a specific routing protocol routes over another routing protocol running on the same router

7.8.1 Routing Policy Example

Policy definition:

```
#Create policy
set policy route-map setmet rule 2 action 'permit'
set policy route-map setmet rule 2 set as-path-prepend '2 2 2'

#Apply policy to BGP
set protocols bgp 1 neighbor 1.1.1.2 address-family ipv4-unicast route-map import
↪'setmet'
```

(continues on next page)

(continued from previous page)

```
set protocols bgp 1 neighbor 1.1.1.2 address-family ipv4-unicast soft-reconfiguration
↪'inbound' <<<< ***

*** get policy update without bouncing the neighbor
```

Routes learned before routing policy applied:

```
vyos@vos1:~$ show ip bgp
BGP table version is 0, local router ID is 192.168.56.101
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 22.22.22.22/32   1.1.1.2                  1             0 2 i  < Path

Total number of prefixes 1
```

Routes learned after routing policy applied:

```
vyos@vos1:~$ sho ip b
BGP table version is 0, local router ID is 192.168.56.101
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 22.22.22.22/32   1.1.1.2                  1             0 2 2 2 2 i < longer AS_
↪path length

Total number of prefixes 1
vyos@vos1:~$
```

7.9 IGMP Proxy

Internet Group Management Protocol (IGMP)

A IGMP Proxy to send IGMP host messages on behalf of a connected client. The configuration must define one upstream interface, and one or more downstream interfaces. If multicast traffic originates outside the upstream subnet, the "alt-subnet" option can be used in order to define legal multicast sources.

7.9.1 simple example:

Interface eth1 LAN is behind NAT. In order to subscribe 10.0.0.0/23 subnet multicast which is in eth0 WAN we need igmp-proxy.

```
# show protocols igmp-proxy
interface eth0 {
    alt-subnet 10.0.0.0/23
    role upstream
}
interface eth1 {
```

(continues on next page)

(continued from previous page)

```
    role downstream
}
```

CHAPTER 8

Firewall

VyOS makes use of Linux netfilter for packet filtering.

The firewall supports the creation of groups for ports, addresses, and networks (implemented using netfilter ipset) and the option of interface or zone based firewall policy.

Important note on usage of terms: The firewall makes use of the terms *in*, *out*, and *local* for firewall policy. Users experienced with netfilter often confuse *in* to be a reference to the *INPUT* chain, and *out* the *OUTPUT* chain from netfilter. This is not the case. These instead indicate the use of the *FORWARD* chain and either the input or output interface. The *INPUT* chain, which is used for local traffic to the OS, is a reference to as *local* with respect to its input interface.

8.1 Zone-based Firewall Policy

As an alternative to applying policy to an interface directly, a zone-based firewall can be created to simplify configuration when multiple interfaces belong to the same security zone. Instead of applying to rulesets to interfaces they are applied to source zone-destination zone pairs.

An example to zone-based firewalls can be found here: *Zone-Policy example*.

8.2 Groups

Firewall groups represent collections of IP addresses, networks, or ports. Once created, a group can be referenced by firewall rules as either a source or destination. Members can be added or removed from a group without changes to or the need to reload individual firewall rules.

Note: Groups can also be referenced by NAT configuration.

While network groups accept IP networks in CIDR notation, specific IP addresses can be added as a 32-bit prefix. If you foresee the need to add a mix of addresses and networks, the network group is recommended.

Here is an example of a network group for the IP networks that make up the internal network:

```
set firewall group network-group NET-INSIDE network 192.168.0.0/24
set firewall group network-group NET-INSIDE network 192.168.1.0/24
```

A port group represents only port numbers, not the protocol. Port groups can be referenced for either TCP or UDP. It is recommended that TCP and UDP groups are created separately to avoid accidentally filtering unnecessary ports. Ranges of ports can be specified by using -.

Here is an example of a port group a server:

```
set firewall group port-group PORT-TCP-SERVER1 port 80
set firewall group port-group PORT-TCP-SERVER1 port 443
set firewall group port-group PORT-TCP-SERVER1 port 5000-5010
```

8.3 Rule-Sets

A rule-set is a named collection of firewall rules that can be applied to an interface or zone. Each rule is numbered, has an action to apply if the rule is matched, and the ability to specify the criteria to match.

Example of a rule-set to filter traffic to the internal network:

```
set firewall name INSIDE-OUT default-action drop
set firewall name INSIDE-OUT rule 1010 action accept
set firewall name INSIDE-OUT rule 1010 state established enable
set firewall name INSIDE-OUT rule 1010 state related enable
set firewall name INSIDE-OUT rule 1020 action drop
set firewall name INSIDE-OUT rule 1020 state invalid enable
```

8.4 Applying a Rule-Set to an Interface

Once a rule-set is created, it can be applied to an interface.

Note: Only one rule-set can be applied to each interface for *in*, *out*, or *local* traffic for each protocol (IPv4 and IPv6).

```
set interfaces ethernet eth1 firewall out name INSIDE-OUT
```

8.5 Applying a Rule-Set to a Zone

A named rule-set can also be applied to a zone relationship (note, zones must first be created):

```
set zone-policy zone INSIDE from OUTSIDE firewall name INSIDE-OUT
```

8.6 How VyOS replies when being pinged

By default, when VyOS receives an ICMP echo request packet destined for itself, it will answer with an ICMP echo reply, unless you avoid it through its firewall.

With the firewall you can set rules to accept, drop or reject ICMP in, out or local traffic. You can also use the general **firewall all-ping** command. This command affects only to LOCAL (packets destined for your VyOS system), not to IN or OUT traffic.

Note: **firewall all-ping** affects only to LOCAL and it always behaves in the most restrictive way

```
set firewall all-ping enable
```

When the command above is set, VyOS will answer every ICMP echo request addressed to itself, but that will only happen if no other rule is applied droping or rejecting local echo requests. In case of conflict, VyOS will not answer ICMP echo requests.

```
set firewall all-ping disable
```

When the comand above is set, VyOS will answer no ICMP echo request addressed to itself at all, no matter where it comes from or whether more specific rules are being applied to accept them.

8.7 Example Partial Config

```
firewall {
   all-ping enable
   broadcast-ping disable
   config-trap disable
   group {
       network-group BAD-NETWORKS {
           network 198.51.100.0/24
           network 203.0.113.0/24
       }
       network-group GOOD-NETWORKS {
           network 192.0.2.0/24
       }
       port-group BAD-PORTS {
           port 65535
       }
   }
   name FROM-INTERNET {
       default-action accept
       description "From the Internet"
       rule 10 {
           action accept
           description "Authorized Networks"
           protocol all
           source {
               group {
                   network-group GOOD-NETWORKS
               }
           }
       }
       rule 11 {
           action drop
           description "Bad Networks"
           protocol all
           source {
```

(continues on next page)

(continued from previous page)

```
                group {
                    network-group BAD-NETWORKS
                }
            }
        }
        rule 30 {
            action drop
            description "BAD PORTS"
            destination {
                group {
                    port-group BAD-PORTS
                }
            }
            log enable
            protocol all
        }
    }
}
interfaces {
    ethernet eth1 {
        address dhcp
        description OUTSIDE
        duplex auto
        firewall {
            in {
                name FROM-INTERNET
            }
        }
    }
}
```

CHAPTER 9

NAT

9.1 Source NAT

Source NAT is typically referred to simply as NAT. To be more correct, what most people refer to as NAT is actually the process of **Port Address Translation (PAT)**, or **NAT Overload**. The process of having many internal host systems communicate to the Internet using a single or subset of IP addresses.

To setup SNAT, we need to know:

- The internal IP addresses we want to translate;
- The outgoing interface to perform the translation on;
- The external IP address to translate to.

In the example used for the Quick Start configuration above, we demonstrate the following configuration:

```
set nat source rule 100 outbound-interface 'eth0'
set nat source rule 100 source address '192.168.0.0/24'
set nat source rule 100 translation address 'masquerade'
```

Which generates the following configuration:

```
rule 100 {
    outbound-interface eth0
    source {
        address 192.168.0.0/24
    }
    translation {
        address masquerade
    }
}
```

In this example, we use **masquerade** as the translation address instead of an IP address. The **masquerade** target is effectively an alias to say "use whatever IP address is on the outgoing interface", rather than a statically configured IP

address. This is useful if you use DHCP for your outgoing interface and do not know what the external address will be.

When using NAT for a large number of host systems it recommended that a minimum of 1 IP address is used to NAT every 256 host systems. This is due to the limit of 65,000 port numbers available for unique translations and a reserving an average of 200-300 sessions per host system.

Example: For an ~8,000 host network a source NAT pool of 32 IP addresses is recommended.

A pool of addresses can be defined by using a **-** in the *set nat source rule [n] translation address* statement.

```
set nat source rule 100 translation address '203.0.113.32-203.0.113.63'
```

Note: Avoiding "leaky" NAT

Linux netfilter will not NAT traffic marked as INVALID. This often confuses people into thinking that Linux (or specifically VyOS) has a broken NAT implementation because non-NATed traffic is seen leaving an external interface. This is actually working as intended, and a packet capture of the "leaky" traffic should reveal that the traffic is either an additional TCP "RST", "FIN,ACK", or "RST,ACK" sent by client systems after Linux netfilter considers the connection closed. The most common is the additional TCP RST some host implementations send after terminating a connection (which is implementation- specific).

In other words, connection tracking has already observed the connection be closed and has transition the flow to INVALID to prevent attacks from attempting to reuse the connection.

You can avoid the "leaky" behavior by using a firewall policy that drops "invalid" state packets.

Having control over the matching of INVALID state traffic, e.g. the ability to selectively log, is an important troubleshooting tool for observing broken protocol behavior. For this reason, VyOS does not globally drop invalid state traffic, instead allowing the operator to make the determination on how the traffic is handled.

9.1.1 NAT Reflection/Hairpin NAT

Note: Avoiding NAT breakage in the absence of split-DNS

A typical problem with using NAT and hosting public servers is the ability for internal systems to reach an internal server using it's external IP address. The solution to this is usually the use of split-DNS to correctly point host systems to the internal address when requests are made internally. Because many smaller networks lack DNS infrastructure, a work-around is commonly deployed to facilitate the traffic by NATing the request from internal hosts to the source address of the internal interface on the firewall. This technique is commonly referred to as **NAT Reflection**, or **Hairpin NAT**.

In this example, we will be using the example Quick Start configuration above as a starting point.

To setup a NAT reflection rule, we need to create a rule to NAT connections from the internal network to the same internal network to use the source address of the internal interface.

```
set nat source rule 110 description 'NAT Reflection: INSIDE'
set nat source rule 110 destination address '192.168.0.0/24'
set nat source rule 110 outbound-interface 'eth1'
set nat source rule 110 source address '192.168.0.0/24'
set nat source rule 110 translation address 'masquerade'
```

Which results in a configuration of:

```
rule 110 {
    description "NAT Reflection: INSIDE"
    destination {
        address 192.168.0.0/24
    }
    outbound-interface eth1
    source {
        address 192.168.0.0/24
    }
    translation {
        address masquerade
    }
}
```

9.2 Destination NAT

DNAT is typically referred to as a **Port Forward**. When using VyOS as a NAT router and firewall, a common configuration task is to redirect incoming traffic to a system behind the firewall.

In this example, we will be using the example Quick Start configuration above as a starting point.

To setup a destination NAT rule we need to gather:

- The interface traffic will be coming in on;
- The protocol and port we wish to forward;
- The IP address of the internal system we wish to forward traffic to.

In our example, we will be forwarding web server traffic to an internal web server on 192.168.0.100. HTTP traffic makes use of the TCP protocol on port 80. For other common port numbers, see: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Our configuration commands would be:

```
set nat destination rule 10 description 'Port Forward: HTTP to 192.168.0.100'
set nat destination rule 10 destination port '80'
set nat destination rule 10 inbound-interface 'eth0'
set nat destination rule 10 protocol 'tcp'
set nat destination rule 10 translation address '192.168.0.100'
```

Which would generate the following NAT destination configuration:

```
nat {
    destination {
        rule 10 {
            description "Port Forward: HTTP to 192.168.0.100"
            destination {
                port 80
            }
            inbound-interface eth0
            protocol tcp
            translation {
                address 192.168.0.100
            }
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Note: If forwarding traffic to a different port than it is arriving on, you may also configure the translation port using *set nat destination rule [n] translation port*.

This establishes our Port Forward rule, but if we created a firewall policy it will likely block the traffic.

It is important to note that when creating firewall rules that the DNAT translation occurs **before** traffic traverses the firewall. In other words, the destination address has already been translated to 192.168.0.100.

So in our firewall policy, we want to allow traffic coming in on the outside interface, destined for TCP port 80 and the IP address of 192.168.0.100.

```
set firewall name OUTSIDE-IN rule 20 action 'accept'
set firewall name OUTSIDE-IN rule 20 destination address '192.168.0.100'
set firewall name OUTSIDE-IN rule 20 destination port '80'
set firewall name OUTSIDE-IN rule 20 protocol 'tcp'
set firewall name OUTSIDE-IN rule 20 state new 'enable'
```

This would generate the following configuration:

```
rule 20 {
    action accept
    destination {
        address 192.168.0.100
        port 80
    }
    protocol tcp
    state {
        new enable
    }
}
```

Note: If you have configured the *INSIDE-OUT* policy, you will need to add additional rules to permit inbound NAT traffic.

9.3 1-to-1 NAT

Another term often used for DNAT is **1-to-1 NAT**. For a 1-to-1 NAT configuration, both DNAT and SNAT are used to NAT all traffic from an external IP address to an internal IP address and vice-versa.

Typically, a 1-to-1 NAT rule omits the destination port (all ports) and replaces the protocol with either **all** or **ip**.

Then a corresponding SNAT rule is created to NAT outgoing traffic for the internal IP to a reserved external IP. This dedicates an external IP address to an internal IP address and is useful for protocols which don't have the notion of ports, such as GRE.

9.4 1-to-1 NAT example

Here's an extract of a simple 1-to-1 NAT configuration with one internal and one external interface:

```
set interfaces ethernet eth0 address '192.168.1.1/24'
set interfaces ethernet eth0 description 'Inside interface'
set interfaces ethernet eth1 address '192.0.2.30/24'
set interfaces ethernet eth1 description 'Outside interface'
set nat destination rule 2000 description '1-to-1 NAT example'
set nat destination rule 2000 destination address '192.0.2.30'
set nat destination rule 2000 inbound-interface 'eth1'
set nat destination rule 2000 translation address '192.168.1.10'
set nat source rule 2000 description '1-to-1 NAT example'
set nat source rule 2000 outbound-interface 'eth1'
set nat source rule 2000 source address '192.168.1.10'
set nat source rule 2000 translation address '192.0.2.30'
```

Firewall rules are written as normal, using the internal IP address as the source of outbound rules and the destination of inbound rules.

9.5 NPTv6 (RFC6296)

NPTv6 stands for Network Prefix Translation. It's a form of NAT for IPv6. It's described in RFC6296. NPTv6 is supported in linux kernel since version 3.13.

9.5.1 Usage

NPTv6 is very useful for IPv6 multihoming. It is also commonly used when the external IPv6 prefix is dynamic, as it prevents the need for renumbering of internal hosts when the extern prefix changes.

Let's assume the following network configuration:

- eth0 : LAN
- eth1 : WAN1, with 2001:db8:e1::/48 routed towards it
- eth2 : WAN2, with 2001:db8:e2::/48 routed towards it

Regarding LAN hosts addressing, why would you choose 2001:db8:e1::/48 over 2001:db8:e2::/48? What happens when you get a new provider with a different routed IPv6 subnet?

The solution here is to assign to your hosts ULAs and to prefix-translate their address to the right subnet when going through your router.

- LAN Subnet : fc00:dead:beef::/48
- WAN 1 Subnet : 2001:db8:e1::/48
- WAN 2 Subnet : 2001:db8:e2::/48
- eth0 addr : fc00:dead:beef::1/48
- eth1 addr : 2001:db8:e1::1/48
- eth2 addr : 2001:db8:e2::1/48

9.5.2 VyOS Support

NPTv6 support has been added in VyOS 1.2 (Crux) and is available through *nat nptv6* configuration nodes.

```
set rule 10 inside-prefix 'fc00:dead:beef::/48'
set rule 10 outside-interface 'eth1'
set rule 10 outside-prefix '2001:db8:e1::/48'
set rule 20 inside-prefix 'fc00:dead:beef::/48'
set rule 20 outside-interface 'eth2'
set rule 20 outside-prefix '2001:db8:e2::/48'
```

Resulting in the following ip6tables rules:

```
Chain VYOS_DNPT_HOOK (1 references)
 pkts bytes target   prot opt in   out   source               destination
    0     0 DNPT     all    eth1   any   anywhere             2001:db8:e1::/48  src-
↪pfx 2001:db8:e1::/48 dst-pfx fc00:dead:beef::/48
    0     0 DNPT     all    eth2   any   anywhere             2001:db8:e2::/48  src-
↪pfx 2001:db8:e2::/48 dst-pfx fc00:dead:beef::/48
    0     0 RETURN   all    any    any   anywhere             anywhere
Chain VYOS_SNPT_HOOK (1 references)
 pkts bytes target   prot opt in   out   source               destination
    0     0 SNPT     all    any    eth1  fc00:dead:beef::/48 anywhere          src-
↪pfx fc00:dead:beef::/48 dst-pfx 2001:db8:e1::/48
    0     0 SNPT     all    any    eth2  fc00:dead:beef::/48 anywhere          src-
↪pfx fc00:dead:beef::/48 dst-pfx 2001:db8:e2::/48
    0     0 RETURN   all    any    any   anywhere             anywhere
```

9.6 NAT before VPN

Some application service providers (ASPs) operate a VPN gateway to provide access to their internal resources, and require that a connecting organisation translate all traffic to the service provider network to a source address provided by the ASP.

9.6.1 Example Network

Here's one example of a network environment for an ASP. The ASP requests that all connections from this company should come from 172.29.41.89 - an address that is assigned by the ASP and not in use at the customer site.

Fig. 1: NAT before VPN Topology

9.6.2 Configuration

The required configuration can be broken down into 4 major pieces:

- A dummy interface for the provider-assigned IP;
- NAT (specifically, Source NAT);
- IPSec IKE and ESP Groups;
- IPSec VPN tunnels.

Dummy interface

The dummy interface allows us to have an equivalent of the Cisco IOS Loopback interface - a router-internal interface we can use for IP addresses the router must know about, but which are not actually assigned to a real network.

We only need a single step for this interface:

```
set interfaces dummy dum0 address '172.29.41.89/32'
```

NAT Configuration

```
set nat source rule 110 description 'Internal to ASP'
set nat source rule 110 destination address '172.27.1.0/24'
set nat source rule 110 outbound-interface 'any'
set nat source rule 110 source address '192.168.43.0/24'
set nat source rule 110 translation address '172.29.41.89'
set nat source rule 120 description 'Internal to ASP'
set nat source rule 120 destination address '10.125.0.0/16'
set nat source rule 120 outbound-interface 'any'
set nat source rule 120 source address '192.168.43.0/24'
set nat source rule 120 translation address '172.29.41.89'
```

IPSec IKE and ESP

The ASP has documented their IPSec requirements:

- IKE Phase:
  - aes256 Encryption
  - sha256 Hashes
- ESP Phase:
  - aes256 Encryption
  - sha256 Hashes
  - DH Group 14

Additionally, we want to use VPNs only on our eth1 interface (the external interface in the image above)

```
set vpn ipsec ike-group my-ike ikev2-reauth 'no'
set vpn ipsec ike-group my-ike key-exchange 'ikev1'
set vpn ipsec ike-group my-ike lifetime '7800'
set vpn ipsec ike-group my-ike proposal 1 dh-group '14'
```

(continues on next page)

(continued from previous page)

```
set vpn ipsec ike-group my-ike proposal 1 encryption 'aes256'
set vpn ipsec ike-group my-ike proposal 1 hash 'sha256'

set vpn ipsec esp-group my-esp compression 'disable'
set vpn ipsec esp-group my-esp lifetime '3600'
set vpn ipsec esp-group my-esp mode 'tunnel'
set vpn ipsec esp-group my-esp pfs 'disable'
set vpn ipsec esp-group my-esp proposal 1 encryption 'aes256'
set vpn ipsec esp-group my-esp proposal 1 hash 'sha256'

set vpn ipsec ipsec-interfaces interface 'eth1'
```

IPSec VPN Tunnels

We'll use the IKE and ESP groups created above for this VPN. Because we need access to 2 different subnets on the far side, we will need two different tunnels. If you changed the names of the ESP group and IKE group in the previous step, make sure you use the correct names here too.

```
set vpn ipsec site-to-site peer 198.51.100.243 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 198.51.100.243 authentication pre-shared-secret
↪'PASSWORD IS HERE'
set vpn ipsec site-to-site peer 198.51.100.243 connection-type 'initiate'
set vpn ipsec site-to-site peer 198.51.100.243 default-esp-group 'my-esp'
set vpn ipsec site-to-site peer 198.51.100.243 ike-group 'my-ike'
set vpn ipsec site-to-site peer 198.51.100.243 ikev2-reauth 'inherit'
set vpn ipsec site-to-site peer 198.51.100.243 local-address '203.0.113.46'
set vpn ipsec site-to-site peer 198.51.100.243 tunnel 0 local prefix '172.29.41.89/32'
set vpn ipsec site-to-site peer 198.51.100.243 tunnel 0 remote prefix '172.27.1.0/24'
set vpn ipsec site-to-site peer 198.51.100.243 tunnel 1 local prefix '172.29.41.89/32'
set vpn ipsec site-to-site peer 198.51.100.243 tunnel 1 remote prefix '10.125.0.0/16'
```

9.6.3 Testing and Validation

If you've completed all the above steps you no doubt want to see if it's all working.

Start by checking for IPSec SAs (Security Associations) with:

```
$ show vpn ipsec sa

Peer ID / IP                            Local ID / IP
------------                            -------------
198.51.100.243                          203.0.113.46

    Tunnel  State  Bytes Out/In   Encrypt  Hash    NAT-T  A-Time  L-Time  Proto
    ------  -----  -------------  -------  ----    -----  ------  ------  -----
    0       up     0.0/0.0        aes256   sha256  no     1647    3600    all
    1       up     0.0/0.0        aes256   sha256  no     865     3600    all
```

That looks good - we defined 2 tunnels and they're both up and running.

CHAPTER 10

VPN

This chapter descriptes the available VPN services provided by VyOS.

10.1 OpenVPN

Traditionally hardware routers implement IPsec exclusively due to relative ease of implementing it in hardware and insufficient CPU power for doing encryption in software. Since VyOS is a software router, this is less of a concern. OpenVPN has been widely used on UNIX platform for a long time and is a popular option for remote access VPN, though it's also capable of site-to-site connections.

Advantages of OpenVPN are:

- It uses a single TCP or UDP connection and does not rely on packet source addresses, so it will work even through a double NAT: perfect for public hotspots and such
- It's easy to setup and offers very flexible split tunneling
- There's a variety of client GUI frontends for any platform

Disadvantages are:

- It's slower than IPsec due to higher protocol overhead and the fact it runs in user mode while IPsec, on Linux, is in kernel mode
- None of the operating systems have client software installed by default

In the VyOS CLI, a key point often overlooked is that rather than being configured using the *set vpn* stanza, OpenVPN is configured as a network interface using *set interfaces openvpn*.

10.1.1 OpenVPN Site-To-Site

While many are aware of OpenVPN as a Client VPN solution, it is often overlooked as a site-to-site VPN solution due to lack of support for this mode in many router platforms.

Site-to-site mode supports x.509 but doesn't require it and can also work with static keys, which is simpler in many cases. In this example, we'll configure a simple site-to-site OpenVPN tunnel using a 2048-bit pre-shared key.

First, one one of the systems generate the key using the operational command *generate openvpn key <filename>*. This will generate a key with the name provided in the */config/auth/* directory. Once generated, you will need to copy this key to the remote router.

In our example, we used the filename *openvpn-1.key* which we will reference in our configuration.

- The public IP address of the local side of the VPN will be 198.51.100.10
- The remote will be 203.0.113.11
- The tunnel will use 10.255.1.1 for the local IP and 10.255.1.2 for the remote.
- OpenVPN allows for either TCP or UDP. UDP will provide the lowest latency, while TCP will work better for lossy connections; generally UDP is preferred when possible.
- The official port for OpenVPN is 1194, which we reserve for client VPN; we will use 1195 for site-to-site VPN.
- The *persistent-tunnel* directive will allow us to configure tunnel-related attributes, such as firewall policy as we would on any normal network interface.
- If known, the IP of the remote router can be configured using the *remote-host* directive; if unknown, it can be omitted. We will assume a dynamic IP for our remote router.

Local Configuration:

```
set interfaces openvpn vtun1 mode site-to-site
set interfaces openvpn vtun1 protocol udp
set interfaces openvpn vtun1 persistent-tunnel
set interfaces openvpn vtun1 local-host '198.51.100.10'
set interfaces openvpn vtun1 local-port '1195'
set interfaces openvpn vtun1 remote-port '1195'
set interfaces openvpn vtun1 shared-secret-key-file '/config/auth/openvpn-1.key'
set interfaces openvpn vtun1 local-address '10.255.1.1'
set interfaces openvpn vtun1 remote-address '10.255.1.2'
```

Remote Configuration:

```
set interfaces openvpn vtun1 mode site-to-site
set interfaces openvpn vtun1 protocol udp
set interfaces openvpn vtun1 persistent-tunnel
set interfaces openvpn vtun1 remote-host '198.51.100.10'
set interfaces openvpn vtun1 local-port '1195'
set interfaces openvpn vtun1 remote-port '1195'
set interfaces openvpn vtun1 shared-secret-key-file '/config/auth/openvpn-1.key'
set interfaces openvpn vtun1 local-address '10.255.1.2'
set interfaces openvpn vtun1 remote-address '10.255.1.1'
```

The configurations above will default to using 128-bit Blowfish in CBC mode for encryption and SHA-1 for HMAC authentication. These are both considered weak, but a number of other encryption and hashing algorithms are available:

For Encryption:

```
vyos@vyos# set interfaces openvpn vtun1 encryption
Possible completions:
  des          DES algorithm
  3des         DES algorithm with triple encryption
  bf128        Blowfish algorithm with 128-bit key
  bf256        Blowfish algorithm with 256-bit key
```

(continues on next page)

(continued from previous page)

```
  aes128       AES algorithm with 128-bit key
  aes192       AES algorithm with 192-bit key
  aes256       AES algorithm with 256-bit key
```

For Hashing:

```
vyos@vyos# set interfaces openvpn vtun1 hash
Possible completions:
  md5          MD5 algorithm
  sha1         SHA-1 algorithm
  sha256       SHA-256 algorithm
  sha512       SHA-512 algorithm
```

If you change the default encryption and hashing algorithms, be sure that the local and remote ends have matching configurations, otherwise the tunnel will not come up.

Static routes can be configured referencing the tunnel interface; for example, the local router will use a network of 10.0.0.0/16, while the remote has a network of 10.1.0.0/16:

Local Configuration:

```
set protocols static interface-route 10.1.0.0/16 next-hop-interface vtun1
```

Remote Configuration:

```
set protocols static interface-route 10.0.0.0/16 next-hop-interface vtun1
```

Firewall policy can also be applied to the tunnel interface for *local*, *in*, and *out* directions and function identically to ethernet interfaces.

If making use of multiple tunnels, OpenVPN must have a way to distinguish between different tunnels aside from the pre-shared-key. This is either by referencing IP address or port number. One option is to dedicate a public IP to each tunnel. Another option is to dedicate a port number to each tunnel (e.g. 1195,1196,1197. . .).

OpenVPN status can be verified using the *show openvpn* operational commands. See the built-in help for a complete list of options.

10.1.2 OpenVPN Server

Multi-client server is the most popular OpenVPN mode on routers. It always uses x.509 authentication and therefore requires a PKI setup. This guide assumes you have already setup a PKI and have a CA certificate, a server certificate and key, a certificate revokation list, a Diffie-Hellman key exchange parameters file. You do not need client certificates and keys for the server setup.

In this example we will use the most complicated case: a setup where each client is a router that has its own subnet (think HQ and branch offices), since simpler setups are subsets of it.

Suppose you want to use 10.23.1.0/24 network for client tunnel endpoints and all client subnets belong to 10.23.0.0/20. All clients need access to the 192.168.0.0/16 network.

First we need to specify the basic settings. 1194/UDP is the default. The *persistent-tunnel* option is recommended, it prevents the TUN/TAP device from closing on connection resets or daemon reloads.

Note: Using **openvpn-option -reneg-sec** can be tricky. This option is used to renegotiate data channel after n seconds. When used at both server and client, the lower value will trigger the renegotiation. If you set it to 0 on one side of the connection (to disable it), the chosen value on the other side will determine when the renegotiation will occur.

```
set interfaces openvpn vtun10 mode server
set interfaces openvpn vtun10 local-port 1194
set interfaces openvpn vtun10 persistent-tunnel
set interfaces openvpn vtun10 protocol udp
```

Then we need to specify the location of the cryptographic materials. Suppose you keep the files in */config/auth/openvpn*

```
set interfaces openvpn vtun10 tls ca-cert-file /config/auth/openvpn/ca.crt
set interfaces openvpn vtun10 tls cert-file /config/auth/openvpn/server.crt
set interfaces openvpn vtun10 tls key-file /config/auth/openvpn/server.key
set interfaces openvpn vtun10 tls crl-file /config/auth/openvpn/crl.pem
set interfaces openvpn vtun10 tls dh-file /config/auth/openvpn/dh2048.pem
```

Now we need to specify the server network settings. In all cases we need to specify the subnet for client tunnel endpoints. Since we want clients to access a specific network behind out router, we will use a push-route option for installing that route on clients.

```
set interfaces openvpn vtun10 server push-route 192.168.0.0/16
set interfaces openvpn vtun10 server subnet 10.23.1.0/24
```

Since it's a HQ and branch offices setup, we will want all clients to have fixed addresses and we will route traffic to specific subnets through them. We need configuration for each client to achieve this.

Note: Clients are identified by the CN field of their x.509 certificates, in this example the CN is `client0`:

```
set interfaces openvpn vtun10 server client client0 ip 10.23.1.10
set interfaces openvpn vtun10 server client client0 subnet 10.23.2.0/25
```

OpenVPN **will not** automatically create routes in the kernel for client subnets when they connect and will only use client-subnet association internally, so we need to create a route to the 10.23.0.0/20 network ourselves:

```
set protocols static interface-route 10.23.0.0/20 next-hop-interface vtun10
```

Client Authentication

OpenLDAP

Enterprise installations usually ship a kind of directory service which is used to have a single password store for all employes. VyOS and OpenVPN support using LDAP/AD as single user backend.

Authentication is done by using the `openvpn-auth-ldap.so` plugin which is shiped with every VyOS installation. A dedicated configuration file is required. It is best practise to store it in `/config` to survive image updates

```
set interfaces openvpn vtun0 openvpn-option "--plugin /usr/lib/openvpn/openvpn-auth-
↪ldap.so /config/auth/ldap-auth.config"
```

The required config file may look like:

```
<LDAP>
# LDAP server URL
URL             ldap://ldap.example.com
# Bind DN (If your LDAP server doesn't support anonymous binds)
BindDN          cn=LDAPUser,dc=example,dc=com
```

(continues on next page)

(continued from previous page)

```
# Bind Password password
Password        S3cr3t
# Network timeout (in seconds)
Timeout         15
</LDAP>

<Authorization>
# Base DN
BaseDN          "ou=people,dc=example,dc=com"
# User Search Filter
SearchFilter    "(&(uid=%u)(objectClass=shadowAccount))"
# Require Group Membership - allow all users
RequireGroup    false
</Authorization>
```

Active Directory

Despite the fact that AD is a superset of LDAP

```
<LDAP>
  # LDAP server URL
  URL ldap://dc01.example.com
  # Bind DN (If your LDAP server doesn't support anonymous binds)
  BindDN CN=LDAPUser,DC=example,DC=com
  # Bind Password
  Password mysecretpassword
  # Network timeout (in seconds)
  Timeout  15
  # Enable Start TLS
  TLSEnable no
  # Follow LDAP Referrals (anonymously)
  FollowReferrals no
</LDAP>

<Authorization>
  # Base DN
  BaseDN        "DC=example,DC=com"
  # User Search Filter, user must be a member of the VPN AD group
  SearchFilter  "(&(sAMAccountName=%u)(memberOf=CN=VPN,OU=Groups,DC=example,DC=com))"
  # Require Group Membership
  RequireGroup    false # already handled by SearchFilter
  <Group>
    BaseDN        "OU=Groups,DC=example,DC=com"
    SearchFilter  "(|(cn=VPN))"
    MemberAttribute  memberOf
  </Group>
</Authorization>
```

If you only wan't to check if the user account is enabled and can authenticate (against the primary group) the following snipped is sufficient:

```
<LDAP>
  URL ldap://dc01.example.com
  BindDN CN=SA_OPENVPN,OU=ServiceAccounts,DC=example,DC=com
  Password ThisIsTopSecret
```

(continues on next page)

(continued from previous page)

```
  Timeout  15
  TLSEnable no
  FollowReferrals no
</LDAP>

<Authorization>
  BaseDN          "DC=example,DC=com"
  SearchFilter    "sAMAccountName=%u"
  RequireGroup    false
</Authorization>
```

A complete LDAP auth OpenVPN configuration could look like the following example:

```
vyos@vyos# show interfaces openvpn
 openvpn vtun0 {
     mode server
     openvpn-option "--tun-mtu 1500 --fragment 1300 --mssfix"
     openvpn-option "--plugin /usr/lib/openvpn/openvpn-auth-ldap.so /config/auth/ldap-
↪auth.config"
     openvpn-option "--push redirect-gateway"
     openvpn-option --duplicate-cn
     openvpn-option --client-cert-not-required
     openvpn-option --comp-lzo
     openvpn-option --persist-key
     openvpn-option --persist-tun
     server {
         domain-name example.com
         max-connections 5
         name-server 1.1.1.1
         name-server 9.9.9.9
         subnet 172.18.100.128/29
     }
     tls {
         ca-cert-file /config/auth/ca.crt
         cert-file /config/auth/server.crt
         dh-file /config/auth/dh1024.pem
         key-file /config/auth/server.key
     }
 }
```

10.1.3 OpenVPN Client

VyOS can not only act as an OpenVPN site-to-site or Server for multiple clients. You can indeed also configure any VyOS OpenVPN interface as an OpenVPN client connecting to a VyOS OpenVPN server or any other OpenVPN server.

Given the following example we have one VyOS router acting as OpenVPN server and another VyOS router acting as OpenVPN client. The Server also pushes a static client IP address to the OpenVPN client. Remember, clients are identified using their CN attribute in the SSL certificate.

Server

```
set interfaces openvpn vtun10 encryption 'aes256'
set interfaces openvpn vtun10 hash 'sha512'
set interfaces openvpn vtun10 local-host '172.18.201.10'
set interfaces openvpn vtun10 local-port '1194'
set interfaces openvpn vtun10 mode 'server'
set interfaces openvpn vtun10 persistent-tunnel
set interfaces openvpn vtun10 protocol 'udp'
set interfaces openvpn vtun10 server client client1 ip '10.10.0.10'
set interfaces openvpn vtun10 server domain-name 'vyos.net'
set interfaces openvpn vtun10 server max-connections '250'
set interfaces openvpn vtun10 server name-server '172.16.254.30'
set interfaces openvpn vtun10 server subnet '10.10.0.0/24'
set interfaces openvpn vtun10 server topology 'subnet'
set interfaces openvpn vtun10 tls ca-cert-file '/config/auth/ca.crt'
set interfaces openvpn vtun10 tls cert-file '/config/auth/server.crt'
set interfaces openvpn vtun10 tls dh-file '/config/auth/dh.pem'
set interfaces openvpn vtun10 tls key-file '/config/auth/server.key'
set interfaces openvpn vtun10 use-lzo-compression
```

Client

```
set interfaces openvpn vtun10 encryption 'aes256'
set interfaces openvpn vtun10 hash 'sha512'
set interfaces openvpn vtun10 mode 'client'
set interfaces openvpn vtun10 persistent-tunnel
set interfaces openvpn vtun10 protocol 'udp'
set interfaces openvpn vtun10 remote-host '172.18.201.10'
set interfaces openvpn vtun10 remote-port '1194'
set interfaces openvpn vtun10 tls ca-cert-file '/config/auth/ca.crt'
set interfaces openvpn vtun10 tls cert-file '/config/auth/client1.crt'
set interfaces openvpn vtun10 tls key-file '/config/auth/client1.key'
set interfaces openvpn vtun10 use-lzo-compression
```

10.1.4 Options

We do not have CLI nodes for every single OpenVPN options. If an option is missing, a feature request should be opened at https://phabricator.vyos.net so all users can benefit from it.

If you are a hacker or want to try on your own we support passing raw OpenVPN options to OpenVPN.

```
set interfaces openvpn vtun10 openvpn-option 'persistent-key'
```

Will add *persistent-key* at the end of the generated OpenVPN configuration. Please use this only as last resort - things might break and OpenVPN won't start if you pass invalid options/syntax.

10.2 L2TP over IPsec

Example for configuring a simple L2TP over IPsec VPN for remote access (works with native Windows and Mac VPN clients):

```
set vpn ipsec ipsec-interfaces interface eth0
set vpn ipsec nat-traversal enable
set vpn ipsec nat-networks allowed-network 0.0.0.0/0

set vpn l2tp remote-access outside-address 203.0.113.2
set vpn l2tp remote-access client-ip-pool start 192.168.255.1
set vpn l2tp remote-access client-ip-pool stop 192.168.255.254
set vpn l2tp remote-access ipsec-settings authentication mode pre-shared-secret
set vpn l2tp remote-access ipsec-settings authentication pre-shared-secret <secret>
set vpn l2tp remote-access authentication mode local
set vpn l2tp remote-access authentication local-users username <username> password
↪<password>
```

In the example above an external IP of 203.0.113.2 is assumed.

If a local firewall policy is in place on your external interface you will need to open:

- UDP port 500 (IKE)
- IP protocol number 50 (ESP)
- UDP port 1701 for IPsec

In addition when NAT is detected by the VPN client ESP is encapsulated in UDP for NAT-traversal:

- UDP port 4500 (NAT-T)

Example:

```
set firewall name OUTSIDE-LOCAL rule 40 action 'accept'
set firewall name OUTSIDE-LOCAL rule 40 destination port '50'
set firewall name OUTSIDE-LOCAL rule 40 protocol 'esp'
set firewall name OUTSIDE-LOCAL rule 41 action 'accept'
set firewall name OUTSIDE-LOCAL rule 41 destination port '500'
set firewall name OUTSIDE-LOCAL rule 41 protocol 'udp'
set firewall name OUTSIDE-LOCAL rule 42 action 'accept'
set firewall name OUTSIDE-LOCAL rule 42 destination port '4500'
set firewall name OUTSIDE-LOCAL rule 42 protocol 'udp'
set firewall name OUTSIDE-LOCAL rule 43 action 'accept'
set firewall name OUTSIDE-LOCAL rule 43 destination port '1701'
set firewall name OUTSIDE-LOCAL rule 43 ipsec 'match-ipsec'
set firewall name OUTSIDE-LOCAL rule 43 protocol 'udp'
```

Also note that if you wish to allow the VPN to be used for external access you will need to add the appropriate source NAT rules to your configuration.

```
set nat source rule 110 outbound-interface 'eth0'
set nat source rule 110 source address '192.168.255.0/24'
set nat source rule 110 translation address masquerade
```

To be able to resolve when connected to the VPN, the following DNS rules are needed as well.

```
set vpn l2tp remote-access dns-servers server-1 '8.8.8.8'
set vpn l2tp remote-access dns-servers server-2 '8.8.4.4'
```

Note: Those are the Google public DNS servers. You can also use the public available servers from Quad9 (9.9.9.9) or Cloudflare (1.1.1.1).

Established sessions can be viewed using the **show vpn remote-access** operational command.

```
vyos@vyos:~$ show vpn remote-access
Active remote access VPN sessions:
User            Proto Iface     Tunnel IP       TX byte RX byte  Time
----            ----- -----     ---------       ------- -------  ----
vyos            L2TP  l2tp0     192.168.255.1      3.2K    8.0K  00h06m13s
```

10.2.1 RADIUS authentication

The above configuration made use of local accounts on the VyOS router for authenticating L2TP/IPSec clients. In bigger environments usually something like RADIUS (FreeRADIUS or Microsoft Network Policy Server, NPS) is used.

VyOS supports either *local* or *radius* user authentication:

```
set vpn l2tp remote-access authentication mode <local|radius>
```

In addition one or more RADIUS servers can be configured to server for user authentication. This is done using the *radius server* and *radius server key* nodes:

```
set vpn l2tp remote-access authentication radius server 1.1.1.1 key 'foo'
set vpn l2tp remote-access authentication radius server 2.2.2.2 key 'foo'
```

Note: Some RADIUS severs make use of an access control list who is allowed to query the server. Please configure your VyOS router in the allowed client list.

RADIUS source address

If you are using e.g. OSPF as IGP always the nearest interface facing the RADIUS server is used. With VyOS 1.2 you can bind all outgoing RADIUS requests to a single source IP e.g. the loopback interface.

```
set vpn l2tp remote-access authentication radius source-address 3.3.3.3
```

Above command will use *3.3.3.3* as source IPv4 address for all RADIUS queries on this NAS.

10.3 Site-to-Site

Site-to-site mode provides a way to add remote peers, which could be configured to exchange encrypted information between them and VyOS itself or connected/routed networks.

To configure site-to-site connection you need to add peers with the `set vpn ipsec site-to-site` command.

You can identify a remote peer with:

- IPv4 or IPv6 address. This mode is easiest for configuration and mostly used when a peer has a public static IP address;
- Hostname. This mode is similar to IP address, only you define DNS name instead of an IP. Could be used when a peer has a public IP address and DNS name, but an IP address could be changed from time to time;
- Remote ID of the peer. In this mode, there is no predefined remote address nor DNS name of the peer. This mode is useful when a peer doesn't have a publicly available IP address (NAT between it and VyOS), or IP address could be changed.

Each site-to-site peer has the next options:

- `authentication` - configure authentication between VyOS and a remote peer. Suboptions:
  - `id` - ID for the local VyOS router. If defined, during the authentication it will be send to remote peer;
  - `mode` - mode for authentication between VyOS and remote peer:
  - `pre-shared-secret` - use predefined shared secret phrase, must be the same for local and remote side;
  - `rsa` - use simple shared RSA key. The key must be defined in the `set vpn rsa-keys` section;
  - `x509` - use certificates infrastructure for authentication.
  - `pre-shared-secret` - predefined shared secret. Used if configured `mode pre-shared-secret`;
  - `remote-id` - define an ID for remote peer, instead of using peer name or address. Useful in case if the remote peer is behind NAT or if `mode x509` is used;
  - `rsa-key-name` - shared RSA key for authentication. The key must be defined in the `set vpn rsa-keys` section;
  - `use-x509-id` - use local ID from x509 certificate. Cannot be used when `id` is defined;
  - `x509` - options for x509 authentication mode:
    - `ca-cert-file` - CA certificate file. Using for authenticating remote peer;
    - `cert-file` - certificate file, which will be used for authenticating local router on remote peer;
    - `crl-file` - file with the Certificate Revocation List. Using to check if a certificate for the remote peer is valid or revocated;
    - `key` - a private key, which will be used for authenticating local router on remote peer:
    - `file` - path to the key file;
    - `password` - passphrase private key, if needed.
- `connection-type` - how to handle this connection process. Possible variants:
- `initiate` - do initial connection to remote peer immediately after configuring and after boot. In this mode the connection will not be restarted in case of disconnection, therefore should be used only together with DPD or another session tracking methods;
- `respond` - do not try to initiate a connection to a remote peer. In this mode, the IPSec session will be established only after initiation from a remote peer. Could be useful when there is no direct connectivity to the peer due to firewall or NAT in the middle of the local and remote side.
- `default-esp-group` - ESP group to use by default for traffic encryption. Might be overwritten by individual settings for tunnel or VTI interface binding;
- `description` - description for this peer;
- `dhcp-interface` - use an IP address, received from DHCP for IPSec connection with this peer, instead of `local-address`;
- `force-encapsulation` - force encapsulation of ESP into UDP datagrams. Useful in case if between local and remote side is firewall or NAT, which not allows passing plain ESP packets between them;
- `ike-group` - IKE group to use for key exchanges;
- `ikev2-reauth` - reauthenticate remote peer during the rekeying process. Can be used only with IKEv2:

- `yes` - create a new IKE_SA from the scratch and try to recreate all IPsec SAs;
- `no` - rekey without uninstalling the IPsec SAs;
- `inherit` - use default behavior for the used IKE group.
- `local-address` - local IP address for IPSec connection with this peer. If defined `any`, then an IP address which configured on interface with default route will be used;
- `tunnel` - define criteria for traffic to be matched for encrypting and send it to a peer:
  - `disable` - disable this tunnel;
  - `esp-group` - define ESP group for encrypt traffic, defined by this tunnel;
  - `local` - define a local source for match traffic, which should be encrypted and send to this peer:
  - `port` - define port. Have effect only when used together with `prefix`;
  - `prefix` - IP network at local side.
  - `protocol` - define the protocol for match traffic, which should be encrypted and send to this peer;
  - `remote` - define the remote destination for match traffic, which should be encrypted and send to this peer:
  - `port` - define port. Have effect only when used together with `prefix`;
  - `prefix` - IP network at remote side.
- `vti` - use a VTI interface for traffic encryption. Any traffic, which will be send to VTI interface will be encrypted and send to this peer. Using VTI makes IPSec configuration much flexible and easier in complex situation, and allows to dynamically add/delete remote networks, reachable via a peer, as in this mode router don't need to create additional SA/policy for each remote network:
- `bind` - select a VTI interface to bind to this peer;
- `esp-group` - define ESP group for encrypt traffic, passed this VTI interface.

10.3.1 Examples:

IKEv1

Example:

- WAN interface on *eth1*
- left subnet: *192.168.0.0/24* site1, server side (i.e. locality, actually there is no client or server roles)
- left local_ip: *198.51.100.3* # server side WAN IP
- right subnet: *10.0.0.0/24* site2,remote office side
- right local_ip: *203.0.113.2* # remote office side WAN IP

```
# server config
set vpn ipsec esp-group office-srv-esp compression 'disable'
set vpn ipsec esp-group office-srv-esp lifetime '1800'
set vpn ipsec esp-group office-srv-esp mode 'tunnel'
set vpn ipsec esp-group office-srv-esp pfs 'enable'
set vpn ipsec esp-group office-srv-esp proposal 1 encryption 'aes256'
set vpn ipsec esp-group office-srv-esp proposal 1 hash 'sha1'
set vpn ipsec ike-group office-srv-ike ikev2-reauth 'no'
```

(continues on next page)

(continued from previous page)

```
set vpn ipsec ike-group office-srv-ike key-exchange 'ikev1'
set vpn ipsec ike-group office-srv-ike lifetime '3600'
set vpn ipsec ike-group office-srv-ike proposal 1 encryption 'aes256'
set vpn ipsec ike-group office-srv-ike proposal 1 hash 'sha1'
set vpn ipsec ipsec-interfaces interface 'eth1'
set vpn ipsec site-to-site peer 203.0.113.2 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 203.0.113.2 authentication pre-shared-secret
↪'SomePreSharedKey'
set vpn ipsec site-to-site peer 203.0.113.2 ike-group 'office-srv-ike'
set vpn ipsec site-to-site peer 203.0.113.2 local-address '198.51.100.3'
set vpn ipsec site-to-site peer 203.0.113.2 tunnel 0 allow-nat-networks 'disable'
set vpn ipsec site-to-site peer 203.0.113.2 tunnel 0 allow-public-networks 'disable'
set vpn ipsec site-to-site peer 203.0.113.2 tunnel 0 esp-group 'office-srv-esp'
set vpn ipsec site-to-site peer 203.0.113.2 tunnel 0 local prefix '192.168.0.0/24'
set vpn ipsec site-to-site peer 203.0.113.2 tunnel 0 remote prefix '10.0.0.0/21'

# remote office config
set vpn ipsec esp-group office-srv-esp compression 'disable'
set vpn ipsec esp-group office-srv-esp lifetime '1800'
set vpn ipsec esp-group office-srv-esp mode 'tunnel'
set vpn ipsec esp-group office-srv-esp pfs 'enable'
set vpn ipsec esp-group office-srv-esp proposal 1 encryption 'aes256'
set vpn ipsec esp-group office-srv-esp proposal 1 hash 'sha1'
set vpn ipsec ike-group office-srv-ike ikev2-reauth 'no'
set vpn ipsec ike-group office-srv-ike key-exchange 'ikev1'
set vpn ipsec ike-group office-srv-ike lifetime '3600'
set vpn ipsec ike-group office-srv-ike proposal 1 encryption 'aes256'
set vpn ipsec ike-group office-srv-ike proposal 1 hash 'sha1'
set vpn ipsec ipsec-interfaces interface 'eth1'
set vpn ipsec site-to-site peer 198.51.100.3 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 198.51.100.3 authentication pre-shared-secret
↪'SomePreSharedKey'
set vpn ipsec site-to-site peer 198.51.100.3 ike-group 'office-srv-ike'
set vpn ipsec site-to-site peer 198.51.100.3 local-address '203.0.113.2'
set vpn ipsec site-to-site peer 198.51.100.3 tunnel 0 allow-nat-networks 'disable'
set vpn ipsec site-to-site peer 198.51.100.3 tunnel 0 allow-public-networks 'disable'
set vpn ipsec site-to-site peer 198.51.100.3 tunnel 0 esp-group 'office-srv-esp'
set vpn ipsec site-to-site peer 198.51.100.3 tunnel 0 local prefix '10.0.0.0/21'
set vpn ipsec site-to-site peer 198.51.100.3 tunnel 0 remote prefix '192.168.0.0/24'
```

Show status of new setup:

```
vyos@srv-gw0:~$ show vpn ike sa
Peer ID / IP                            Local ID / IP
------------                            -------------
203.0.113.2                                 198.51.100.3
   State  Encrypt  Hash    D-H Grp  NAT-T  A-Time  L-Time
   -----  -------  ----    -------  -----  ------  ------
   up     aes256   sha1    5        no     734     3600

vyos@srv-gw0:~$ show vpn ipsec sa
Peer ID / IP                            Local ID / IP
------------                            -------------
203.0.113.2                                 198.51.100.3
   Tunnel  State  Bytes Out/In   Encrypt  Hash    NAT-T  A-Time  L-Time  Proto
   ------  -----  -------------  -------  ----    -----  ------  ------  -----
   0       up     7.5M/230.6K    aes256   sha1    no     567     1800    all
```

If there is SNAT rules on eth1, need to add exclude rule

```
# server side
set nat source rule 10 destination address '10.0.0.0/24'
set nat source rule 10 'exclude'
set nat source rule 10 outbound-interface 'eth1'
set nat source rule 10 source address '192.168.0.0/24'

# remote office side
set nat source rule 10 destination address '192.168.0.0/24'
set nat source rule 10 'exclude'
set nat source rule 10 outbound-interface 'eth1'
set nat source rule 10 source address '10.0.0.0/24'
```

To allow traffic to pass through to clients, you need to add the following rules. (if you used the default configuration at the top of this page)

```
# server side
set firewall name OUTSIDE-LOCAL rule 32 action 'accept'
set firewall name OUTSIDE-LOCAL rule 32 source address '10.0.0.0/24'

# remote office side
set firewall name OUTSIDE-LOCAL rule 32 action 'accept'
set firewall name OUTSIDE-LOCAL rule 32 source address '192.168.0.0/24'
```

IKEv2

Imagine the following topology

Note: Don't get confused about the used /31 tunnel subnet. RFC3031 gives you additional information for using /31 subnets on point-to-point links.

left

```
set interfaces vti vti10 address '10.0.0.2/31'

set vpn ipsec esp-group ESP_DEFAULT compression 'disable'
set vpn ipsec esp-group ESP_DEFAULT lifetime '3600'
set vpn ipsec esp-group ESP_DEFAULT mode 'tunnel'
set vpn ipsec esp-group ESP_DEFAULT pfs 'dh-group19'
set vpn ipsec esp-group ESP_DEFAULT proposal 10 encryption 'aes256gcm128'
set vpn ipsec esp-group ESP_DEFAULT proposal 10 hash 'sha256'
set vpn ipsec ike-group IKEv2_DEFAULT dead-peer-detection action 'hold'
set vpn ipsec ike-group IKEv2_DEFAULT dead-peer-detection interval '30'
set vpn ipsec ike-group IKEv2_DEFAULT dead-peer-detection timeout '120'
set vpn ipsec ike-group IKEv2_DEFAULT ikev2-reauth 'no'
set vpn ipsec ike-group IKEv2_DEFAULT key-exchange 'ikev2'
set vpn ipsec ike-group IKEv2_DEFAULT lifetime '10800'
set vpn ipsec ike-group IKEv2_DEFAULT mobike 'disable'
set vpn ipsec ike-group IKEv2_DEFAULT proposal 10 dh-group '19'
set vpn ipsec ike-group IKEv2_DEFAULT proposal 10 encryption 'aes256gcm128'
set vpn ipsec ike-group IKEv2_DEFAULT proposal 10 hash 'sha256'
set vpn ipsec ipsec-interfaces interface 'eth0.201'
set vpn ipsec site-to-site peer 172.18.202.10 authentication id '172.18.201.10'
set vpn ipsec site-to-site peer 172.18.202.10 authentication mode 'pre-shared-secret'
```

(continues on next page)

Fig. 1: IPSec IKEv2 site2site VPN (source ./draw.io/vpn_s2s_ikev2.drawio)

(continued from previous page)

```
set vpn ipsec site-to-site peer 172.18.202.10 authentication pre-shared-secret
↪'secretkey'
set vpn ipsec site-to-site peer 172.18.202.10 authentication remote-id '172.18.202.10'
set vpn ipsec site-to-site peer 172.18.202.10 connection-type 'initiate'
set vpn ipsec site-to-site peer 172.18.202.10 ike-group 'IKEv2_DEFAULT'
set vpn ipsec site-to-site peer 172.18.202.10 ikev2-reauth 'inherit'
set vpn ipsec site-to-site peer 172.18.202.10 local-address '172.18.201.10'
set vpn ipsec site-to-site peer 172.18.202.10 vti bind 'vti10'
set vpn ipsec site-to-site peer 172.18.202.10 vti esp-group 'ESP_DEFAULT'
```

right

```
set interfaces vti vti10 address '10.0.0.3/31'

set vpn ipsec esp-group ESP_DEFAULT compression 'disable'
set vpn ipsec esp-group ESP_DEFAULT lifetime '3600'
set vpn ipsec esp-group ESP_DEFAULT mode 'tunnel'
set vpn ipsec esp-group ESP_DEFAULT pfs 'dh-group19'
set vpn ipsec esp-group ESP_DEFAULT proposal 10 encryption 'aes256gcm128'
set vpn ipsec esp-group ESP_DEFAULT proposal 10 hash 'sha256'
set vpn ipsec ike-group IKEv2_DEFAULT dead-peer-detection action 'hold'
set vpn ipsec ike-group IKEv2_DEFAULT dead-peer-detection interval '30'
set vpn ipsec ike-group IKEv2_DEFAULT dead-peer-detection timeout '120'
set vpn ipsec ike-group IKEv2_DEFAULT ikev2-reauth 'no'
set vpn ipsec ike-group IKEv2_DEFAULT key-exchange 'ikev2'
set vpn ipsec ike-group IKEv2_DEFAULT lifetime '10800'
set vpn ipsec ike-group IKEv2_DEFAULT mobike 'disable'
set vpn ipsec ike-group IKEv2_DEFAULT proposal 10 dh-group '19'
set vpn ipsec ike-group IKEv2_DEFAULT proposal 10 encryption 'aes256gcm128'
set vpn ipsec ike-group IKEv2_DEFAULT proposal 10 hash 'sha256'
set vpn ipsec ipsec-interfaces interface 'eth0.202'
set vpn ipsec site-to-site peer 172.18.201.10 authentication id '172.18.202.10'
set vpn ipsec site-to-site peer 172.18.201.10 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 172.18.201.10 authentication pre-shared-secret
↪'secretkey'
set vpn ipsec site-to-site peer 172.18.201.10 authentication remote-id '172.18.201.10'
set vpn ipsec site-to-site peer 172.18.201.10 connection-type 'initiate'
set vpn ipsec site-to-site peer 172.18.201.10 ike-group 'IKEv2_DEFAULT'
set vpn ipsec site-to-site peer 172.18.201.10 ikev2-reauth 'inherit'
set vpn ipsec site-to-site peer 172.18.201.10 local-address '172.18.202.10'
set vpn ipsec site-to-site peer 172.18.201.10 vti bind 'vti10'
set vpn ipsec site-to-site peer 172.18.201.10 vti esp-group 'ESP_DEFAULT'
```

10.4 GRE/IPsec

Generic Routing Encapsulation (GRE), GRE/IPsec (or IPIP/IPsec, SIT/IPsec, or any other stateless tunnel protocol over IPsec) is the usual way to protect the traffic inside a tunnel.

An advantage of this scheme is that you get a real interface with its own address, which makes it easier to setup static routes or use dynamic routing protocols without having to modify IPsec policies. The other advantage is that it greatly simplifies router to router communication, which can be tricky with plain IPsec because the external outgoing address of the router usually doesn't match the IPsec policy of typical site-to-site setup and you need to add special configuration for it, or adjust the source address for outgoing traffic of your applications. GRE/IPsec has no such problem and is completely transparent for the applications.

GRE/IPIP/SIT and IPsec are widely accepted standards, which make this scheme easy to implement between VyOS and virtually any other router.

For simplicity we'll assume that the protocol is GRE, it's not hard to guess what needs to be changed to make it work with a different protocol. We assume that IPsec will use pre-shared secret authentication and will use AES128/SHA1 for the cipher and hash. Adjust this as necessary.

Note: VMWare users should ensure that VMXNET3 adapters used, e1000 adapters have known issue with GRE processing

10.4.1 IPsec policy matching GRE

The first and arguably cleaner option is to make your IPsec policy match GRE packets between external addresses of your routers. This is the best option if both routers have static external addresses.

Suppose the LEFT router has external address 192.0.2.10 on its eth0 interface, and the RIGHT router is 203.0.113.45

On the LEFT:

```
# GRE tunnel
set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 local-ip 192.0.2.10
set interfaces tunnel tun0 remote-ip 203.0.113.45
set interfaces tunnel tun0 address 10.10.10.1/30

## IPsec
set vpn ipsec ipsec-interfaces interface eth0

# IKE group
set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group '2'
set vpn ipsec ike-group MyIKEGroup proposal 1 encryption 'aes128'
set vpn ipsec ike-group MyIKEGroup proposal 1 hash 'sha1'

# ESP group
set vpn ipsec esp-group MyESPGroup proposal 1 encryption 'aes128'
set vpn ipsec esp-group MyESPGroup proposal 1 hash 'sha1'

# IPsec tunnel
set vpn ipsec site-to-site peer 203.0.113.45 authentication mode pre-shared-secret
set vpn ipsec site-to-site peer 203.0.113.45 authentication pre-shared-secret␣
↪MYSECRETKEY

set vpn ipsec site-to-site peer 203.0.113.45 ike-group MyIKEGroup
set vpn ipsec site-to-site peer 203.0.113.45 default-esp-group MyESPGroup

set vpn ipsec site-to-site peer 203.0.113.45 local-address 192.0.2.10

# This will match all GRE traffic to the peer
set vpn ipsec site-to-site peer 203.0.113.45 tunnel 1 protocol gre
```

On the RIGHT, setup by analogy and swap local and remote addresses.

10.4.2 Source tunnel from loopbacks

The scheme above doesn't work when one of the routers has a dynamic external address though. The classic workaround for this is to setup an address on a loopback interface and use it as a source address for the GRE tunnel, then setup an IPsec policy to match those loopback addresses.

We assume that the LEFT router has static 192.0.2.10 address on eth0, and the RIGHT router has a dynamic address on eth0.

Setting up the GRE tunnel

On the LEFT:

```
set interfaces loopback lo address 192.168.99.1/32

set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 address 10.10.10.1/30
set interfaces tunnel tun0 local-ip 192.168.99.1
set interfaces tunnel tun0 remote-ip 192.168.99.2
```

On the RIGHT:

```
set interfaces loopback lo address 192.168.99.2/32

set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 address 10.10.10.2/30
set interfaces tunnel tun0 local-ip 192.168.99.2
set interfaces tunnel tun0 remote-ip 192.168.99.1
```

Setting up IPSec

However, now you need to make IPsec work with dynamic address on one side. The tricky part is that pre-shared secret authentication doesn't work with dynamic address, so we'll have to use RSA keys.

First, on both routers run the operational command "generate vpn rsa-key bits 2048". You may choose different length than 2048 of course.

```
vyos@left# run generate vpn rsa-key bits 2048
Generating rsa-key to /config/ipsec.d/rsa-keys/localhost.key

Your new local RSA key has been generated
The public portion of the key is:

0sAQO2335[long string here]
```

Then on the opposite router, add the RSA key to your config.

```
set vpn rsa-keys rsa-key-name LEFT rsa-key KEYGOESHERE
```

Now you are ready to setup IPsec. You'll need to use an ID instead of address for the peer on the dynamic side.

On the LEFT (static address):

```
set vpn rsa-keys rsa-key-name RIGHT rsa-key <PUBLIC KEY FROM THE RIGHT>

set vpn ipsec ipsec-interfaces interface eth0

set vpn ipsec esp-group MyESPGroup proposal 1 encryption aes128
set vpn ipsec esp-group MyESPGroup proposal 1 hash sha1
```

(continues on next page)

(continued from previous page)

```
set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group 2
set vpn ipsec ike-group MyIKEGroup proposal 1 encryption aes128
set vpn ipsec ike-group MyIKEGroup proposal 1 hash sha1

set vpn ipsec site-to-site peer @RIGHT authentication mode rsa
set vpn ipsec site-to-site peer @RIGHT authentication rsa-key-name RIGHT
set vpn ipsec site-to-site peer @RIGHT default-esp-group MyESPGroup
set vpn ipsec site-to-site peer @RIGHT ike-group MyIKEGroup
set vpn ipsec site-to-site peer @RIGHT local-address 192.0.2.10
set vpn ipsec site-to-site peer @RIGHT connection-type respond
set vpn ipsec site-to-site peer @RIGHT tunnel 1 local prefix 192.168.99.1/32  #␣
↪Additional loopback address on the local
set vpn ipsec site-to-site peer @RIGHT tunnel 1 remote prefix 192.168.99.2/32 #␣
↪Additional loopback address on the remote
```

On the RIGHT (dynamic address):

```
set vpn rsa-keys rsa-key-name LEFT rsa-key <PUBLIC KEY FROM THE LEFT>

set vpn ipsec ipsec-interfaces interface eth0

set vpn ipsec esp-group MyESPGroup proposal 1 encryption aes128
set vpn ipsec esp-group MyESPGroup proposal 1 hash sha1

set vpn ipsec ike-group MyIKEGroup proposal 1 dh-group 2
set vpn ipsec ike-group MyIKEGroup proposal 1 encryption aes128
set vpn ipsec ike-group MyIKEGroup proposal 1 hash sha1

set vpn ipsec site-to-site peer 192.0.2.10 authentication id @RIGHT
set vpn ipsec site-to-site peer 192.0.2.10 authentication mode rsa
set vpn ipsec site-to-site peer 192.0.2.10 authentication rsa-key-name LEFT
set vpn ipsec site-to-site peer 192.0.2.10 remote-id @LEFT
set vpn ipsec site-to-site peer 192.0.2.10 connection-type initiate
set vpn ipsec site-to-site peer 192.0.2.10 default-esp-group MyESPGroup
set vpn ipsec site-to-site peer 192.0.2.10 ike-group MyIKEGroup
set vpn ipsec site-to-site peer 192.0.2.10 local-address any
set vpn ipsec site-to-site peer 192.0.2.10 tunnel 1 local prefix 192.168.99.2/32  #␣
↪Additional loopback address on the local
set vpn ipsec site-to-site peer 192.0.2.10 tunnel 1 remote prefix 192.168.99.1/32 #␣
↪Additional loopback address on the remote
```

10.5 DMVPN

D ynamic **M** ultipoint **V** irtual **P** rivate **N** etworking

DMVPN is a dynamic VPN technology originally developed by Cisco. While their implementation was somewhat proprietary, the underlying technologies are actually standards based. The three technologies are:

- **NHRP** - NBMA Next Hop Resolution Protocol RFC2332
- **mGRE** - Multipoint Generic Routing Encapsulation / mGRE RFC1702
- **IPSec** - IP Security (too many RFCs to list, but start with RFC4301)

NHRP provides the dynamic tunnel endpoint discovery mechanism (endpoint registration, and endpoint discovery/lookup), mGRE provides the tunnel encapsulation itself, and the IPSec protocols handle the key exchange, and crypto mechanism.

In short, DMVPN provides the capability for creating a dynamic-mesh VPN network without having to pre-configure (static) all possible tunnel end-point peers.

Note: DMVPN only automates the tunnel endpoint discovery and setup. A complete solution also incorporates the use of a routing protocol. BGP is particularly well suited for use with DMVPN.

Baseline Configuration:

STEPS:

1. Create tunnel config (*interfaces tunnel*)
2. Create nhrp (*protocols nhrp*)
3. Create ipsec vpn (optional, but recommended for security) (*vpn ipsec*)

The tunnel will be set to mGRE if for encapsulation *gre* is set, and no *remote-ip* is set. If the public ip is provided by DHCP the tunnel *local-ip* can be set to "0.0.0.0"

Fig. 2: Baseline DMVPN topology

10.5.1 HUB Configuration

```
interfaces
    tunnel <tunN> {
        address <ipv4>
        encapsulation gre
        local-ip <public ip>
        multicast enable
        description <txt>
        parameters {
            ip {
                <usual IP options>
            }
        }
    }
}
protocols {
    nhrp {
        tunnel <tunN> {
```

(continues on next page)

(continued from previous page)

```
            cisco-authentication <key phrase>
            holding-time <seconds>
            multicast dynamic
            redirect
        }
    }
}
vpn {
    ipsec {
        esp-group <text> {
            lifetime <30-86400>
            mode tunnel
            pfs enable
            proposal <1-65535> {
                encryption aes256
                hash sha1
            }
            proposal <1-65535> {
                encryption 3des
                hash md5
            }
        }
        ike-group <text> {
            key-exchange ikev1
            lifetime <30-86400>
            proposal <1-65535> {
                encryption aes256
                hash sha1
            }
            proposal <1-65535> {
                encryption aes128
                hash sha1
            }
        }
        ipsec-interfaces {
            interface <ethN>
        }
        profile <text> {
            authentication {
                mode pre-shared-secret
                pre-shared-secret <key phrase>
            }
            bind {
                tunnel <tunN>
            }
            esp-group <text>
            ike-group <text>
        }
    }
}
```

HUB Example Configuration:

```
set interfaces ethernet eth0 address '198.51.100.41/30'
set interfaces ethernet eth1 address '192.168.1.1/24'
set system host-name 'HUB'
```

(continues on next page)

(continued from previous page)

```
set interfaces tunnel tun0 address 10.0.0.1/24
set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 local-ip 198.51.100.41
set interfaces tunnel tun0 multicast enable
set interfaces tunnel tun0 parameters ip key 1

set protocols nhrp tunnel tun0 cisco-authentication SECRET
set protocols nhrp tunnel tun0 holding-time  300
set protocols nhrp tunnel tun0 multicast dynamic
set protocols nhrp tunnel tun0 redirect

set vpn ipsec ipsec-interfaces interface eth0
set vpn ipsec ike-group IKE-HUB proposal 1
set vpn ipsec ike-group IKE-HUB proposal 1 encryption aes256
set vpn ipsec ike-group IKE-HUB proposal 1 hash sha1
set vpn ipsec ike-group IKE-HUB proposal 2 encryption aes128
set vpn ipsec ike-group IKE-HUB proposal 2 hash sha1
set vpn ipsec ike-group IKE-HUB lifetime 3600
set vpn ipsec esp-group ESP-HUB proposal 1 encryption aes256
set vpn ipsec esp-group ESP-HUB proposal 1 hash sha1
set vpn ipsec esp-group ESP-HUB proposal 2 encryption 3des
set vpn ipsec esp-group ESP-HUB proposal 2 hash md5
set vpn ipsec esp-group ESP-HUB lifetime 1800
set vpn ipsec esp-group ESP-HUB pfs dh-group2

set vpn ipsec profile NHRPVPN
set vpn ipsec profile NHRPVPN authentication mode pre-shared-secret
set vpn ipsec profile NHRPVPN authentication pre-shared-secret SECRET
set vpn ipsec profile NHRPVPN bind tunnel tun0
set vpn ipsec profile NHRPVPN esp-group ESP-HUB
set vpn ipsec profile NHRPVPN ike-group IKE-HUB

set protocols static route 0.0.0.0/0 next-hop 1.1.1.2
set protocols static route 192.168.2.0/24 next-hop 10.0.0.2
set protocols static route 192.168.3.0/24 next-hop 10.0.0.3
```

10.5.2 SPOKE Configuration

SPOKE1 Configuration:

```
interfaces
    tunnel <tunN> {
        address <ipv4>
        encapsulation gre
        local-ip <public ip>
        multicast enable
        description <txt>
        parameters {
            ip {
                <usual IP options>
            }
        }
    }
}
protocols {
```

(continues on next page)

(continued from previous page)

```
    nhrp {
        tunnel <tunN> {
            cisco-authentication <key phrase>
            map <ipv4/net> {
                nbma-address <ipv4>
                register
            }
            holding-time <seconds>
            multicast nhs
            redirect
            shortcut
        }
    }
}
vpn {
    ipsec {
        esp-group <text> {
            lifetime <30-86400>
            mode tunnel
            pfs enable
            proposal <1-65535> {
                encryption aes256
                hash sha1
            }
            proposal <1-65535> {
                encryption 3des
                hash md5
            }
        }
        ike-group <text> {
            key-exchange ikev1
            lifetime <30-86400>
            proposal <1-65535> {
                encryption aes256
                hash sha1
            }
            proposal <1-65535> {
                encryption aes128
                hash sha1
            }
        }
        ipsec-interfaces {
            interface <ethN>
        }
        profile <text> {
            authentication {
                mode pre-shared-secret
                pre-shared-secret <key phrase>
            }
            bind {
                tunnel <tunN>
            }
            esp-group <text>
            ike-group <text>
        }
    }
}
```

SPOKE1 Example Configuration

```
set interfaces ethernet eth0 address 'dhcp'
set interfaces ethernet eth1 address '192.168.2.1/24'
set system host-name 'SPOKE1'

set interfaces tunnel tun0 address 10.0.0.2/24
set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 local-ip 0.0.0.0
set interfaces tunnel tun0 multicast enable
set interfaces tunnel tun0 parameters ip key 1

set protocols nhrp tunnel tun0 cisco-authentication 'SECRET'
set protocols nhrp tunnel tun0 map 10.0.0.1/24 nbma-address 198.51.100.41
set protocols nhrp tunnel tun0 map 10.0.0.1/24 'register'
set protocols nhrp tunnel tun0 multicast 'nhs'
set protocols nhrp tunnel tun0 'redirect'
set protocols nhrp tunnel tun0 'shortcut'

set vpn ipsec ipsec-interfaces interface eth0
set vpn ipsec ike-group IKE-SPOKE proposal 1
set vpn ipsec ike-group IKE-SPOKE proposal 1 encryption aes256
set vpn ipsec ike-group IKE-SPOKE proposal 1 hash sha1
set vpn ipsec ike-group IKE-SPOKE proposal 2 encryption aes128
set vpn ipsec ike-group IKE-SPOKE proposal 2 hash sha1
set vpn ipsec ike-group IKE-SPOKE lifetime 3600
set vpn ipsec esp-group ESP-SPOKE proposal 1 encryption aes256
set vpn ipsec esp-group ESP-SPOKE proposal 1 hash sha1
set vpn ipsec esp-group ESP-SPOKE proposal 2 encryption 3des
set vpn ipsec esp-group ESP-SPOKE proposal 2 hash md5
set vpn ipsec esp-group ESP-SPOKE lifetime 1800
set vpn ipsec esp-group ESP-SPOKE pfs dh-group2

set vpn ipsec profile NHRPVPN
set vpn ipsec profile NHRPVPN authentication mode pre-shared-secret
set vpn ipsec profile NHRPVPN authentication pre-shared-secret SECRET
set vpn ipsec profile NHRPVPN bind tunnel tun0
set vpn ipsec profile NHRPVPN esp-group ESP-SPOKE
set vpn ipsec profile NHRPVPN ike-group IKE-SPOKE

set protocols static route 192.168.1.0/24 next-hop 10.0.0.1
set protocols static route 192.168.3.0/24 next-hop 10.0.0.3
```

SPOKE2 Configuration

```
interfaces
    tunnel <tunN> {
        address <ipv4>
        encapsulation gre
        local-ip <public ip>
        multicast enable
        description <txt>
        parameters {
            ip {
                <usual IP options>
            }
        }
    }
```

(continues on next page)

(continued from previous page)

```
}
protocols {
    nhrp {
        tunnel <tunN> {
            cisco-authentication <key phrase>
            map <ipv4/net> {
                nbma-address <ipv4>
                register
            }
            holding-time <seconds>
            multicast nhs
            redirect
            shortcut
        }
    }
}
vpn {
    ipsec {
        esp-group <text> {
            lifetime <30-86400>
            mode tunnel
            pfs enable
            proposal <1-65535> {
                encryption aes256
                hash sha1
            }
            proposal <1-65535> {
                encryption 3des
                hash md5
            }
        }
        ike-group <text> {
            key-exchange ikev1
            lifetime <30-86400>
            proposal <1-65535> {
                encryption aes256
                hash sha1
            }
            proposal <1-65535> {
                encryption aes128
                hash sha1
            }
        }
        ipsec-interfaces {
            interface <ethN>
        }
        profile <text> {
            authentication {
                mode pre-shared-secret
                pre-shared-secret <key phrase>
            }
            bind {
                tunnel <tunN>
            }
            esp-group <text>
            ike-group <text>
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

SPOKE2 Example Configuration

```
set interfaces ethernet eth0 address 'dhcp'
set interfaces ethernet eth1 address '192.168.3.1/24'
set system host-name 'SPOKE2'

set interfaces tunnel tun0 address 10.0.0.3/24
set interfaces tunnel tun0 encapsulation gre
set interfaces tunnel tun0 local-ip 0.0.0.0
set interfaces tunnel tun0 multicast enable
set interfaces tunnel tun0 parameters ip key 1

set protocols nhrp tunnel tun0 cisco-authentication SECRET
set protocols nhrp tunnel tun0 map 10.0.0.1/24 nbma-address 198.51.100.41
set protocols nhrp tunnel tun0 map 10.0.0.1/24 register
set protocols nhrp tunnel tun0 multicast nhs
set protocols nhrp tunnel tun0 redirect
set protocols nhrp tunnel tun0 shortcut

set vpn ipsec ipsec-interfaces interface eth0
set vpn ipsec ike-group IKE-SPOKE proposal 1
set vpn ipsec ike-group IKE-SPOKE proposal 1 encryption aes256
set vpn ipsec ike-group IKE-SPOKE proposal 1 hash sha1
set vpn ipsec ike-group IKE-SPOKE proposal 2 encryption aes128
set vpn ipsec ike-group IKE-SPOKE proposal 2 hash sha1
set vpn ipsec ike-group IKE-SPOKE lifetime 3600
set vpn ipsec esp-group ESP-SPOKE proposal 1 encryption aes256
set vpn ipsec esp-group ESP-SPOKE proposal 1 hash sha1
set vpn ipsec esp-group ESP-SPOKE proposal 2 encryption 3des
set vpn ipsec esp-group ESP-SPOKE proposal 2 hash md5
set vpn ipsec esp-group ESP-SPOKE lifetime 1800
set vpn ipsec esp-group ESP-SPOKE pfs dh-group2

set vpn ipsec profile NHRPVPN
set vpn ipsec profile NHRPVPN authentication mode pre-shared-secret
set vpn ipsec profile NHRPVPN authentication pre-shared-secret SECRET
set vpn ipsec profile NHRPVPN bind tunnel tun0
set vpn ipsec profile NHRPVPN esp-group ESP-SPOKE
set vpn ipsec profile NHRPVPN ike-group IKE-SPOKE

set protocols static route 192.168.1.0/24 next-hop 10.0.0.1
set protocols static route 192.168.2.0/24 next-hop 10.0.0.2
```

10.6 PPTP-Server

The Point-to-Point Tunneling Protocol (*PPTP*) has been implemented in VyOS only for backwards compatibility. PPTP has many well known security issues and you should use one of the many other new VPN implementations.

As per default and if not otherwise defined, mschap-v2 is being used for authentication and mppe 128-bit (stateless) for encryption. If no gateway-address is set within the configuration, the lowest IP out of the /24 client-ip-pool is being used. For instance, in the example below it would be 192.168.0.1.

10.6.1 server example

```
set vpn pptp remote-access authentication local-users username test password 'test'
set vpn pptp remote-access authentication mode 'local'
set vpn pptp remote-access client-ip-pool start '192.168.0.10'
set vpn pptp remote-access client-ip-pool stop '192.168.0.15'
set vpn pptp remote-access gateway-address '10.100.100.1'
set vpn pptp remote-access outside-address '10.1.1.120'
```

10.6.2 client example (debian 9)

Install the client software via apt and execute pptpsetup to generate the configuration.

```
apt-get install pptp-linux
pptpsetup --create TESTTUNNEL --server 10.1.1.120 --username test --password test --
↪encrypt
pon TESTTUNNEL
```

The command pon TESTUNNEL establishes the PPTP tunnel to the remote system.

All tunnel sessions can be checked via:

```
run sh pptp-server sessions
 ifname | username | calling-sid |      ip      | type | comp | state  |  uptime
--------+----------+-------------+--------------+------+------+--------+----------
 ppp0   | test     | 10.1.1.99   | 192.168.0.10 | pptp | mppe | active | 00:00:58
```

10.7 WireGuard VPN Interface

WireGuard is an extremely simple yet fast and modern VPN that utilizes state-of-the-art cryptography. See https://www.wireguard.com for more information.

10.7.1 Configuration

Wireguard requires the generation of a keypair, a private key which will decrypt incoming traffic and a public key, which the peer(s) will use to encrypt traffic.

Generate a keypair

Generate the keypair, which creates a public and private part and stores it within VyOS. It will be used per default on any configured wireguard interface, even if multiple interfaces are being configured.

```
wg01:~$ configure
wg01# run generate wireguard keypair
```

The public key is being shared with your peer(s), your peer will encrypt all traffic to your system using this public key.

```
wg01# run show wireguard pubkey
u41jO3OF73Gq1WARMMFG7tOfk7+r8o8AzPxJ1FZRhzk=
```

Generate named keypairs

Named keypairs can be used on a interface basis, if configured. If multiple wireguard interfaces are being configured, each can have their own keypairs.

The commands below will generate 2 keypairs, which are not releated to each other.

```
wg01:~$ configure
wg01# run generate wireguard named-keypairs KP01
wg01# run generate wireguard named-keypairs KP02
```

Wireguard Interface configuration

The next step is to configure your local side as well as the policy based trusted destination addresses. If you only initiate a connection, the listen port and endpoint is optional, if you however act as a server and endpoints initiate the connections to your system, you need to define a port your clients can connect to, otherwise it's randomly chosen and may make it difficult with firewall rules, since the port may be a different one when you reboot your system.

You will also need the public key of your peer as well as the network(s) you want to tunnel (allowed-ips) to configure a wireguard tunnel. The public key below is always the public key from your peer, not your local one.

local side

```
set interfaces wireguard wg01 address '10.1.0.1/24'
set interfaces wireguard wg01 description 'VPN-to-wg02'
set interfaces wireguard wg01 peer to-wg02 allowed-ips '10.2.0.0/24'
set interfaces wireguard wg01 peer to-wg02 endpoint '192.168.0.142:12345'
set interfaces wireguard wg01 peer to-wg02 pubkey
↪'XMrlPykaxhdAAiSjhtPlvi30NVkvLQliQuKP7AI7CyI='
set interfaces wireguard wg01 port '12345'
set protocols static interface-route 10.2.0.0/24 next-hop-interface wg01
```

The last step is to define an interface route for 10.2.0.0/24 to get through the wireguard interface *wg01*. Multiple IPs or networks can be defined and routed, the last check is allowed-ips which either prevents or allows the traffic.

To use a named key on an interface, the option private-key needs to be set.

```
set interfaces wireguard wg01 private-key KP01
set interfaces wireguard wg02 private-key KP02
```

The command `run show wireguard named-keypairs pubkey KP01` will then show the public key, which needs to be shared with the peer.

remote side

```
set interfaces wireguard wg01 address '10.2.0.1/24'
set interfaces wireguard wg01 description 'VPN-to-wg01'
set interfaces wireguard wg01 peer to-wg02 allowed-ips '10.1.0.0/24'
set interfaces wireguard wg01 peer to-wg02 endpoint '192.168.0.124:12345'
set interfaces wireguard wg01 peer to-wg02 pubkey
↪'u41jO3OF73Gq1WARMMFG7tOfk7+r8o8AzPxJ1FZRhzk='
set interfaces wireguard wg01 port '12345'
set protocols static interface-route 10.1.0.0/24 next-hop-interface wg01
```

Assure that your firewall rules allow the traffic, in which case you have a working VPN using wireguard.

```
wg01# ping 10.2.0.1
PING 10.2.0.1 (10.2.0.1) 56(84) bytes of data.
64 bytes from 10.2.0.1: icmp_seq=1 ttl=64 time=1.16 ms
64 bytes from 10.2.0.1: icmp_seq=2 ttl=64 time=1.77 ms

wg02# ping 10.1.0.1
PING 10.1.0.1 (10.1.0.1) 56(84) bytes of data.
64 bytes from 10.1.0.1: icmp_seq=1 ttl=64 time=4.40 ms
64 bytes from 10.1.0.1: icmp_seq=2 ttl=64 time=1.02 ms
```

An additional layer of symmetric-key crypto can be used on top of the asymmetric crypto, which is optional.

```
wg01# run generate wireguard preshared-key
rvVDOoc2IYEnV+k5p7TNAmHBMEGTHbPU8Qqg8c/sUqc=
```

Copy the key, as it is not stored on the local file system. Make sure you distribute that key in a safe manner, it's a symmetric key, so only you and your peer should have knowledge of its content.

```
wg01# set interfaces wireguard wg01 peer to-wg02 preshared-key
↪'rvVDOoc2IYEnV+k5p7TNAmHBMEGTHbPU8Qqg8c/sUqc='
wg02# set interfaces wireguard wg01 peer to-wg01 preshared-key
↪'rvVDOoc2IYEnV+k5p7TNAmHBMEGTHbPU8Qqg8c/sUqc='
```

10.7.2 Operational commands

Show interface status

```
vyos@wg01# run show interfaces wireguard wg01

interface: wg01
public key: xHvgSJC8RTClfvjc0oX6OALxU6GGLapjthjw7x82CSw=
private key: (hidden)
listening port: 12345

peer: 9Ek3R30mG6Vk+GHsENtPF0b9Ul+ftxx4dDBa1bdBxX8=
endpoint: 192.168.0.142:12345
allowed ips: 10.2.0.0/24
latest handshake: 4 minutes, 22 seconds ago
transfer: 860 B received, 948 B sent
```

Show public key of the default key

```
vyos@wg01# run show wireguard keypair pubkey default
FAXCPb6EbTlSH5200J5zTopt9AYXneBthAySPBLbZwM=
```

Show public key of a named key

```
vyos@wg01# run show wireguard keypair pubkey KP01
HUtsu198toEnm1poGoRTyqkUKfKUdyh54f45dtcahDM=
```

Delete wireguard keypairs

```
vyos@wg01# wireguard keypair default
```

CHAPTER 11

QoS and Traffic Policy

VyOS uses tc as backend for QoS. VyOS provides its users with configuration nodes for the following shaping/queueing/policing disciplines:

- HTB
- HFSC
- SFQ
- pfifo
- network-emulator
- PRIO
- GRED
- TBF
- DRR

11.1 Configuration nodes

VyOS QoS configuration is done in two steps. The first one consists in setting up your classes/queues and traffic filters to distribute traffic amongst them. The second step is to apply such traffic policy to an interface ingress or egress.

11.1.1 Creating a traffic policy

Such configuration takes place under the *traffic-policy* tree.

Available subtrees :

```
set traffic-policy drop-tail NAME
set traffic-policy fair-queue NAME
set traffic-policy limiter NAME
set traffic-policy network-emulator NAME
set traffic-policy priority-queue NAME
set traffic-policy random-detect NAME
set traffic-policy rate-control NAME
set traffic-policy round-robin NAME
set traffic-policy shaper NAME
set traffic-policy shaper-hfsc NAME
```

11.1.2 Apply traffic policy to an interface

Once a traffic-policy is created, you can apply it to an interface :

```
set interfaces ethernet eth0 traffic-policy in WAN-IN
set interfaces etherhet eth0 traffic-policy out WAN-OUT
```

11.1.3 A Real-World Example

This policy sets download and upload bandwidth maximums (roughly 90% of the speeds possible), then divvies up the traffic into buckets of importance, giving guaranteed bandwidth chunks to types of traffic that are necessary for general interactive internet use, like web browsing, streaming, or gaming.

After identifying and prioritizing that traffic, it drops the remaining traffic into a general-priority bucket, which it gives a lower priority than what is required for real-time use. If there is no real-time traffic that needs the bandwidth, the lower-priority traffic can use most of the connection. This ensures that the connection can be used fully by whatever wants it, without suffocating real-time traffic or throttling background traffic too much.

```
set traffic-policy shaper download bandwidth '175mbit'
set traffic-policy shaper download class 10 bandwidth '10%'
set traffic-policy shaper download class 10 burst '15k'
set traffic-policy shaper download class 10 ceiling '100%'
set traffic-policy shaper download class 10 match dns ip source port '53'
set traffic-policy shaper download class 10 match icmp ip protocol 'icmp'
set traffic-policy shaper download class 10 match ssh ip source port '22'
set traffic-policy shaper download class 10 priority '5'
set traffic-policy shaper download class 10 queue-type 'fair-queue'
set traffic-policy shaper download class 20 bandwidth '10%'
set traffic-policy shaper download class 20 burst '15k'
set traffic-policy shaper download class 20 ceiling '100%'
set traffic-policy shaper download class 20 match http ip source port '80'
set traffic-policy shaper download class 20 match https ip source port '443'
set traffic-policy shaper download class 20 priority '4'
set traffic-policy shaper download class 20 queue-type 'fair-queue'
set traffic-policy shaper download default bandwidth '70%'
set traffic-policy shaper download default burst '15k'
set traffic-policy shaper download default ceiling '100%'
set traffic-policy shaper download default priority '3'
set traffic-policy shaper download default queue-type 'fair-queue'
set traffic-policy shaper upload bandwidth '18mbit'
set traffic-policy shaper upload class 2 bandwidth '10%'
set traffic-policy shaper upload class 2 burst '15k'
set traffic-policy shaper upload class 2 ceiling '100%'
```

(continues on next page)

(continued from previous page)

```
set traffic-policy shaper upload class 2 match ack ip tcp ack
set traffic-policy shaper upload class 2 match dns ip destination port '53'
set traffic-policy shaper upload class 2 match icmp ip protocol 'icmp'
set traffic-policy shaper upload class 2 match ssh ip destination port '22'
set traffic-policy shaper upload class 2 match syn ip tcp syn
set traffic-policy shaper upload class 2 priority '5'
set traffic-policy shaper upload class 2 queue-limit '16'
set traffic-policy shaper upload class 2 queue-type 'fair-queue'
set traffic-policy shaper upload class 5 bandwidth '10%'
set traffic-policy shaper upload class 5 burst '15k'
set traffic-policy shaper upload class 5 ceiling '100%'
set traffic-policy shaper upload class 5 match http ip destination port '80'
set traffic-policy shaper upload class 5 match https ip destination port '443'
set traffic-policy shaper upload class 5 priority '4'
set traffic-policy shaper upload class 5 queue-type 'fair-queue'
set traffic-policy shaper upload default bandwidth '60%'
set traffic-policy shaper upload default burst '15k'
set traffic-policy shaper upload default ceiling '100%'
set traffic-policy shaper upload default priority '3'
set traffic-policy shaper upload default queue-type 'fair-queue'
```

11.2 Traffic policies in VyOS

An overview of QoS traffic policies supported by VyOS.

11.2.1 Drop-tail (FIFO)

A packet queuing mechanism on a FIFO (First In, First Out) basis; packets are sent out in the same order as they arrive. The queue has a defined length, packets arriving after the queue is filled up will be dropped (hence the name *drop tail*, the "tail" of the queue will be dropped). With this policy in place, all traffic is treated equally and put into a single queue. Applicable to outbound traffic only.

Available commands:

- Define a drop-tail policy (unique name, exclusive to this policy):

  ```
  set traffic-policy drop-tail <policy name>
  ```

- Add a description:

  ```
  set traffic-policy drop-tail <policy name> description <description>
  ```

- Set the queue length limit (max. number of packets in queue), range 0…4294967295 packets:

  ```
  set traffic-policy drop-tail <policy name> queue-limit <limit>
  ```

11.2.2 Fair queue (SFQ)

Fair queue is a packet queuing mechanism that separates traffic flows based on their source/destination IP addresses and/or source port and places them into buckets. Bandwidth is allocated fairly between buckets based on the Stochastic airness Queuing algorithm. Applicable to outbound traffic only.

Available commands:

- Define a fair queue policy:

  ```
  set traffic-policy fair-queue <policy name>
  ```

- Add a description:

  ```
  set traffic-policy fair-queue <policy name> description <description>
  ```

- Set hash update interval; the algorithm used is stochastic and thus not 'truly' fair, hash collisions can occur, in which case traffic flows may be put into the same bucket. To mitigate this, the hashes can be updated at a set interval, Range 0…4294967295 seconds:

  ```
  set traffic-policy fair-queue <policy name> hash-interval <seconds>
  ```

- Set the queue-limit (max. number of packets in queue), range 0…4294967295 packets, default 127:

  ```
  set traffic-policy fair-queue <policy name> queue-limit <limit>
  ```

11.2.3 Limiter

The limiter performs ingress policing of traffic flows. Multiple classes of traffic can be defined and traffic limits can be applied to each class. Traffic exceeding the defined bandwidth limits is dropped. Applicable to inbound traffic only.

Available commands:

- Define a traffic limiter policy: `set traffic-policy limiter <policy-name>`
- Add a description: `set traffic-policy limiter <policy-name> description <description>`

Traffic classes

- Define a traffic class for a limiter policy, range for class ID is 1…4095:

  ```
  set traffic-policy limiter <policy-name> class <class ID>
  ```

- Add a class description:

  ```
  set traffic-policy limiter <policy-name> class <class ID> description
  <description>
  ```

- Specify a bandwidth limit for a class, in kbit/s:

  ```
  set traffic-policy limiter <policy-name> class <class ID> bandwidth
  <rate>.
  ```

  Available suffixes:

- kbit (kilobits per second, default)
- mbit (megabits per second)
- gbit (gigabits per second)
- kbps (kilobytes per second)
- mbps (megabytes per second)
- gbps (gigabytes per second)
- Set a burst size for a class, the maximum amount of traffic that can be sent, in bytes:

  ```
  set traffic-policy limiter <policy-name> class <class ID> burst
  <burst-size>.
  ```

Available suffixes:

- kb (kilobytes)
- mb (megabytes)
- gb (gigabytes)

Default class

- Define a default class for a limiter policy that applies to traffic not matching any other classes for this policy:

  ```
  set traffic-policy limiter <policy name> default
  ```

- Specify a bandwidth limit for the default class, in kbit/s:

  ```
  set traffic-policy limiter <policy name> default bandwidth <rate>.
  ```

  Available suffixes:

- kbit (kilobits per second, default)
- mbit (megabits per second)
- gbit (gigabits per second)
- kbps (kilobytes per second)
- mbps (megabytes per second)
- gbps (gigabytes per second)
- Set a burst size for the default class, the maximum amount of traffic that can be sent, in bytes:

  ```
  set traffic-policy limiter <policy-name> default burst <burst-size>.
  ```

  Available suffixes:

- kb (kilobytes)
- mb (megabytes)
- gb (gigabytes)
- Specify the priority of the default class to set the order in which the rules are evaluated, the higher the number the lower the priority, range 0. . . 20 (default 20):

  ```
  set traffic-policy limiter <policy name> default priority <priority>
  ```

Matching rules

- Define a traffic class matching rule:

  ```
  set traffic-policy limiter <policy name> class <class ID> match <match
  name>
  ```

- Add a description:

  ```
  set traffic-policy limiter <policy name> class <class ID> match <match
  name> description <description>
  ```

- Specify the priority of a matching rule to set the order in which the rules are evaluated, the higher the number the lower the priority, range 0. . . 20 (default 20):

```
set traffic-policy limiter <policy name> class <class ID> priority
<priority>
```

- Specify a match criterion based on a **destination MAC address** (format: xx:xx:xx:xx:xx:xx):

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ether destination <MAC address>
```

- Specify a match criterion based on a **source MAC address** (format: xx:xx:xx:xx:xx:xx):

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ether source <MAC address>
```

- Specify a match criterion based on **packet type/protocol**, range 0. . . 65535:

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ether protocol <number>
```

- Specify a match criterion based on the **fwmark field**, range 0. . . .4294967295:

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> mark <fwmark>
```

- Specify a match criterion based on **VLAN ID**, range 1. . . 4096:

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> vif <VLAN ID>
```

IPv4

- Specify a match criterion based on **destination IPv4 address** and/or port, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ip destination <IPv4 address|port>
```

- Specify a match criterion based on **source IPv4 address** and/or port, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ip source <IPv4 address|port>
```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ip dscp <DSCP value>
```

- Specify a match criterion based on **IPv4 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ip protocol <proto>
```

IPv6

- Specify a match criterion based on **destination IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ipv6 destination <IPv6 address|port>
```

- Specify a match criterion based on **source IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ipv6 source <IPv6 address|port>
```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ipv6 dscp <DSCP value>
```

- Specify a match criterion based on **IPv6 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

```
set traffic-policy limiter <policy name> class <class ID> match <match
name> ipv6 protocol <proto>
```

11.2.4 Network emulator

The network emulator policy emulates WAN traffic, which is useful for testing purposes. Applicable to outbound traffic only.

Available commands:

- Define a network emulator policy:

```
set traffic-policy network-emulator <policy name>
```

- Add a description:

```
set traffic-policy network-emulator <policy name> description
<description>
```

- Specify a bandwidth limit in kbit/s:

```
set traffic-policy network-emulator <policy name> bandwidth <rate>
```

Available suffixes:

- kbit (kilobits per second, default)
- mbit (megabits per second)
- gbit (gigabits per second)
- kbps (kilobytes per second)
- mbps (megabytes per second)
- gbps (gigabytes per second)
- Set a burst size, the maximum amount of traffic that can be sent, in bytes:

```
set traffic-policy network-emulator <policy name> burst <burst size>
```

Available suffixes:

- kb (kilobytes)
- mb (megabytes)
- gb (gigabytes)
- Define a delay between packets:

```
set traffic-policy network-emulator <policy name> network-delay <delay>
```

Available suffixes:

- secs (seconds)
- ms (milliseconds, default)
- us (microseconds)
- Set a percentage of corrupted of packets (one bit flip, unchanged checksum):

  ```
  set traffic-policy network-emulator <policy name> packet-corruption
  <percent>
  ```

- Set a percentage of random packet loss:

  ```
  set traffic-policy network-emulator <policy name> packet-loss <percent>
  ```

- Set a percentage of packets for random reordering:

  ```
  set traffic-policy network-emulator <policy name> packet-reordering
  <percent>
  ```

- Set a queue length limit in packets, range 0…4294967295, default 127:

  ```
  set traffic-policy network-emulator <policy name> queue-limit <limit>
  ```

11.2.5 Priority queue

Up to seven queues with differing priorities can be defined, packets are placed into queues based on associated match criteria. Packets are transmitted from the queues in priority order. If queues with a higher order are being filled with packets continuously, packets from lower priority queues will only be transmitted after traffic volume from higher priority queues decreases.

Available commands:

- Define a priority queue:

  ```
  set traffic-policy priority-queue <policy name>
  ```

- Add a description:

  ```
  set traffic-policy priority-queue <policy name> description <description>
  ```

Traffic classes

- Define a traffic class, each class is a separate queue, range for class ID is 1…7, while 1 being the lowest priority:

  ```
  set traffic-policy priority-queue <policy name> class <class ID>
  ```

- Add a class description:

  ```
  set traffic-policy priority-queue <policy name> class <class ID>
  description <description>
  ```

- Set a queue length limit in packets, default 1000:

  ```
  set traffic-policy priority-queue <policy name> class <class ID>
  queue-limit <limit>
  ```

- Specify a queue type for a traffic class, available queue types:
  - drop-tail
  - fair-queue
  - random-detect

```
set traffic-policy priority-queue <policy name> class <class
ID> queue-type <type>
```

Default class

- Define a default priority queue:

  ```
  set traffic-policy priority-queue <policy name> default
  ```

- Define a maximum queue length for the default traffic class in packets:

  ```
  set traffic-policy priority-queue <policy name> default queue-limit
  <limit>
  ```

- Specify the queuing type for the default traffic class, available queue types:
  - drop-tail
  - fair-queue
  - random-detect

    ```
    set traffic-policy priority-queue <policy name> default
    queue-type <type>
    ```

Matching rules

- Define a class matching rule:

  ```
  set traffic-policy priority-queue <policy name> class <class ID> match
  <match name>
  ```

- Add a match rule description:

  ```
  set traffic-policy priority-queue <policy name> class <class ID> match
  <match name> description <description>
  ```

- Specify a match criterion based on a **destination MAC address** (format: xx:xx:xx:xx:xx:xx):

  ```
  set traffic-policy priority-queue <policy name> class <class ID> match
  <match name> ether destination <MAC address>
  ```

- Specify a match criterion based on a **source MAC address** (format: xx:xx:xx:xx:xx:xx):

  ```
  set traffic-policy priority-queue <policy name> class <class ID> match
  <match name> ether source <MAC address>
  ```

- Specify a match criterion based on **packet type/protocol**, range 0. . . 65535:

  ```
  set traffic-policy priority-queue <policy name> class <class ID> match
  <match name> ether protocol <number>
  ```

- Specify a match criterion based on **ingress interface**:

  ```
  set traffic-policy priority-queue <policy name> class <class ID> match
  <match name> interface <interface>
  ```

- Specify a match criterion based on the **fwmark field**, range 0. . . .4294967295:

  ```
  set traffic-policy priority-queue <policy name> class <class ID> match
  <match name> mark <fwmark>
  ```

- Specify a match criterion based on **VLAN ID**, range 1…4096:

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> vif <VLAN ID>
```

IPv4

- Specify a match criterion based on **destination IPv4 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ip destination <IPv4 address|port>
```

- Specify a match criterion based on **source IPv4 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ip source <IPv4 address|port>
```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ip dscp <DSCP value>
```

- Specify a match criterion based on **IPv4 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ip protocol <proto>
```

IPv6

- Specify a match criterion based on **destination IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ipv6 destination <IPv6 address|port>
```

- Specify a match criterion based on **source IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ipv6 source <IPv6 address|port>
```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ipv6 dscp <DSCP value>
```

- Specify a match criterion based on **IPv6 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

```
set traffic-policy priority-queue <policy name> class <class ID> match
<match name> ipv6 protocol <proto>
```

11.2.6 Random Early Detection (RED/WRED)

RED

A Random Early Detection (RED) policy starts randomly dropping packets from a queue before it reaches its queue limit thus avoiding congestion. It is also beneficial for TCP connections as the gradual dropping of packets acts as a signal for the sender to decrease its transmission rate, avoiding global TCP synchronisation. Applicable to outbound traffic only.

Available commands:

- Define a RED policy:

  ```
  set traffic-policy random-detect <policy name>
  ```

- Add a description:

  ```
  set traffic-policy random-detect <policy name> description <description>
  ```

- Set a bandwidth limit, default auto:

  ```
  set traffic-policy random-detect <policy name> bandwidth <rate>
  ```

  Available suffixes:</u>

- auto (bandwidth limit based on interface speed, default)
- kbit (kilobits per second)
- mbit (megabits per second)
- gbit (gigabits per second)
- kbps (kilobytes per second)
- mbps (megabytes per second)
- gbps (gigabytes per second)

WRED

In contrast to RED, Weighted Random Early Detection (WRED) differentiates between classes of traffic in a single queue and assigns different precedence to traffic flows accordingly; low priority packets are dropped from a queue earlier than high priority packets. This is achieved by using the first three bits of the ToS (Type of Service) field to categorise data streams and in accordance with the defined precedence parameters a decision is made. A WRED policy is defined with the following parameters:

- precedence
- min-threshold
- max-threshold
- average-packet
- mark-probability
- queue-limit

If the average queue size is lower than the `min-threshold`, an arriving packet is placed in the queue. If the average queue size is between `min-threshold` and `max-threshold` an arriving packet is either dropped or placed in the queue depending on the defined `mark-probability`. In case the average queue size is larger than `max-threshold`, packets are dropped. If the current queue size is larger than `queue-limit`, packets are dropped. The average queue size depends on its former average size and its current size. If `max-threshold` is set but `min-threshold` is not, then `min-threshold` is scaled to 50% of `max-threshold`. In principle, values must be `min-threshold` < `max-threshold` < `queue-limit`. Applicable to outbound traffic only.

Possible values for WRED parameters:

- precedence - IP precedence, first three bits of the ToS field as defined in RFC791.

Precedence	Priority
7	Network Control
6	Internetwork Control
5	CRITIC/ECP
4	Flash Override
3	Flash
2	Immediate
1	Priority
0	Routine

- min-threshold - Min value for the average queue length, packets are dropped if the average queue length reaches this threshold. Range 0…4096, default is dependent on precedence:

Precedence	default min-threshold
7	16
6	15
5	14
4	13
3	12
2	11
1	10
0	9

- max-threshold - Max value for the average queue length, packets are dropped if this value is exceeded. Range 0…4096 packets, default 18.
- average-packet - Average packet size in bytes, default 1024.
- mark-probability - The fraction of packets (n/probability) dropped from the queue when the average queue length reaches <code>max-threshold</code>, default 10.
- queue-limit - Packets are dropped when the current queue length reaches this value, default 4*<code>max-threshold</code>.

Usage:

```
set traffic-policy random-detect <policy-name> precedence <precedence>
[average-packet <bytes> | mark-probability <probability> | max-threshold <max>
| min-threshold <min> | queue-limit <packets>]
```

11.2.7 Rate control (TBF)

The rate control policy uses the Token Bucket Filter (TBF) algorithm to limit the packet flow to a set rate. Short bursts can be allowed to exceed the limit. Applicable to outbound traffic only.

Available commands:

- Define a rate control policy:

  ```
  set traffic-policy rate-control <policy-name>
  ```

- Add a description:

  ```
  set traffic-policy rate-control <policy-name> description <description>
  ```

- Specify a bandwidth limit in kbits/s:

  ```
  set traffic-policy rate-control <policy-name> bandwidth <rate>
  ```

  Available suffixes:</u>
- kbit (kilobits per second, default)
- mbit (megabits per second)
- gbit (gigabits per second)
- kbps (kilobytes per second)
- mbps (megabytes per second)
- gbps (gigabytes per second)
- Specify a burst size in bytes, default 15 kilobytes:

  ```
  set traffic-policy rate-control <policy-name> burst <burst-size>
  ```

  Available suffixes:
- kb (kilobytes)
- mb (megabytes)
- gb (gigabytes)
- Specify a latency in milliseconds; the maximum amount of time packets are allowed to wait in the queue, default 50 milliseconds:

  ```
  set traffic-policy rate-control <policy-name> latency
  ```

  Available suffixes:
- secs (seconds)
- ms (milliseconds, default)
- us (microseconds)

11.2.8 Round robin (DRR)

The round robin policy divides available bandwidth between all defined traffic classes.

Available commands:

- Define a round robin policy:

  ```
  set traffic-policy round-robin <policy-name>
  ```
- Add a description:

  ```
  set traffic-policy round-robin <policy-name> description <description>
  ```
- Define a traffic class ID, range 2…4095:

  ```
  set traffic-policy round-robin <policy-name> class <class>
  ```

Default policy:

- Define a default priority queue:

  ```
  set traffic-policy round-robin <policy name> default
  ```

- Set the number of packets that can be sent per scheduling quantum:

  ```
  set traffic-policy round-robin <policy name> default quantum <packets>
  ```

- Define a maximum queue lenght for the default policy in packets:

  ```
  set traffic-policy round-robin <policy name> default queue-limit <limit>
  ```

- Specify the queuing type for the default policy, available queue types:
  - drop-tail
  - fair-queue
  - priority (based on the DSCP values in the ToS byte)

    ```
    set traffic-policy round-robin <policy name> default
    queue-type <type>
    ```

Matching rules

- Define a class matching rule:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name>
  ```

- Add a match rule description:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name> description <description>
  ```

- Specify a match criterion based on a **destination MAC address** (format: xx:xx:xx:xx:xx:xx):

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name> ether destination <MAC address>
  ```

- Specify a match criterion based on a **source MAC address** (format: xx:xx:xx:xx:xx:xx):

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name> ether source <MAC address>
  ```

- Specify a match criterion based on **packet type/protocol**, range 0. . . 65535:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name> ether protocol <number>
  ```

- Specify a match criterion based on **ingress interface**:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name> interface <interface>
  ```

- Specify a match criterion based on the **fwmark field**, range 0. . . .4294967295:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name> mark <fwmark>
  ```

- Specify a match criterion based on **VLAN ID**, range 1. . . 4096:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match
  name> vif <VLAN ID>*
  ```

IPv4

- Specify a match criterion based on **destination IPv4 address and/or port**, port may be specified as number or service name (i.e. ssh):

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ip destination <IPv4 address|port>
  ```

- Specify a match criterion based on **source IPv4 address and/or port**, port may be specified as number or service name (i.e. ssh):

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ip source <IPv4 address|port>
  ```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ip dscp <DSCP value>
  ```

- Specify a match criterion based on **IPv4 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ip protocol <proto>
  ```

IPv6

- Specify a match criterion based on **destination IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ipv6 destination <IPv6 address|port>
  ```

- Specify a match criterion based on **source IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ipv6 source <IPv6 address|port>
  ```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ipv6 dscp <DSCP value>
  ```

- Specify a match criterion based on **IPv6 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

  ```
  set traffic-policy round-robin <policy name> class <class ID> match <match name> ipv6 protocol <proto>
  ```

11.2.9 Traffic shaper

The shaper policy uses the Hierarchical Token Bucket algorithm to allocate different amounts of bandwidth to different traffic classes. In contrast to round robin, shaper limits bandwidth allocation by traffic class whereas round robin divides the total available bandwidth between classes.

Avialable commands:

- Define a shaper policy:

  ```
  set traffic-policy shaper <policy-name>
  ```

- Add a description:

  ```
  set traffic-policy shaper <policy-name> description <description>
  ```

- Set the available bandwidth for all combined traffic of this policy in kbit/s, default 100%:

  ```
  set traffic-policy shaper <policy-name> bandwidth <rate>
  ```

  Available suffixes:

- % (percentage of total bandwidth)
- kbit (kilobits per second)
- mbit (megabits per second)
- gbit (gigabits per second)
- kbps (kilobytes per second)
- mbps (megabytes per second)
- gbps (gigabytes per second)

Traffic classes

- Define a traffic class for a shaper policy, range for class ID is 2…4095:

  ```
  set traffic-policy shaper <policy-name> class <class ID>
  ```

- Add a class description:

  ```
  set traffic-policy shaper <policy name> class <class ID> description
  <description>
  ```

- Specify a bandwidth limit for a class, in kbit/s:

  ```
  set traffic-policy shaper <policy-name> class <class ID> bandwidth <rate>
  ```

  Available suffixes:

- kbit (kilobits per second, default)
- mbit (megabits per second)
- gbit (gigabits per second)
- kbps (kilobytes per second)
- mbps (megabytes per second)
- gbps (gigabytes per second)
- Set a burst size for a class, the maximum amount of traffic that can be sent, in bytes:

  ```
  set traffic-policy shaper <policy-name> class <class ID> burst
  <burst-size>
  ```

  Available suffixes:

- kb (kilobytes)
- mb (megabytes)
- gb (gigabytes)

- Set a bandwidth ceiling for a class in kbit/s:

  ```
  set traffic-policy shaper <policy-name> class <class ID> ceiling <rate>
  ```

  Available suffixes:

- % (percentage of total bandwidth)
- kbit (kilobits per second)
- mbit (megabits per second)
- gbit (gigabits per second)
- Set the priority of a class for allocation of additional bandwidth, if unused bandwidth is available. Range 0…7, lowest number has lowest priority, default 0:

  ```
  set traffic-policy shaper <policy-name> class <class ID> priority
  <priority>
  ```

- Set a queue length limit in packets:

  ```
  set traffic-policy shaper <policy name> class <class ID> queue-limit
  <limit>
  ```

- Specify a queue type for a traffic class, default fair-queue. Available queue types:
  - drop-tail
  - fair-queue
  - random-detect
  - priority

    ```
    set traffic-policy shaper <policy name> class <class ID>
    queue-type <type>
    ```

- Modify DSCP field; the DSCP field value of packets in a class can be rewritten to change the forwarding behaviour and allow for traffic conditioning:

  ```
  set traffic-policy shaper <policy name> class <class ID> set-dscp <value>
  ```

  DSCP values as per RFC2474 and RFC4595:

Binary value	Configured value	Drop rate	Description
101110	46	–	Expedited forwarding (EF)
000000	0	–	Best effort traffic, default
001010	10	Low	Assured Forwarding(AF) 11
001100	12	Medium	Assured Forwarding(AF) 12
001110	14	High	Assured Forwarding(AF) 13
010010	18	Low	Assured Forwarding(AF) 21
010100	20	Medium	Assured Forwarding(AF) 22
010110	22	High	Assured Forwarding(AF) 23
011010	26	Low	Assured Forwarding(AF) 31
011100	28	Medium	Assured Forwarding(AF) 32
011110	30	High	Assured Forwarding(AF) 33
100010	34	Low	Assured Forwarding(AF) 41
100100	36	Medium	Assured Forwarding(AF) 42
100110	38	High	Assured Forwarding(AF) 43

Matching rules

- Define a class matching rule:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name>
```

- Add a match rule description:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> description <description>
```

- Specify a match criterion based on a **destination MAC address** (format: xx:xx:xx:xx:xx:xx):

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ether destination <MAC address>
```

- Specify a match criterion based on a **source MAC address** (format: xx:xx:xx:xx:xx:xx):

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ether source <MAC address>
```

- Specify a match criterion based on **packet type/protocol**, range 0. . . 65535:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ether protocol <number>
```

- Specify a match criterion based on **ingress interface**:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> interface <interface>
```

- Specify a match criterion based on the **fwmark field**, range 0. . . .4294967295:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> mark <fwmark>
```

- Specify a match criterion based on **VLAN ID**, range 1. . . 4096:

```
set traffic-policy round-robin <policy name> class <class ID> match <match
name> vif <VLAN ID>
```

IPv4

- Specify a match criterion based on **destination IPv4 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ip destination <IPv4 address|port>
```

- Specify a match criterion based on **source IPv4 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ip source <IPv4 address|port>
```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ip dscp <DSCP value>
```

- Specify a match criterion based on **IPv4 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ip protocol <proto>
```

IPv6

- Specify a match criterion based on **destination IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ipv6 destination <IPv6 address|port>
```

- Specify a match criterion based on **source IPv6 address and/or port**, port may be specified as number or service name (i.e. ssh):

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ipv6 source <IPv6 address|port>
```

- Specify a match criterion based on **DSCP (Differentiated Services Code Point) value**, DSCP value may be specified as decimal or hexadecimal number:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ipv6 dscp <DSCP value>
```

- Specify a match criterion based on **IPv6 protocol**, protocol may be specified by name (i.e. icmp) or IANA-assigned number:

```
set traffic-policy shaper <policy name> class <class ID> match <match
name> ipv6 protocol <proto>
```

11.2.10 shaper-hfsc (HFSC + sfq)

TBD

11.3 Ingress shaping

The case of ingress shaping. Only a **limiter** policy can be applied directly for ingress traffic on an interface. It is possible though to use what is called an Intermediate Functional Block (IFB) to allow the usage of any policy on the ingress traffic.

Let's assume eth0 is your WAN link. You created two traffic-policies: *WAN-IN* and *WAN-OUT*.

Steps to do:

- First, create the IFB:

  ```
  set interfaces input ifb0 description "WAN Input"
  ```

- Apply the *WAN-IN* traffic-policy to ifb0 input.

  ```
  set interfaces input ifb0 traffic-policy out WAN-IN
  ```

- Redirect traffic from eth0 to ifb0

  ```
  set interfaces ethernet eth0 redirect ifb0
  ```

11.4 Classful policies and traffic matching

limiter, *round-robin*, *priority-queue*, *shaper* and *shaper-hfsc* distribute traffic into different classes with different options. In VyOS, classes are numbered and work like firewall rules. e.g:

```
set traffic-policy shaper SHAPER class 30
```

11.4.1 Matching traffic

A class can have multiple match filters:

```
set traffic-policy <POLICY> <POLICY-NAME> class N match MATCH-FILTER-NAME
```

Example:

```
set traffic-policy shaper SHAPER class 30 match HTTP
set traffic-policy shaper SHAPER class 30 match HTTPs
```

A match filter contains multiple criteria and will match traffic if all those criteria are true.

For example:

```
set traffic-policy shaper SHAPER class 30 match HTTP ip protocol tcp
set traffic-policy shaper SHAPER class 30 match HTTP ip source port 80
```

This will match tcp traffic with source port 80.

description

```
set traffic-policy shaper SHAPER class 30 match MATCH description "match filter␣
↪description"
```

ether

```
edit traffic-policy shaper SHAPER class 30 match MATCH ether
```

destination

protocol

source

interface

```
edit traffic-policy shaper SHAPER class 30 match MATCH interface <interface-name>
```

ip

```
edit traffic-policy shaper SHAPER class 30 match MATCH ip
```

destination

```
set destination address IPv4-SUBNET
set destination port U32-PORT
```

dscp

```
set dscp DSCPVALUE
```

max-length

```
set max-length U32-MAXLEN
```

Will match ipv4 packets with a total length lesser than set value.

protocol

```
set protocol <IP PROTOCOL>
```

source

```
set source address IPv4-SUBNET
set source port U32-PORT
```

tcp

Note: You must set ip protocol to TCP to use the TCP filters.

Note: This filter will only match packets with an IPv4 header length of 20 bytes (which is the majority of IPv4 packets anyway).

```
set tcp ack
```

Will match tcp packets with ACK flag set.

```
set tcp syn
```

Will match tcp packets with SYN flag set.

ipv6

```
edit traffic-policy shaper SHAPER class 30 match MATCH ipv6
```

destination

```
set destination address IPv6-SUBNET
set destination port U32-PORT
```

dscp

```
set dscp DSCPVALUE
```

max-length

```
set max-length U32-MAXLEN
```

Will match ipv6 packets with a payload length lesser than set value.

protocol

```
set protocol IPPROTOCOL
```

source

```
set source address IPv6-SUBNET
set source port U32-PORT
```

tcp

Note: You must set ipv6 protocol to TCP to use the TCP filters.

Note: This filter will only match IPv6 packets with no header extension, see http://en.wikipedia.org/wiki/IPv6_packet#Extension_headers for no header extension.

```
set tcp ack
```

Will match tcp packets with ACK flag set.

```
set tcp syn
```

Will match tcp packets with SYN flag set.

mark

```
set traffic-policy shaper SHAPER class 30 match MATCH mark **firewall-mark**
```

vif

```
set traffic-policy shaper SHAPER class 30 match MATCH vif **vlan-tag**
```

```
set interfaces ethernet eth0 traffic-policy out 'WAN-OUT'
set interfaces ethernet eth1 traffic-policy out 'LAN-OUT'
```

CHAPTER 12

Services

This chapter descriptes the available system/network services provided by VyOS.

12.1 Conntrack

One of the important features built on top of the Netfilter framework is connection tracking. Connection tracking allows the kernel to keep track of all logical network connections or sessions, and thereby relate all of the packets which may make up that connection. NAT relies on this information to translate all related packets in the same way, and iptables can use this information to act as a stateful firewall.

The connection state however is completely independent of any upper-level state, such as TCP's or SCTP's state. Part of the reason for this is that when merely forwarding packets, i.e. no local delivery, the TCP engine may not necessarily be invoked at all. Even connectionless-mode transmissions such as UDP, IPsec (AH/ESP), GRE and other tunneling protocols have, at least, a pseudo connection state. The heuristic for such protocols is often based upon a preset timeout value for inactivity, after whose expiration a Netfilter connection is dropped.

Each Netfilter connection is uniquely identified by a (layer-3 protocol, source address, destination address, layer-4 protocol, layer-4 key) tuple. The layer-4 key depends on the transport protocol; for TCP/UDP it is the port numbers, for tunnels it can be their tunnel ID, but otherwise is just zero, as if it were not part of the tuple. To be able to inspect the TCP port in all cases, packets will be mandatorily defragmented.

12.1.1 Configuration

```
# Protocols only for which local conntrack entries will be synced (tcp, udp, icmp,␣
↪sctp)
set service conntrack-sync accept-protocol

# Queue size for listening to local conntrack events (in MB)
set service conntrack-sync event-listen-queue-size <int>

# Protocol for which expect entries need to be synchronized. (all, ftp, h323, nfs,␣
↪sip, sqlnet)
```

(continues on next page)

(continued from previous page)

```
set service conntrack-sync expect-sync

# Failover mechanism to use for conntrack-sync [REQUIRED]
set service conntrack-sync failover-mechanism

set service conntrack-sync cluster group <string>
set service conntrack-sync vrrp sync-group <1-255>

# IP addresses for which local conntrack entries will not be synced
set service conntrack-sync ignore-address ipv4 <x.x.x.x>

# Interface to use for syncing conntrack entries [REQUIRED]
set service conntrack-sync interface <ifname>

# Multicast group to use for syncing conntrack entries
set service conntrack-sync mcast-group <x.x.x.x>

# Queue size for syncing conntrack entries (in MB)
set service conntrack-sync sync-queue-size <size>
```

12.1.2 Example

The next exemple is a simple configuration of conntrack-sync.

Fig. 1: Conntrack Sync Example

First of all, make sure conntrack is enabled by running

```
show conntrack table ipv4
```

If the table is empty and you have a warning message, it means conntrack is not enabled. To enable conntrack, just create a NAT or a firewall rule.

```
set firewall state-policy established action accept
```

You now should have a conntrack table

```
$ show conntrack table ipv4
TCP state codes: SS - SYN SENT, SR - SYN RECEIVED, ES - ESTABLISHED,
                 FW - FIN WAIT, CW - CLOSE WAIT, LA - LAST ACK,
                 TW - TIME WAIT, CL - CLOSE, LI - LISTEN

CONN ID    Source                 Destination            Protocol         TIMEOUT
1015736576 10.35.100.87:58172     172.31.20.12:22        tcp [6] ES       430279
1006235648 10.35.101.221:57483    172.31.120.21:22       tcp [6] ES       413310
1006237088 10.100.68.100          172.31.120.21          icmp [1]         29
1015734848 10.35.100.87:56282     172.31.20.12:22        tcp [6] ES       300
1015734272 172.31.20.12:60286     239.10.10.14:694       udp [17]         29
1006239392 10.35.101.221          172.31.120.21          icmp [1]         29
```

Now configure conntrack-sync service on `router1` **and** `router2`

```
set service conntrack-sync accept-protocol 'tcp,udp,icmp'
set service conntrack-sync event-listen-queue-size '8'
set service conntrack-sync failover-mechanism cluster group 'GROUP' # Or VRRP
set service conntrack-sync interface 'eth0'
set service conntrack-sync mcast-group '225.0.0.50'
set service conntrack-sync sync-queue-size '8'
```

On the active router, you should have informations in the internal-cache of conntrack-sync. The same current active connections number should be shown in the external-cache of the standby router

On active router run:

```
$ show conntrack-sync statistics

Main Table Statistics:

cache internal:
current active connections:              10
connections created:                   8517    failed:            0
connections updated:                    127    failed:            0
connections destroyed:                 8507    failed:            0

cache external:
current active connections:               0
connections created:                      0    failed:            0
connections updated:                      0    failed:            0
connections destroyed:                    0    failed:            0

traffic processed:
                   0 Bytes                          0 Pckts

multicast traffic (active device=eth0):
              868780 Bytes sent               224136 Bytes recv
```

(continues on next page)

(continued from previous page)

```
               20595 Pckts sent                14034 Pckts recv
                   0 Error send                   0 Error recv

message tracking:
                   0 Malformed msgs                   0 Lost msgs



On standby router run:


$ show conntrack-sync statistics

Main Table Statistics:

cache internal:
current active connections:                0
connections created:                       0    failed:            0
connections updated:                       0    failed:            0
connections destroyed:                     0    failed:            0

cache external:
current active connections:               10
connections created:                     888    failed:            0
connections updated:                     134    failed:            0
connections destroyed:                   878    failed:            0

traffic processed:
                   0 Bytes                        0 Pckts

multicast traffic (active device=eth0):
              234184 Bytes sent               907504 Bytes recv
               14663 Pckts sent                21495 Pckts recv
                   0 Error send                   0 Error recv

message tracking:
                   0 Malformed msgs                   0 Lost msgs
```

12.2 DHCP Server

Multiple DHCP Servers can be run from a single machine. Each DHCP service is identified by a *shared-network-name*.

12.2.1 DHCP Server Example

In this example, we are offering address space in the 172.16.17.0/24 network, which is on eth1, and pppoe0 is our connection to the internet. We are using the network name *dhcpexample*.

12.2.2 Prerequisites

Configuring the PPPoE interface is assumed to be done already, and appears on *pppoe0*

12.2.3 Interface Configuration

```
set interface eth1 address 172.16.17.1/24
```

Multiple ranges can be defined and can contain holes.

```
set service dhcp-server shared-network-name dhcpexample authoritative
set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 default-
↪router 172.16.17.1
set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 dns-
↪server 172.16.17.1
set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 lease␣
↪86400
set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 range 0␣
↪start 172.16.17.100
set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 range 0␣
↪stop 172.16.17.199
```

12.2.4 Explanation

- `set service dhcp-server shared-network-name dhcpexample authoritative`

  This says that this device is the only DHCP server for this network. If other devices are trying to offer DHCP leases, this machine will send 'DHCPNAK' to any device trying to request an IP address that is not valid for this network.

- `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 default-router 172.16.17.1`

  This is a configuration parameter for the subnet, saying that as part of the response, tell the client that I am the default router for this network

- `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 dns-server 172.16.17.1`

  This is a configuration parameter for the subnet, saying that as part of the response, tell the client that I am the DNS server for this network. If you do not want to run a DNS server, you could also provide one of the public DNS servers, such as google's. You can add multiple entries by repeating the line.

- `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 lease 86400`

  Assign the IP address to this machine for 24 hours. It is unlikely you'd need to shorten this period, unless you are running a network with lots of devices appearing and disappearing.

- `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 range 0 start 172.16.17.100`

  Make a range of addresses available for clients starting from .100 [...]

- `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 range 0 stop 172.16.17.199`

  [...] and ending at .199

12.2.5 Failover

VyOS provides support for DHCP failover:

```
set service dhcp-server shared-network-name 'LAN' subnet '192.168.0.0/24' failover␣
↪local-address '192.168.0.1'
set service dhcp-server shared-network-name 'LAN' subnet '192.168.0.0/24' failover␣
↪name 'foo'
set service dhcp-server shared-network-name 'LAN' subnet '192.168.0.0/24' failover␣
↪peer-address '192.168.0.2'
```

Note: *name* must be identical on both sides!

The primary and secondary statements determines whether the server is primary or secondary

```
set service dhcp-server shared-network-name 'LAN' subnet '192.168.0.0/24' failover␣
↪status 'primary'
```

or

```
set service dhcp-server shared-network-name 'LAN' subnet '192.168.0.0/24' failover␣
↪status 'secondary'
```

Note: In order for the primary and the secondary DHCP server to keep their lease tables in sync, they must be able to reach each other on TCP port 647. If you have firewall rules in effect, adjust them accordingly.

12.2.6 Static mappings MAC/IP

```
set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 static-
↪mapping static-mapping-01 ip-address 172.16.17.10
set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 static-
↪mapping static-mapping-01 mac-address ff:ff:ff:ff:ff:ff
```

12.2.7 DHCP server options

default-router (DHCP option 003) `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 default-router <ROUTER-IP>`

dns-server (DHCP option 006) `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 dns-server <DNS-SERVER-IP>`

domain-name Client domain name (DHCP option 015) `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 domain-name "<DOMAIN-NAME>"`

domain-search (DHCP option 119) This option can be given multiple times if you need multiple search domains `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 domain-search "<DOMAIN_NAME_1>"` `set service dhcp-server shared-network-name dhcpexample subnet 172.16.17.0/24 domain-search "<DOMAIN_NAME_2>"`

12.3 DHCPv6 server

VyOS provides DHCPv6 server functionality which is described in this section. In order to use the DHCPv6 server it has to be enabled first:

```
set service dhcpv6-server
```

To restart the DHCPv6 server (operational mode):

```
restart dhcpv6 server
```

To show the current status of the DHCPv6 server use:

```
show dhcpv6 server status
```

Show statuses of all assigned leases:

```
show dhcpv6 server leases
```

12.3.1 DHCPv6 server options

DHCPv6 server preference value

Clients receiving advertise messages from multiple servers choose the server with the highest preference value. The range for this value is *0. . . 255*. Set a preference value for the DHCPv6 server:

```
set service dhcpv6-server preference <preference value>
```

Delete a preference:

```
set service dhcpv6-server preference
```

Show current preference:

```
show service dhcpv6-server preference
```

Specify address lease time

The default lease time for DHCPv6 leases is 24 hours. This can be changed by supplying a *default-time*, *maximum-time* and *minimum-time* (all values in seconds):

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> lease-time
↪{default <default-time> | maximum <maximum-time> | minimum <minimum-time>}
```

Reset the custom lease times:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> lease-time
↪{default | maximum | minimum}
```

Show the current configuration:

```
show service dhcpv6-server shared-network-name <name> subnet <ipv6net> lease-time
↪{default | maximum | minimum}
```

Specify NIS domain

A Network Information (NIS) domain can be set to be used for DHCPv6 clients:

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-domain <nis-
↪domain-name>
```

To Delete the NIS domain:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-domain
↪<nis-domain-name>
```

Show a configured NIS domain:

```
show service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-domain
↪<nis-domain-name>
```

Specify NIS+ domain

The procedure to specify a Network Information Service Plus (NIS+) domain is similar to the NIS domain one:

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-domain
↪<nisplus-domain-name>
```

To Delete the NIS+ domain:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-
↪domain <nisplus-domain-name>
```

Show a configured NIS domain:

> # show service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-domain <nisplus-domain-name>

Specify NIS server address

To specify a NIS server address for DHCPv6 clients:

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-server
↪<IPv6 address>
```

Delete a specified NIS server address:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-server
↪<IPv6 address>
```

Show specified NIS server addresses:

```
show service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-server
```

Specify NIS+ server address

To specify a NIS+ server address for DHCPv6 clients:

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-server
↪<IPv6 address>
```

Delete a specified NIS+ server address:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-
↪server <IPv6 address>
```

Show specified NIS+ server addresses:

```
show service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-server
```

Specify a SIP server address for DHCPv6 clients

By IPv6 address

A Session Initiation Protocol (SIP) server address can be specified for DHCPv6 clients:

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-
↪address <IPv6 address>
```

Delete a specified SIP server address:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-
↪address <IPv6 address>
```

Show specified SIP server addresses:

```
show service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-
↪address
```

By FQDN

A name for SIP server can be specified:

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-name
↪<sip-server-name>
```

Delete a specified SIP server name:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-
↪name <sip-server-name>
```

Show specified SIP server names:

```
show service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-name
```

Simple Network Time Protocol (SNTP) server address for DHCPv6 clients

A SNTP server address can be specified for DHCPv6 clients:

```
set service dhcpv6-server shared-network-name <name> subnet <ipv6net> sntp-server-
↪address <IPv6 address>
```

Delete a specified SNTP server address:

```
delete service dhcpv6-server shared-network-name <name> subnet <ipv6net> sntp-server-
↪address <IPv6 address>
```

Show specified SNTP server addresses:

```
show service dhcpv6-server shared-network-name <name> subnet <ipv6net> sntp-server-
↪address
```

12.3.2 DHCPv6 address pools

DHCPv6 address pools must be configured for the system to act as a DHCPv6 server. The following example describes a common scenario.

Example 1: DHCPv6 address pool

A shared network named *NET1* serves subnet *2001:db8:100::/64* which is connected to *eth1*, a DNS server at *2001:db8:111::111* is used for name services. The range of the address pool shall be *::100* through *::199*. The lease time will be left at the default value which is 24 hours.

```
set service dhcpv6-server shared-network-name NET1 subnet 2001:db8:100::/64 address-
↪range start 2001:db8:100::100 stop 2001:db8:100::199
set service dhcpv6-server shared-network-name NET1 subnet 2001:db8:100::/64 name-
↪server 2001:db8:111::111
```

Commit the changes and show the configuration:

```
commit
show service dhcpv6-server
    shared-network-name NET1 {
        subnet 2001:db8:100::/64 {
           address-range {
              start 2001:db8:100::100 {
                 stop 2001:db8:100::199
              }
           }
           name-server 2001:db8:111::111
        }
    }
```

12.3.3 Static mappings

In order to map specific IPv6 addresses to specific hosts static mappings can be created. The following example explains the process.

Example 1: Static IPv6 MAC-based mapping

IPv6 address *2001:db8:100::101* shall be statically mapped to a device with MAC address *00:15:c5:b7:5e:23*, this host-specific mapping shall be named *client1*.

Note: The MAC address identifier is defined by the last 4 byte of the MAC address.

```
set service dhcpv6-server shared-network-name NET1 subnet 2001:db8:100::/64 static-
↪mapping client1 ipv6-address 2001:db8:100::101
set service dhcpv6-server shared-network-name NET1 subnet 2001:db8:100::/64 static-
↪mapping client1 identifier c5b75e23
```

Commit the changes and show the configuration:

```
show service dhcp-server shared-network-name NET1
   shared-network-name NET1 {
       subnet 2001:db8:100::/64 {
          name-server 2001:db8:111::111
          address-range {
              start 2001:db8:100::100 {
                 stop 2001:db8:100::199 {
              }
          }
          static-mapping client1 {
             ipv6-address 2001:db8:100::101
             identifier c5b75e23
          }
       }
    }
```

12.4 DHCP Relay

If you want your router to forward DHCP requests to an external DHCP server you can configure the system to act as a DHCP relay agent. The DHCP relay agent works with IPv4 and IPv6 addresses.

All interfaces used for the DHCP relay must be configured. See https://wiki.vyos.net/wiki/Network_address_setup.

12.4.1 DHCP relay example

In this example the interfaces used for the DHCP relay are eth1 and eth2. The router receives DHCP client requests on eth1 and relays them through eth2 to the DHCP server at 10.0.1.4.

12.4.2 Configuration

Enable DHCP relay for eth1 and eth2:

```
set service dhcp-relay interface eth1
set service dhcp-relay interface eth2
```

Set the IP address of the DHCP server:

Fig. 2: DHCP relay example

```
set service dhcp-relay server 10.0.1.4
```

The router should discard DHCP packages already containing relay agent information to ensure that only requests from DHCP clients are forwarded:

```
set service dhcp-relay relay-options relay-agents-packets discard
```

Commit the changes and show the results:

```
commit
show service dhcp-relay
    interface eth1
    interface eth2
    server 10.0.1.4
    relay-options {
       relay-agents-packets discard
    }
```

The DHCP relay agent can be restarted with:

```
restart dhcp relay-agent
```

12.4.3 DHCPv6 relay example

In this example DHCPv6 requests are received by the router on eth1 (*listening interface*) and forwarded through eth2 (*upstream interface*) to the external DHCPv6 server at 2001:db8:100::4.

Configuration

Set eth1 to be the listening interface for the DHCPv6 relay:

```
set service dhcpv6-relay listen-interface eth1
```

Set eth2 to be the upstream interface and specify the IPv6 address of the DHCPv6 server:

Fig. 3: DHCPv6 relay example

```
set service dhcpv6-relay upstream-interface eth2 address 2001:db8:100::4
```

Commit the changes and show results:

```
commit
show service dhcpv6-relay
    listen-interface eth1 {
    }
    upstream-interface eth2 {
       address 2001:db8:100::4
    }
```

Show the current status of the DHCPv6 relay agent:

```
show dhcpv6 relay-agent status
```

The DHCPv6 relay agent can be restarted with:

```
restart dhcpv6 relay-agent
```

12.4.4 Additional parameters

DHCP relay agent options

Set the maximum hop count before packets are discarded. Range 0. . . 255, default 10.

- `set service dhcp-relay relay-options hop-count 'count'`

Set maximum size of DHCP packets including relay agent information. If a DHCP packet size surpasses this value it will be forwarded without appending relay agent information. Range 64. . . 1400, default 576.

- `set service dhcp-relay relay-options max-size 'size'`

Four policies for reforwarding DHCP packets exist:

- **append:** The relay agent is allowed to append its own relay information to a received DHCP packet, disregarding relay information already present in the packet.

- **discard:** Received packets which already contain relay information will be discarded.
- **forward:** All packets are forwarded, relay information already present will be ignored.
- **replace:** Relay information already present in a packet is stripped and replaced with the router's own relay information set.
- `set service dhcp-relay relay-options relay-agents-packet 'policy'`

DHCPv6 relay agent options

Set maximum hop count before packets are discarded. Default: 10.

- `set service dhcpv6-relay max-hop-count 'count'`

If this is set the relay agent will insert the interface ID. This option is set automatically if more than one listening interfaces are in use.

- `set service dhcpv6-relay use-interface-id-option`

12.5 DNS Forwarding

Use DNS forwarding if you want your router to function as a DNS server for the local network. There are several options, the easiest being 'forward all traffic to the system DNS server(s)' (defined with set system name-server):

```
set service dns forwarding system
```

Manually setting DNS servers for forwarding:

```
set service dns forwarding name-server 8.8.8.8
set service dns forwarding name-server 8.8.4.4
```

Manually setting DNS servers with IPv6 connectivity:

```
set service dns forwarding name-server 2001:4860:4860::8888
set service dns forwarding name-server 2001:4860:4860::8844
```

Setting a forwarding DNS server for a specific domain:

```
set service dns forwarding domain example.com server 192.0.2.1
```

12.5.1 Example 1

Router with two interfaces eth0 (WAN link) and eth1 (LAN). Split DNS for example.com.

- DNS request for a local domain (example.com) get forwarded to 192.0.2.1
- Other DNS requests are forwarded to Google's DNS servers.
- The IP address for the LAN interface is 192.168.0.1.

```
set service dns forwarding domain example.com server 192.0.2.1
set service dns forwarding name-server 8.8.8.8
set service dns forwarding name-server 8.8.4.4
set service dns forwarding listen-address 192.168.0.1
```

12.5.2 Example 2

Same as example 1 but with additional IPv6 addresses for Google's public DNS servers.

The IP addresses for the LAN interface are 192.168.0.1 and 2001:db8::1

```
set service dns forwarding domain example.com server 192.0.2.1
set service dns forwarding name-server 8.8.8.8
set service dns forwarding name-server 8.8.4.4
set service dns forwarding name-server 2001:4860:4860::8888
set service dns forwarding name-server 2001:4860:4860::8844
set service dns forwarding listen-address 2001:db8::1
set service dns forwarding listen-address 192.168.0.1
```

12.6 Dynamic DNS

VyOS is able to update a remote DNS record when an interface gets a new IP address. In order to do so, VyOS includes ddclient, a perl script written for this exact purpose.

ddclient uses two methods to update a DNS record. The first one will send updates directly to the DNS daemon, in compliance with RFC2136. The second one involves a third party service, like DynDNS.com or any other similar website. This method uses HTTP requests to transmit the new IP address. You can configure both in VyOS.

12.6.1 VyOS CLI and RFC2136

First, create an RFC2136 config node :

```
edit service dns dynamic interface eth0 rfc2136 <confignodename>
```

Present your RNDC key to ddclient :

```
set key /config/dyndns/mydnsserver.rndc.key
```

Set the DNS server IP/FQDN :

```
set server dns.mydomain.com
```

Set the NS zone to be updated :

```
set zone mydomain.com
```

Set the records to be updated :

```
set record dyn
set record dyn2
```

You can optionally set a TTL (note : default value is 600 seconds) :

```
set ttl 600
```

This will generate the following ddclient config blocks:

```
server=dns.mydomain.com
protocol=nsupdate
password=/config/dyndns/mydnsserver.rndc.key
ttl=600
zone=mydomain.com
dyn
server=dns.mydomain.com
protocol=nsupdate
password=/config/dyndns/mydnsserver.rndc.key
ttl=600
zone=mydomain.com
dyn2
```

You can also keep a different dns zone updated. Just create a new config node:

```
edit service dns dynamic interface eth0 rfc2136 <confignode2>
```

12.6.2 VyOS CLI and HTTP dynamic DNS services

VyOS is also able to use any service relying on protocols supported by ddclient.

To use such a service, you must define a login, a password, one or multiple hostnames, a protocol and a server.

```
edit service dns dynamic interface eth0 service HeNet
set login my-login # set password my-password
set host-name my-tunnel-id
set protocol dyndns2
set server ipv4.tunnelbroker.net
```

VyOS is also shipped with a list of known services. You don't need to set the protocol and server value as VyOS has defaults provided for those. These are the services VyOS knows about:

- afraid
- changeip
- dnspark
- dslreports
- dyndns
- easydns
- namecheap
- noip
- zoneedit

To use DynDNS for example:

```
edit service dns dynamic interface eth0 service dyndns
set login my-login
set password my-password
set host-name my-dyndns-hostname
```

It's possible to use multiple services :

```
edit service dns dynamic interface eth0 service dyndns
set login my-login
set password my-password
set host-name my-dyndns-hostname
edit service dns dynamic interface eth0 service HeNet
set login my-login
set password my-password
set host-name my-tunnel-id
set protocol dyndns2
set server ipv4.tunnelbroker.net
```

12.6.3 ddclient behind NAT

By default, ddclient will update a dynamic dns record using the IP address directly attached to the interface. If your VyOS instance is behind NAT, your record will be updated to point to your internal IP.

ddclient has another way to determine the WAN IP address. This is controlled by these two options:

```
set service dns dynamic interface eth0 use-web url
set service dns dynamic interface eth0 use-web skip
```

ddclient will load the webpage at *[url]* and will try to extract an IP address for the response. ddclient will skip any address located before the string set in *[skip]*.

12.7 LLDP

The Link Layer Discovery Protocol (LLDP) is a vendor-neutral link layer protocol in the Internet Protocol Suite used by network devices for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network, principally wired Ethernet.[1] The protocol is formally referred to by the IEEE as Station and Media Access Control Connectivity Discovery specified in IEEE 802.1AB and IEEE 802.3-2012 section 6 clause 79.

LLDP performs functions similar to several proprietary protocols, such as Cisco Discovery Protocol, Foundry Discovery Protocol, Nortel Discovery Protocol and Link Layer Topology Discovery.

12.7.1 Information gathered

Information gathered with LLDP is stored in the device as a management information database (MIB) and can be queried with the Simple Network Management Protocol (SNMP) as specified in RFC 2922. The topology of an LLDP-enabled network can be discovered by crawling the hosts and querying this database. Information that may be retrieved include:

- System name and description
- Port name and description
- VLAN name
- IP management address
- System capabilities (switching, routing, etc.)
- MAC/PHY information
- MDI power
- Link aggregation

12.7.2 Configuration

- Enable service with:

  `set service lldp`

Options

- Configure a Define management-address:

  `set service lldp management-address <x.x.x.x>`
- Define listening interfaces

  `set service lldp interface <all|interface name>`
- LLDPd also implements an SNMP subagent. To Enable SNMP queries of the LLDP database:

  `set service lldp snmp enable`
- Enable optional/other protocols

  `set service lldp legacy-protocols cdp`

  Supported legacy protocols:
- `cdp` - Listen for CDP for Cisco routers/switches
- `edp` - Listen for EDP for Extreme routers/switches
- `fdp` - Listen for FDP for Foundry routers/switches
- `sonmp` - Listen for SONMP for Nortel routers/switches

12.7.3 Display neighbors

- Display with:

`show lldp neighbors`

Exemple:

```
vyos@vyos:~# show lldp neighbors
Capability Codes: R - Router, B - Bridge, W - Wlan r - Repeater, S - Station
                 D - Docsis, T - Telephone, O - Other
Device ID                 Local  Proto  Cap   Platform             Port ID
---------                 -----  -----  ---   --------             -------
swA309                    eth0   LLDP   ?     Cisco IOS Software,  GigE0/33
```

- Options:
- `detail` - Show lldp neighbors detail
- `interface` - Show LLDP for specified interface

12.7.4 Troubleshooting

Use operationnal command `show log lldp` to display logs.

12.8 mDNS Repeater

Starting with VyOS 1.2 a Multicast DNS (mDNS) repeater functionality is provided.

Multicast DNS uses the 224.0.0.51 address, which is "administratively scoped" and does not leave the subnet. It re-broadcast mDNS packets from one interface to other interfaces. This enables support for e.g. Apple Airplay devices across multiple VLANs.

To enable mDNS repeater you need to configure at least two interfaces. To re- broadcast all mDNS packets from *eth0* to *eth1* and vice versa run:

```
set service mdns repeater interface eth0
set service mdns repeater interface eth1
```

mDNS repeater can be temporarily disabled without deleting the service using

```
set service mdns repeater disable
```

Note: You can not run this in a VRRP setup, if multiple mDNS repeaters are launched in a subnet you will experience the mDNS packet storm death!

12.9 IPoE server

VyOS utilizes accel-ppp to provide IPoE server functionality. It can be used with local authentication (mac-address) or a connected RADIUS server.

Note: Please be aware, due to an upstream bug, config changes/commits will restart the ppp daemon and will reset existing IPoE sessions, in order to become effective.

12.9.1 Configuration

IPoE can be configure on different interfaces, it will depend on each specific situation which interface will provide IPoE to clients. The clients mac address and the incoming interface is being used as control parameter, to authenticate a client.

The example configuration below will assign an IP to the client on the incoming interface eth2 with the client mac address 08:00:27:2f:d8:06. Other DHCP discovery requests will be ignored, unless the client mac has been enabled in the configuration.

```
set service ipoe-server authentication interface eth2 mac-address 08:00:27:2f:d8:06
set service ipoe-server authentication mode 'local'
set service ipoe-server dns-server server-1 '8.8.8.8'
set service ipoe-server dns-server server-2 '8.8.4.4'
set service ipoe-server interface eth2 client-subnet '192.168.0.0/24'
```

The first address of the parameter `client-subnet`, will be used as the default gateway. Connected sessions can be checked via the `show ipoe-server sessions` command.

```
vyos@vyos:~$ show ipoe-server sessions

ifname | called-sid |    calling-sid    |     ip      | ip6 | ip6-dp | rate-limit |␣
↪state  |  uptime  |        sid
-------+------------+-------------------+-------------+-----+--------+------------+---
↪-----+----------+------------------
ipoe0  | eth2       | 08:00:27:2f:d8:06 | 192.168.0.2 |     |        |            |␣
↪active | 00:45:05 | dccc870fd3134612
```

IPv6 SLAAC and IA-PD

To configure IPv6 assignments for clients, two options need to be configured. A global prefix which is terminated on the clients cpe and a delegated prefix, the client can use for devices routed via the clients cpe.

IPv6 DNS addresses are optional.

```
set service ipoe-server authentication interface eth3 mac-address 08:00:27:2F:D8:06
set service ipoe-server authentication mode 'local'
set service ipoe-server client-ipv6-pool delegate-prefix '2001:db8:1::/48,56'
set service ipoe-server client-ipv6-pool prefix '2001:db8::/48,64'
set service ipoe-server dnsv6-server server-1 '2001:db8::'
set service ipoe-server dnsv6-server server-2 '2001:db8:aaa::'
set service ipoe-server dnsv6-server server-3 '2001:db8:bbb::'
set service ipoe-server interface eth3 client-subnet '192.168.1.0/24'
```

```
vyos@ipoe-server# run sh ipoe-server sessions
ifname | called-sid |    calling-sid    |     ip      |                ip6          ␣
↪  | ip6-dp          | rate-limit | state  |  uptime  |        sid
-------+------------+-------------------+-------------+-------------------------------
↪--+----------------+------------+--------+----------+------------------
ipoe0  | eth3       | 08:00:27:2f:d8:06 | 192.168.1.2 | 2001:db8::a00:27ff:fe2f:d806/
↪64 | 2001:db8:1::/56 |            | active | 01:02:59 | 4626faf71b12cc25
```

The clients cpe can now communicate via IPv4 or IPv6. All devices behind `2001:db8::a00:27ff:fe2f:d806/64` can use addresses from `2001:db8:1::/56` and can globally communicate without the need of any NAT rules.

Automatic VLAN creation

To create VLANs per user during runtime, the following settings are required on a per interface basis. VLAN ID and VLAN range can be present in the configuration at the same time.

```
set service ipoe-server interface eth2 network vlan
set service ipoe-server interface eth2 vlan-id 100
set service ipoe-server interface eth2 vlan-id 200
set service ipoe-server interface eth2 vlan-range 1000-2000
set service ipoe-server interface eth2 vlan-range 2500-2700
```

12.9.2 RADIUS Setup

To use a RADIUS server for authentication and bandwidth-shaping, the following example configuration can be used.

```
set service ipoe-server authentication mode 'radius'
set service ipoe-server authentication radius-server 10.100.100.1 secret 'password'
```

12.9.3 Bandwidth Shaping

Bandwidth rate limits can be set for local users within the configuration or via RADIUS based attributes.

Bandwidth Shaping for local users

The rate-limit is set in kbit/sec.

```
set service ipoe-server authentication interface eth2 mac-address 08:00:27:2f:d8:06␣
↪rate-limit download '500'
set service ipoe-server authentication interface eth2 mac-address 08:00:27:2f:d8:06␣
↪rate-limit upload '500'
set service ipoe-server authentication mode 'local'
set service ipoe-server dns-server server-1 '8.8.8.8'
set service ipoe-server dns-server server-2 '8.8.4.4'
set service ipoe-server interface eth2 client-subnet '192.168.0.0/24'
```

```
vyos@vyos# run show ipoe-server sessions

ifname | called-sid |    calling-sid    |     ip      | ip6 | ip6-dp | rate-limit |␣
↪state  |  uptime  |        sid
-------+------------+-------------------+-------------+-----+--------+------------+---
↪-----+----------+------------------
ipoe0  | eth2       | 08:00:27:2f:d8:06 | 192.168.0.2 |     |        | 500/500    |␣
↪active | 00:00:05 | dccc870fd31349fb
```

12.10 PPPoE server

VyOS utilizes accel-ppp to provide PPPoE server functionality. It can be used with local authentication or a connected RADIUS server.

Note: Please be aware, due to an upstream bug, config changes/commits will restart the ppp daemon and will reset existing PPPoE connections from connected users, in order to become effective.

12.10.1 Configuration

The example below uses ACN as access-concentrator name, assigns an address from the pool 10.1.1.100-111, terminates at the local endpoint 10.1.1.1 and serves requests only on eth1.

```
set service pppoe-server access-concentrator 'ACN'
set service pppoe-server authentication local-users username foo password 'bar'
set service pppoe-server authentication mode 'local'
set service pppoe-server client-ip-pool start '10.1.1.100'
set service pppoe-server client-ip-pool stop '10.1.1.111'
set service pppoe-server dns-servers server-1 '10.100.100.1'
```

(continues on next page)

(continued from previous page)

```
set service pppoe-server dns-servers server-2 '10.100.200.1'
set service pppoe-server interface 'eth1'
set service pppoe-server local-ip '10.1.1.2'
```

Connections can be locally checked via the command

```
show pppoe-server sessions
ifname | username |     ip     |    calling-sid    | rate-limit  | state  |  uptime
| rx-bytes | tx-bytes
-------+----------+------------+-------------------+-------------+--------+----------
+----------+----------
ppp0   | foo      | 10.1.1.100 | 08:00:27:ba:db:15 | 20480/10240 | active | 00:00:11
| 214 B    | 76 B
```

Client IP address pools

To automatically assign the client an IP address as tunnel endpoint, a client IP pool is needed. The source can be either RADIUS or a local subnet or IP range defintion.

Once the local tunnel endpoint `set service pppoe-server local-ip '10.1.1.2'` has been defined, the client IP pool can be either defined as a range or as subnet using CIDR notation. If the CIDR notation is used, multiple subnets can be setup which are used sequentially.

Client IP address via IP range defintion

```
set service pppoe-server client-ip-pool start '10.1.1.100'
set service pppoe-server client-ip-pool stop '10.1.1.111'
```

Client IP subnets via CIDR notation

```
set service pppoe-server client-ip-pool subnet '10.1.1.0/24'
set service pppoe-server client-ip-pool subnet '10.1.2.0/24'
set service pppoe-server client-ip-pool subnet '10.1.3.0/24'
```

RADIUS based IP pools (Framed-IP-Address)

To use a radius server, you need to switch to authentication mode radius and of course need to specify an IP for the server. You can have multiple RADIUS server configured, if you wish to achieve redundancy.

```
set service pppoe-server access-concentrator 'ACN'
set service pppoe-server authentication mode 'radius'
set service pppoe-server authentication radius-server 10.1.100.1 secret 'secret'
set service pppoe-server interface 'eth1'
set service pppoe-server local-ip '10.1.1.2'
```

RADIUS provides the IP addresses in the example above via Framed-IP-Address.

RADIUS sessions management DM/CoA

For remotely disconnect sessions and change some authentication parameters you can configure dae-server

```
set service pppoe-server authentication radius-settings dae-server ip-address '10.1.1.
2'
set service pppoe-server authentication radius-settings dae-server port '3799'
set service pppoe-server authentication radius-settings dae-server secret 'secret123'
```

Example, from radius-server send command for disconnect client with username test

```
root@radius-server:~# echo "User-Name=test" | radclient -x 10.1.1.2:3799 disconnect␣
↪secret123
```

You can also use another attributes for identify client for disconnect, like Framed-IP-Address, Acct-Session-Id, etc. Result commands appears in log

```
show log | match Disconnect*
```

Example for changing rate-limit via RADIUS CoA

```
echo "User-Name=test,Filter-Id=5000/4000" | radclient 10.1.1.2:3799 coa secret123
```

Filter-Id=5000/4000 (means 5000Kbit down-stream rate and 4000Kbit up-stream rate) If attribute Filter-Id redefined, replace it in radius coa request

Automatic VLAN creation

VLAN's can be created by accel-ppp on the fly if via the use of the kernel module vlan_mon, which is monitoring incoming vlans and creates the neccessary VLAN if required and allowed. VyOS supports the use of either VLAN ID's or entire ranges, both values can be defined at the same time for an interface.

```
set service pppoe-server interface eth3 vlan-id 100
set service pppoe-server interface eth3 vlan-id 200
set service pppoe-server interface eth3 vlan-range 500-1000
set service pppoe-server interface eth3 vlan-range 2000-3000
```

The pppoe-server will now create these VLANs if required and once the user session has been cancelled, and the VLAN is not neccessary anymore, it will remove it again.

12.10.2 Bandwidth Shaping

Bandwidth rate limits can be set for local users or RADIUS based attributes.

Bandwidth Shaping for local users

The rate-limit is set in kbit/sec.

```
set service pppoe-server access-concentrator 'ACN'
set service pppoe-server authentication local-users username foo password 'bar'
set service pppoe-server authentication local-users username foo rate-limit download
↪'20480'
set service pppoe-server authentication local-users username foo rate-limit upload
↪'10240'
set service pppoe-server authentication mode 'local'
set service pppoe-server client-ip-pool start '10.1.1.100'
set service pppoe-server client-ip-pool stop '10.1.1.111'
set service pppoe-server dns-servers server-1 '10.100.100.1'
set service pppoe-server dns-servers server-2 '10.100.200.1'
set service pppoe-server interface 'eth1'
set service pppoe-server local-ip '10.1.1.2'
```

Once the user is connected, the user session is using the set limits and can be displayed via 'show pppoe-server sessions'.

```
show pppoe-server sessions
ifname | username |     ip      |    calling-sid    | rate-limit  | state  |  uptime
↪| rx-bytes | tx-bytes
-------+----------+------------+-------------------+-------------+--------+----------
↪+----------+----------
ppp0   | foo      | 10.1.1.100 | 08:00:27:ba:db:15 | 20480/10240 | active | 00:00:11
↪| 214 B    | 76 B
```

RADIUS based shaper setup

The current attribute 'Filter-Id' is being used as default and can be setup within RADIUS:

Filter-Id=2000/3000 (means 2000Kbit down-stream rate and 3000Kbit up-stream rate)

The command below enables it, assuming the RADIUS connection has been setup and is working.

```
set service pppoe-server authentication radius-settings rate-limit enable
```

Other attributes can be used, but they have to be in one of the dictionaries in /usr/share/accel-ppp/radius.

12.10.3 Practical Configuration Examples

Dual-stack provisioning with IPv6 PD via pppoe

The example below covers a dual-stack configuration via pppoe-server.

```
set service pppoe-server authentication local-users username test password 'test'
set service pppoe-server authentication mode 'local'
set service pppoe-server client-ip-pool start '192.168.0.1'
set service pppoe-server client-ip-pool stop '192.168.0.10'
set service pppoe-server client-ipv6-pool delegate-prefix '2001:db8:8003::1/48,56'
set service pppoe-server client-ipv6-pool prefix '2001:db8:8002::1/48,64'
set service pppoe-server dns-servers server-1 '8.8.8.8'
set service pppoe-server dnsv6-servers server-1 '2001:4860:4860::8888'
set service pppoe-server interface 'eth2'
set service pppoe-server local-ip '10.100.100.1'
```

The client, once successfully authenticated, will receive an IPv4 and an IPv6 /64 address, to terminate the pppoe endpoint on the client side and a /56 subnet for the clients internal use.

```
vyos@pppoe-server:~$ sh pppoe-server sessions
 ifname | username |     ip      |            ip6            |       ip6-dp        |
↪ calling-sid    | rate-limit | state  |  uptime  | rx-bytes | tx-bytes
--------+----------+-------------+---------------------------+---------------------+---
↪----------------+------------+--------+----------+----------+----------
 ppp0   | test     | 192.168.0.1 | 2001:db8:8002:0:200::/64 | 2001:db8:8003::1/56 |
↪08:00:27:12:42:eb |            | active | 00:00:49 | 875 B    | 2.1 KiB
```

12.11 SSTP server

VyOS utilizes accel-ppp to provide SSTP server functionality. It can be used with local authentication or a connected RADIUS server.

Note: Please be aware, due to an upstream bug, config changes/commits will restart the ppp daemon and will reset existing PPPoE connections from connected users, in order to become effective.

12.11.1 Configuration

The Secure Socket Tunneling Protocol (SSTP), provides ppp via a SSL/TLS channel. Using publically signed certificates as well a by private PKI, is fully supported. All certficates should be stored on VyOS under `/config/user-data/sstp`.

Self Signed CA and server certificates

To generate the CA, the server private key and certificates the following commands can be used.

```
vyos@vyos:~$ conf
[edit]
vyos@vyos# mkdir -p /config/user-data/sstp && cd /config/user-data/sstp
[edit]
openssl req -newkey rsa:4096 -new -nodes -x509 -days 3650 -keyout server.key -out␣
↪server.crt

Generating a 4096 bit RSA private key
.........................++
.................................................................++
writing new private key to 'server.key'
[...]
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

vyos@vyos# openssl req -new -x509 -key server.key -out ca.crt
[...]
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
[edit]
vyos@vyos#
```

The example below will answer configuration request for the user user `foo`.

Use <tab> to setup the `set sstp-settings ssl-certs ...`, it automatically looks for all files and directories in `/config/user-data/sstp`.

```
set authentication local-users username foo password 'bar'
set authentication mode 'local'
set network-settings client-ip-settings gateway-address '10.100.100.1'
```

(continues on next page)

(continued from previous page)

```
set network-settings client-ip-settings subnet '192.168.0.0/24'
set network-settings dns-server primary-dns '10.100.100.1'
set network-settings dns-server secondary-dns '10.200.100.1'
set sstp-settings ssl-certs ca 'ca.crt'
set sstp-settings ssl-certs server-cert 'server.crt'
set sstp-settings ssl-certs server-key 'server.key'
```

12.12 UDP broadcast relay

Certain vendors use broadcasts to identify their equipemnt within one ethernet segment. Unfortunately if you split your network with multiple VLANs you loose the ability of identifying your equiment.

This is where "UDP broadcast relay" comes into play! It will forward received broadcasts to other configured networks.

Every UDP port which will be forward requires one unique ID. Currently we support 99 IDs!

Example #1: To forward all broadcast packets received on *UDP port 1900* on *eth3*, *eth4* or *eth5* to all other interfaces in this configuration.

```
set service broadcast-relay id 1 description 'SONOS'
set service broadcast-relay id 1 interface 'eth3'
set service broadcast-relay id 1 interface 'eth4'
set service broadcast-relay id 1 interface 'eth5'
set service broadcast-relay id 1 port '1900'
```

Example #2: To Forward all broadcasts packets received on *UDP port 6969* on *eth3* or *eth4* to the other interface in this configuration.

```
set service broadcast-relay id 2 description 'SONOS MGMT'
set service broadcast-relay id 2 interface 'eth3'
set service broadcast-relay id 2 interface 'eth4'
set service broadcast-relay id 2 port '6969'
```

12.12.1 Disable Instance(s)

Each broadcast relay instance can be individually disabled without deleting the configured node by using the following command:

```
set service broadcast-relay id <n> disable
```

In addition you can also disable the whole service without removing the configuration by:

```
set service broadcast-relay disable
```

Note: You can run the UDP broadcast relay service on multiple routers connected to a subnet. There is **NO** UDP broadcast relay packet storm!

12.13 SNMP

Simple Network Management Protocol (SNMP) is an Internet Standard protocol for collecting and organizing information about managed devices on IP networks and for modifying that information to change device behavior. Devices that typically support SNMP include cable modems, routers, switches, servers, workstations, printers, and more.

SNMP is widely used in network management for network monitoring. SNMP exposes management data in the form of variables on the managed systems organized in a management information base (MIB) which describe the system status and configuration. These variables can then be remotely queried (and, in some circumstances, manipulated) by managing applications.

Three significant versions of SNMP have been developed and deployed. SNMPv1 is the original version of the protocol. More recent versions, SNMPv2c and SNMPv3, feature improvements in performance, flexibility and security.

SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects.

12.13.1 Overview and basic concepts

In typical uses of SNMP, one or more administrative computers called managers have the task of monitoring or managing a group of hosts or devices on a computer network. Each managed system executes a software component called an agent which reports information via SNMP to the manager.

An SNMP-managed network consists of three key components:

- Managed devices
- Agent – software which runs on managed devices
- Network management station (NMS) – software which runs on the manager

A managed device is a network node that implements an SNMP interface that allows unidirectional (read-only) or bidirectional (read and write) access to node-specific information. Managed devices exchange node-specific information with the NMSs. Sometimes called network elements, the managed devices can be any type of device, including, but not limited to, routers, access servers, switches, cable modems, bridges, hubs, IP telephones, IP video cameras, computer hosts, and printers.

An agent is a network-management software module that resides on a managed device. An agent has local knowledge of management information and translates that information to or from an SNMP-specific form.

A network management station executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs may exist on any managed network.

Note: VyOS SNMP supports both IPv4 and IPv6.

12.13.2 SNMP protocol versions

VyOS itself supports SNMPv2 (version 2) and SNMPv3 (version 3) where the later is recommended because of improved security (optional authentication and encryption).

Fig. 4: Image thankfully borrowed from https://en.wikipedia.org/wiki/File:SNMP_communication_principles_diagram.PNG which is under the GNU Free Documentation License

12.13.3 SNMPv2

SNMPv2 is the original and most commonly used version. For authorizing clients, SNMP uses the concept of communities. Communities may have authorization set to read only (this is most common) or to read and write (this option is not actively used in VyOS).

SNMP can work synchronously or asynchronously. In synchronous communication, the monitoring system queries the router periodically. In asynchronous, the router sends notification to the "trap" (the monitoring host).

SNMPv2 does not support any authentication mechanisms, other than client source address, so you should specify addresses of clients allowed to monitor the router. Note that SNMPv2 also supports no encryption and always sends data in plain text.

Example

```
# Define a community
set service snmp community routers authorization ro

# Allow monitoring access from the entire network
set service snmp community routers network 192.0.2.0/24
set service snmp community routers network 2001::db8:ffff:eeee::/64

# Allow monitoring access from specific addresses
set service snmp community routers client 203.0.113.10
set service snmp community routers client 203.0.113.20

# Define optional router information
set service snmp location "UK, London"
set service snmp contact "admin@example.com"

# Trap target if you want asynchronous communication
set service snmp trap-target 203.0.113.10
```

(continues on next page)

(continued from previous page)

```
# Listen only on specific IP addresses (port defaults to 161)
set service snmp listen-address 172.16.254.36 port 161
set service snmp listen-address 2001:db8::f00::1
```

12.13.4 SNMPv3

SNMPv3 is an updated version that, among other things, supports encryption and cryptographic authentication of clients.

Example

```
set service snmp v3 engineid '0x0aa0d6c6f450'
set service snmp v3 group defaultgroup mode 'ro'
set service snmp v3 group defaultgroup seclevel 'priv'
set service snmp v3 group defaultgroup view 'defaultview'
set service snmp v3 view defaultview oid '1'

set service snmp v3 user testUser1 auth plaintext-key testUserKey1
set service snmp v3 user testUser1 auth type 'md5'
set service snmp v3 user testUser1 engineid '0x0aa0d6c6f450'
set service snmp v3 user testUser1 group 'defaultgroup'
set service snmp v3 user testUser1 mode 'ro'
set service snmp v3 user testUser1 privacy type aes
set service snmp v3 user testUser1 privacy plaintext-key testUserKey1
```

After commit the resulting configuration will look like:

Note: SNMPv3 keys won't we stored in plaintext. On `commit` the keys will be encrypted and the encrypted key is based on the engineid!

```
vyos@vyos# show service snmp
 v3 {
     engineid 0x0aa0d6c6f450
     group defaultgroup {
         mode ro
         seclevel priv
         view defaultview
     }
     user testUser1 {
         auth {
             encrypted-key 0x3b68d4162c2c817b8e9dfb6f08583e5d
             type md5
         }
         engineid 0x0aa0d6c6f450
         group defaultgroup
         mode ro
         privacy {
             encrypted-key 0x3b68d4162c2c817b8e9dfb6f08583e5d
             type aes
         }
```

(continues on next page)

(continued from previous page)

```
    }
    view defaultview {
        oid 1 {
        }
    }
}
```

12.13.5 SNMP Extensions

To extend SNMP agent functionality, custom scripts can be executed every time the agent is being called. This can be achieved by using arbitrary extension commands. The first step is to create a functional script of course, then upload it to your VyOS instance via the command `scp your_script.sh vyos@your_router:/config/user-data`. Once the script is uploaded, it needs to be configured via the command below.

```
set service snmp script-extensions extension-name my-extension script your_script.sh
commit
```

The OID `.1.3.6.1.4.1.8072.1.3.2.3.1.1.4.116.101.115.116`, once called, will contain the output of the extension.

```
root@vyos:/home/vyos# snmpwalk -v2c  -c public 127.0.0.1 nsExtendOutput1
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."my-extension" = STRING: hello
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."my-extension" = STRING: hello
NET-SNMP-EXTEND-MIB::nsExtendOutNumLines."my-extension" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendResult."my-extension" = INTEGER: 0
```

12.14 SSH

Secure Shell (*SSH*) is a cryptographic network protocol for operating network services securely over an unsecured network.[1] The standard TCP port for SSH is 22. The best known example application is for remote login to computer systems by users.

SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server. Common applications include remote command-line login and remote command execution, but any network service can be secured with SSH. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2.

The most visible application of the protocol is for access to shell accounts on Unix-like operating systems, but it sees some limited use on Windows as well. In 2015, Microsoft announced that they would include native support for SSH in a future release.

SSH was designed as a replacement for Telnet and for unsecured remote shell protocols such as the Berkeley rlogin, rsh, and rexec protocols. Those protocols send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet.

12.14.1 Configuration

Enabling SSH only requires you to add `service ssh port NN`, where 'NN' is the port you want SSH to listen on. By default, SSH runs on port 22.

```
set service ssh port 22
```

Options

- Listening address - Specify the IPv4/IPv6 listening address for connection requests. Multiple `listen-address` nodes can be defined.

  ```
  set service ssh listen-address <address>
  ```

- Allow `root` login, this can be set to allow `root` logins on SSH connections, however it is not advisable to use this setting as this bears serious security risks. The default system user posesses all required privileges.

  ```
  set service ssh allow-root
  ```

- Allowed ciphers - A number of allowed ciphers can be specified, use multiple occurances to allow multiple ciphers.

  ```
  set service ssh ciphers <cipher>
  ```

  Available ciphers:

- *3des-cbc*
- *aes128-cbc*
- *aes192-cbc*
- *aes256-cbc*
- *aes128-ctr*
- *aes192-ctr*
- *aes256-ctr*
- *arcfour128*
- *arcfour256*
- *arcfour*
- *blowfish-cbc*
- *cast128-cbc*
- Disable password authentication - If SSH key authentication is set up, password-based user authetication can be disabled. This hardens security!

  ```
  set service ssh disable-password-authentication
  ```

- Disable host validation - Disable the host validation through reverse DNS lookups.

  ```
  set service ssh disable-host-validation
  ```

- MAC algorithms - Specifies the available MAC (message authentication code) algorithms. The MAC algorithm is used in protocol version 2 for data integrity protection. Multiple algorithms can be entered.

  ```
  set service ssh macs <macs>
  ```

  Supported MACs:

- *hmac-md5*
- *hmac-md5-96*
- *hmac-ripemd160*

- *hmac-sha1*
- *hmac-sha1-96*
- *hmac-sha2-256*
- *hmac-sha2-512*
- *umac-64@openssh.com*
- *umac-128@openssh.com*
- *hmac-md5-etm@openssh.com*
- *hmac-md5-96-etm@openssh.com*
- *hmac-ripemd160-etm@openssh.com*
- *hmac-sha1-etm@openssh.com*
- *hmac-sha1-96-etm@openssh.com*
- *hmac-sha2-256-etm@openssh.com*
- *hmac-sha2-512-etm@openssh.com*
- *umac-64-etm@openssh.com*
- *umac-128-etm@openssh.com*

Key Authentication

It is highly recommended to use SSH Key authentication. By default there is only one user (`vyos`), and you can assign any number of keys to that user. You can generate a ssh key with the `ssh-keygen` command on your local machine, which will (by default) save it as `~/.ssh/id_rsa.pub` which is in three parts:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAA...VByBD5lKwEWB username@host.
example.com
```

Only the type (`ssh-rsa`) and the key (`AAAB3N...`) are used. Note that the key will usually be several hundred characters long, and you will need to copy and paste it. Some terminal emulators may accidentally split this over several lines. Be attentive when you paste it that it only pastes as a single line. The third part is simply an identifier, and is for your own reference.

Assign SSH Key to user

Under the user (in this example, `vyos`), add the public key and the type. The *identifier* is simply a string that is relevant to you.

```
set system login user vyos authentication public-keys 'identifier' key "AAAAB3Nz...."
set system login user vyos authentication public-keys 'identifier' type ssh-rsa"
```

You can assign multiple keys to the same user by changing the identifier. In the following example, both Unicron and xrobau will be able to SSH into VyOS as the `vyos` user using their own keys.

```
set system login user vyos authentication public-keys 'Unicron' key "AAAAB3Nz...."
set system login user vyos authentication public-keys 'Unicron' type ssh-rsa
set system login user vyos authentication public-keys 'xrobau' key "AAAAQ39x...."
set system login user vyos authentication public-keys 'xrobau' type ssh-rsa
```

12.15 TFTP

Trivial File Transfer Protocol (*TFTP*) is a simple lockstep File Transfer Protocol which allows a client to get a file from or put a file onto a remote host. One of its primary uses is in the early stages of nodes booting from a local area network. TFTP has been used for this application because it is very simple to implement.

12.15.1 Example

```
# If you want to enable uploads, else TFTP server will act as read-only (optional)
set service tftp-server allow-upload

# Directory for TFTP server content
set service tftp-server directory '/config/tftpboot'

# On which addresses we want to listen for incoming TFTP connections? (mandatory)
set service tftp-server listen-address '2001:db8:ffee::1'
set service tftp-server listen-address '10.10.1.1'
```

Note: Choose your `directory` location carefully or you will loose the content on image upgrades. Any directory under `/config` is save at this will be migrated.

Note: Configuring a listen-address is essential for the service to work.

The resulting configuration will look like:

```
vyos@vyos# show service
 tftp-server {
    allow-upload
    directory /config/tftpboot
    listen-address 2001:db8:ffee::1
    listen-address 10.10.1.1
 }
```

12.16 Webproxy

The proxy service in VyOS is based on Squid3 and some related modules.

Squid is a caching and forwarding HTTP web proxy. It has a wide variety of uses, including speeding up a web server by caching repeated requests, caching web, DNS and other computer network lookups for a group of people sharing network resources, and aiding security by filtering traffic. Although primarily used for HTTP and FTP, Squid includes limited support for several other protocols including Internet Gopher, SSL,[6] TLS and HTTPS. Squid does not support the SOCKS protocol.

All examples here assumes that your inside ip address is `192.168.0.1`. Replace with your own where applicable.

URL Filtering is provided by Squidguard.

12.16.1 Configuration

```
# Enable proxy service
set service webproxy listen-address 192.168.0.1

# By default it will listen to port 3128. If you wan't something else you have to␣
↪define that.
set service webproxy listen-address 192.168.0.1 port 2050

# By default the transparent proxy on that interface is enabled. To disable that you␣
↪simply
set service webproxy listen-address 192.168.0.1 disable-transparent

# Block specific urls
set service webproxy url-filtering squidguard local-block myspace.com

# If you want to you can log these blocks
set service webproxy url-filtering squidguard log local-block
```

Options

12.16.2 Filtering by category

If you wan't to use existing blacklists you have to create/download a database first. Otherwise you will not be able to commit the config changes.

```
vyos@vyos# commit
[ service webproxy ]
Warning: no blacklists installed
Unknown block-category [ads] for policy [default]

[[service webproxy]] failed
Commit failed
```

- Download/Update complete blacklist

  `update webproxy blacklists`

- Download/Update partial blacklist

  `update webproxy blacklists category ads`

  Use tab completion to get a list of categories.

- To auto update the blacklist files

  `set service webproxy url-filtering squidguard auto-update update-hour 23`

- To configure blocking add the following to the configuration

  `set service webproxy url-filtering squidguard block-category ads`

  `set service webproxy url-filtering squidguard block-category malware`

12.16.3 Authentication

The embedded Squid proxy can use LDAP to authenticate users against a company wide directory. The following configuration is an example of how to use Active Directory as authentication backend. Queries are done via LDAP.

```
vyos@vyos# show service webproxy
 authentication {
     children 5
     credentials-ttl 60
     ldap {
         base-dn DC=example,DC=local
         bind-dn CN=proxyuser,CN=Users,DC=example,DC=local
         filter-expression (cn=%s)
         password Qwert1234
         server ldap.example.local
         username-attribute cn
     }
     method ldap
     realm "VyOS Webproxy"
 }
 cache-size 100
 default-port 3128
 listen-address 192.168.188.103 {
     disable-transparent
 }
```

- `base-dn` set the base directory for the search
- `bind-dn` and `password`: set the user, which is used for the ldap search
- `filter-expression`: set the exact filter which a authorized user match in a ldap-search. In this example every User is able to authorized.

You can find more about the ldap authentication here

12.16.4 Adjusting cache size

The size of the proxy cache can be adjusted by the user.

```
set service webproxy cache-size
 Possible completions:
   <0-4294967295>
                Disk cache size in MB (default 100)
   0            Disable disk caching
   100
```

12.16.5 Bypassing the webproxy

Some services don't work correctly when being handled via a web proxy. So sometimes it is useful to bypass a transparent proxy:

- To bypass the proxy for every request that is directed to a specific destination:

  `set service webproxy whitelist destination-address 198.51.100.33`

  `set service webproxy whitelist destination-address 192.0.2.0/24`

- To bypass the proxy for every request that is coming from a specific source:

  `set service webproxy whitelist source-address 192.168.1.2`

  `set service webproxy whitelist source-address 192.168.2.0/24`

  (This can be useful when a called service has many and/or often changing destination addresses - e.g. Netflix.)

CHAPTER 13

System

After a basic system setup by setting up *Interface Addresses*, VyOS should be ready for further configuration which is described in this chapter.

13.1 Event Handler

Event handler allows you to execute scripts when a string that matches a regex appears in a text stream (e.g. log file).

It uses "feeds" (output of commands, or a named pipes) and "policies" that define what to execute if a regex is matched.

```
system
event-handler
    feed <name>
    description <feed description>
    policy <policy name>
    source
        preset
        syslog # Use the syslog logs for feed
        custom
        command <command to execute> # E.g. "tail -f /var/log/somelogfile"
        named-pipe <path to a names pipe>
    policy <policy name>
    description <policy description>
    event <event name>
        description <event description>
        pattern <regex>
        run <command to run>
```

In this small example a script runs every time a login failed and an interface goes down

```
vyos@vyos# show system event-handler
feed Syslog {
    policy MyPolicy
```

(continues on next page)

(continued from previous page)

```
    source {
        preset syslog
    }
}
policy MyPolicy {
    description "Test policy"
    event BadThingsHappened {
        pattern "authentication failure"
        pattern "interface \.* index \d+ .* DOWN.*"
        run /config/scripts/email-to-admin
    }
}
```

13.2 Flow Accounting

VyOS supports flow accounting through NetFlow or sFlow.

For both types you need to specify the interfaces for which the data will be collected:

```
set system flow-accounting interface eth0
set system flow-accounting interface bond3
```

NetFlow is a protocol originating from Cisco Systems. It works on level3. VyOS supports version 1, 5 and 9

NetFlow v5 example:

```
set system flow-accounting netflow engine-id 100
set system flow-accounting netflow version 5
set system flow-accounting netflow server 192.168.2.10 port 2055
```

13.3 Host Information

This section describes the system's host information and how to configure them, it covers the following topics:

- Host name
- Domain
- IP address
- Default gateway
- Aliases

13.3.1 Host Name

A hostname is the label (name) assigned to a network device (a host) on a network and is used to distinguish one device from another on specific networks or over the internet.

Set a system host name:

```
set system host-name <hostname>
```

Note: Only letters, numbers and hyphens are allowed.

Show host name:

```
show system host-name
```

Delete host name:

```
delete system host-name <hostname>
```

Example: Set system hostname to 'RT01':

```
set system host-name RT01
commit
show system host-name
  host-name RT01
```

13.3.2 Domain Name

A domain name is the label (name) assigned to a computer network and is thus unique.

Set the system's domain:

```
set system domain-name <domain>
```

Note: Only letters, numbers, hyphens and periods are allowed.

Show domain:

```
show system domain-name
```

Remove domain name:

```
set system delete domain-name <domain>
```

Example: Set system domain to example.com:

```
set system domain-name example.com
commit
show system domain-name
  domain-name example.com
```

13.3.3 Static host mappings

How to assign IPs to interfaces is described in chapter *Interface Addresses*. This section shows how to statically map a system IP to its host name for local (meaning on this VyOS instance) DNS resolution:

```
set system static-host-mapping host-name <hostname> inet <IP address>
```

Show static mapping:

```
show system static-host-mapping
```

Example: Create a static mapping between the system's hostname *RT01* and IP address *10.20.30.41*:

```
set system static-host-mapping host-name RT01 inet 10.20.30.41
commit
show system static-host-mapping
  host-name RT01 {
      inet 10.20.30.41
  }
```

Aliases

One or more system aliases (static mappings) can be defined:

```
set system static-host-mapping host-name <hostname> alias <alias>
```

Show aliases:

```
show system static-mapping
```

Delete alias:

```
delete system static-host-mapping host-name <hostname> alias <alias>
```

Example: Set alias *router1* for system with hostname *RT01*:

```
set system static-host-mapping host-name RT01 alias router1
commit
show system static-host-mapping
  host-name RT01 {
      alias router1
      inet 10.20.30.41
  }
```

13.3.4 Default Gateway/Route

In the past (VyOS 1.1.8) used a gateway-address configured in the system tree (*set system gateway-address <IP address>*) this is no longer supported and existing configurations are migrated to the new CLI commands.

It is replaced by inserting a static route into the routing table using:

```
set protocols static route 0.0.0.0/0 next-hop <gateway ip>
```

Delete the default route from the system

```
delete protocols static route 0.0.0.0/0
```

Show default route:

```
vyos@vyos$ show ip route 0.0.0.0
Routing entry for 0.0.0.0/0
  Known via "static", distance 1, metric 0, best
  Last update 3d00h23m ago
  * 172.16.34.6, via eth1
```

13.4 Serial console

13.4.1 Configuring Console

```
vyos@vyos# set system console
Possible completions:
+> device       Serial console device name
+> network      Network based console logging
   powersave    Enable screen blank powersaving on VGA console
```

13.5 Login

The default VyOS user account (*vyos*), as well as newly created user accounts, have all capabilities to configure the system. All accounts have sudo capabilities and therefore can operate as root on the system. Setting the level to admin is optional, all accounts on the system will have admin privileges.

Both local administered and remote administered RADIUS (Remote Authentication Dial-In User Service) accounts are supported.

13.5.1 Local

Create user account *jsmith* and the password *mypassword*.

```
set system login user jsmith full-name "Johan Smith"
set system login user jsmith authentication plaintext-password mypassword
```

The command:

```
show system login
```

will show the contents of `system login` configuration node:

```
user jsmith {
    authentication {
        encrypted-password $6$0OQHjuQ8M$AYXVn7jufdfqPrSk4/
↪XXsDBw99JBtNsETkQKDgVLptXogHA2bU9BWlvViOFPBoFxIi.iqjqrvsQdQ./cfiiPT.
        plaintext-password ""
    }
    full-name "Johan Smith"
    level admin
}
```

SSH with Public Keys

The following command will load the public key *dev.pub* for user *jsmith*

```
loadkey jsmith dev.pub
```

Note: This requires uploading the *dev.pub* public key to the VyOS router first. As an alternative you can also load the SSH public key directly from a remote system:

```
loadkey jsmith scp://devuser@dev001.vyos.net/home/devuser/.ssh/dev.pub
```

In addition SSH public keys can be fully added using the CLI. Each key can be given a unique identifier, *calypso* is used oin the example below to id an SSH key.

```
set system login user jsmith authentication public-keys callisto key 'AAAAB3Hso...Q=='
set system login user jsmith authentication public-keys callisto type 'ssh-rsa'
```

13.5.2 RADIUS

VyOS supports using one or more RADIUS servers as backend for user authentication.

The following command sets up two servers for RADIUS authentication, one with a discrete timeout of *5* seconds and a discrete port of *1812* and the other using a default timeout and port.

```
set system login radius-server 192.168.1.2 secret 's3cr3t0815'
set system login radius-server 192.168.1.2 timeout '5'
set system login radius-server 192.168.1.2 port '1812'
set system login radius-server 192.168.1.3 secret 's3cr3t0816'
```

This configuration results in:

```
show system login
 radius-server 192.168.1.2 {
     secret s3cr3t0815
     timeout 5
     port 1812
 }
 radius-server 192.168.1.3 {
     secret s3cr3t0816
 }
```

Source Address

RADIUS servers could be hardened by only allowing certain IP addresses to connect. As of this the source address of each RADIUS query can be configured. If this is not set incoming connections to the RADIUS server will use the nearest interface address pointing towards the RADIUS server - making it error prone on e.g. OSPF networks when a link fails.

```
set system login radius-source-address 192.168.1.254
```

13.5.3 Login Banner

You are able to set post-login or pre-login messages with the following lines:

```
set system login banner pre-login "UNAUTHORIZED USE OF THIS SYSTEM IS PROHIBITED\n"
set system login banner post-login "Welcome to VyOS"
```

the **\n** create a newline.

13.6 Syslog

Per default VyOSs has minimal syslog logging enabled which is stored and rotated locally. Errors will be always logged to a local file, which includes *local7* error messages, emergency messages will be sent to the console, too.

To configure syslog, you need to switch into configuration mode.

13.6.1 Logging to serial console

The below would log all messages to `/dev/console`.

```
set system syslog console facility all level all
```

Use the **[tab]** function to display all facilities and levels which can be configured.

```
vyos@vyos# set system syslog console facility <TAB>
Possible completions:
> all          All facilities excluding "mark"
> auth         Authentication and authorization
> authpriv     Non-system authorization
> cron         Cron daemon
> daemon       System daemons
> kern         Kernel
> lpr          Line printer spooler
> mail         Mail subsystem
> mark         Timestamp
> news         USENET subsystem
> protocols    depricated will be set to local7
> security     depricated will be set to auth
> syslog       Authentication and authorization
> user         Application processes
> uucp         UUCP subsystem
> local0       Local facility 0
> local1       Local facility 1
> local2       Local facility 2
> local3       Local facility 3
> local4       Local facility 4
> local5       Local facility 5
> local6       Local facility 6
> local7       Local facility 7

vyos@vyos# set system syslog console facility all level <TAB>
Possible completions:
 emerg        Emergency messages
 alert        Urgent messages
 crit         Critical messages
 err          Error messages
 warning      Warning messages
 notice       Messages for further investigation
 info         Informational messages
 debug        Debug messages
 all          Log everything
```

13.6.2 Logging to a custom file

Logging to a custom file, rotation size and the number of rotate files left on the system can be configured.

```
set system syslog file <FILENAME> facility <FACILITY>  level <LEVEL>
set system syslog file <FILENAME> archive file <NUMBER OF FILES>
set system syslog file FILENAME archive size <FILESIZE>
```

The very same setting can be applied to the global configuration, to modify the defaults for the global logging.

13.6.3 Logging to a remote host

Logging to a remote host leaves the local logging configuration intact, it can be configured in parallel. You can log ro multiple hosts at the same time, using either TCP or UDP. The default is sending the messages via UDP.

UDP

```
set system syslog host 10.1.1.1 facility all level all
<optional>
set system syslog host 10.1.1.1 facility all protocol udp
```

TCP

```
set system syslog host 10.1.1.2 facility all level all
set system syslog host 10.1.1.2 facility all protocol tcp
```

13.6.4 Logging to a local user account

If logging to a local useraccount is configured, all defined log messages are display on the console if the local user is logged in, if the user is not logged in, no messages are being displayed.

```
set system syslog user <LOCAL_USERNAME> facility <FACILITY> level <LEVEL>
```

13.6.5 Show logs

Display log files on the console

```
vyos@vyos:~$ show log
Possible completions:
  <Enter>       Execute the current command
  all           Show contents of all master log files
  authorization Show listing of authorization attempts
  cluster       Show log for Cluster
  conntrack-sync
                Show log for Conntrack-sync
  dhcp          Show log for Dynamic Host Control Protocol (DHCP)
  directory     Show listing of user-defined log files
  dns           Show log for Domain Name Service (DNS)
  file          Show contents of user-defined log file
  firewall      Show log for Firewall
  https         Show log for Https
  image         Show logs from an image
  lldp          Show log for Lldp
```

(continues on next page)

(continued from previous page)

```
  nat           Show log for Network Address Translation (NAT)
  openvpn       Show log for Openvpn
  snmp          Show log for Simple Network Monitoring Protocol (SNMP)
  tail          Monitor last lines of messages file
  vpn           Show log for Virtual Private Network (VPN)
  vrrp          Show log for Virtual Router Redundancy Protocol (VRRP)
  webproxy      Show log for Webproxy
```

13.6.6 Show contents of a log file in an image

Log messages from a specified image can be displayed on the console:

```
$ show log image <image name>
$ show log image <image name> [all | authorization | directory | file <file name> |␣
↪tail <lines>]
```

Details of allowed parameters:

all	Display contents of all master log files of the specified image
authorization	Display all authorization attempts of the specified image
directory	Display list of all user-defined log files of the specified image
file <file name>	Display contents of a specified user-defined log file of the specified image
tail	Display last lines of the system log of the specified image
<lines>	Number of lines to be displayed, default 10

When no options/parameters are used, the contents of the main syslog file are displayed.

13.7 Task scheduler

Task scheduler — allows scheduled task execution. Note that scripts excecuted this way are executed as root user - this may be dangerous.
Together with *Command scripting* this can be used for automating configuration.

```
system
    task-scheduler
        task <name>
            cron-spec <UNIX cron time spec>
            executable
                arguments <arguments string>
                path <path to executable>
            interval
                <int32>[mhd]
```

13.7.1 Interval

You are able to set the time as an time interval.

```
set system task-scheduler task <name> interval <value><suffix>
```

Sets the task to execute every N minutes, hours, or days. Suffixes:

- m — minutes
- h — hours
- d — days

If suffix is omitted, minutes are implied.

Or set the execution time in common cron time.

```
set system task-scheduler task TEST crontab-spec "* * * 1 *"
```

13.7.2 Example

```
system
    task-scheduler
        task mytask
            interval 2h
            executable
                path /config/scripts/mytask
                arguments "arg1 arg2 arg3"
        task anothertask
            cron-spec "* * * 1 *"
            executable
                path /config/scripts/anothertask
```

13.8 Config Management

The following changes the number of commit revisions. In the default settings, 20 revisions are stored locally.

```
set system config-management commit-revisions 50
```

If you want to save all config changes to a remote destination. Set the commit-archive location. Every time a commit is successfully the config.boot file will be copied to the defined destinations.

```
set system config-management commit-archive location 'tftp://10.0.0.2'
```

Note: the number of revisions don't effect the commit-archive:

A commit look now like this:

```
vyos@vyos-R1# commit
Archiving config...
tftp://10.0.0.2  OK
[edit]
vyos@vyos-R1#
```

The filename has this format: config.boot-hostname.YYYYMMDD_HHMMSS

13.9 Time Zone

To set the system time zone type:

```
[edit]
vyos@vyos# set system time-zone [time-zone]
```

CHAPTER 14

High availability

VRRP (Virtual Redundancy Protocol) provides active/backup redundancy for routers. Every VRRP router has a physical IP/IPv6 address, and a virtual address. On startup, routers elect the master, and the router with the highest priority becomes the master and assigns the virtual address to its interface. All routers with lower priorities become backup routers. The master then starts sending keepalive packets to notify other routers that it's available. If the master fails and stops sending keepalive packets, the router with the next highest priority becomes the new master and takes over the virtual address.

VRRP keepalive packets use multicast, and VRRP setups are limited to a single datalink layer segment. You can setup multiple VRRP groups (also called virtual routers). Virtual routers are identified by a VRID (Virtual Router IDentifier). If you setup multiple groups on the same interface, their VRIDs must be unique, but it's possible (even if not recommended for readability reasons) to use duplicate VRIDs on different interfaces.

14.1 Basic setup

VRRP groups are created with the `set high-availability vrrp group $GROUP_NAME` commands. The required parameters are interface, vrid, and virtual-address.

minimal config

```
set high-availability vrrp group Foo vrid 10
set high-availability vrrp group Foo interface eth0
set high-availability vrrp group Foo virtual-address 192.0.2.1/24
```

You can verify your VRRP group status with the operational mode `run show vrrp` command:

```
vyos@vyos# run show vrrp
Name        Interface      VRID  State    Last Transition
----------  -----------  ------  -------  -----------------
Foo         eth1             10  MASTER   2s
```

14.2 IPv6 support

The `virtual-address` parameter can be either an IPv4 or IPv6 address, but you cannot mix IPv4 and IPv6 in the same group, and will need to create groups with different VRIDs specially for IPv4 and IPv6.

14.3 Disabling a VRRP group

You can disable a VRRP group with `disable` option:

```
set high-availability vrrp group Foo disable
```

A disabled group will be removed from the VRRP process and your router will not participate in VRRP for that VRID. It will disappear from operational mode commands output, rather than enter the backup state.

14.4 Setting VRRP group priority

VRRP priority can be set with `priority` option:

```
set high-availability vrrp group Foo priority 200
```

The priority must be an interger number from 1 to 255. Higher priority value increases router's precedence in the master elections.

14.5 Preemption

VRRP can use two modes: preemptive and non-preemptive. In the preemptive mode, if a router with a higher priority fails and then comes back, routers with lower priority will give up their master status. In non-preemptive mode, the newly elected master will keep the master status and the virtual address indenfinitely.

By default VRRP uses preemption. You can disable it with the "no-preempt" option:

```
set high-availability vrrp group Foo no-preempt
```

You can also configure the time interval for preemption with the "preempt-delay" option. For example, to set the higher priority router to take over in 180 seconds, use:

```
set high-availability vrrp group Foo preempt-delay 180
```

14.6 Unicast VRRP

By default VRRP uses multicast packets. If your network does not support multicast for whatever reason, you can make VRRP use unicast communication instead.

```
set high-availability vrrp group Foo peer-address 192.0.2.10
set high-availability vrrp group Foo hello-source-address 192.0.2.15
```

14.7 Scripting

VRRP functionality can be extended with scripts. VyOS supports two kinds of scripts: health check scripts and transition scripts. Health check scripts execute custom checks in addition to the master router reachability. Transition scripts are executed when VRRP state changes from master to backup or fault and vice versa and can be used to enable or disable certain services, for example.

14.7.1 Health check scripts

This setup will make the VRRP process execute the `/config/scripts/vrrp-check.sh script` every 60 seconds, and transition the group to the fault state if it fails (i.e. exits with non-zero status) three times:

```
set high-availability vrrp group Foo health-check script /config/scripts/vrrp-check.sh
set high-availability vrrp group Foo health-check interval 60
set high-availability vrrp group Foo health-check failure-count 3
```

14.7.2 Transition scripts

Transition scripts can help you implement various fixups, such as starting and stopping services, or even modifying the VyOS config on VRRP transition. This setup will make the VRRP process execute the `/config/scripts/vrrp-fail.sh` with argument `Foo` when VRRP fails, and the `/config/scripts/vrrp-master.sh` when the router becomes the master:

```
set high-availability vrrp group Foo transition-script backup "/config/scripts/vrrp-
↪fail.sh Foo"
set high-availability vrrp group Foo transition-script fault "/config/scripts/vrrp-
↪fail.sh Foo"
set high-availability vrrp group Foo transition-script master "/config/scripts/vrrp-
↪master.sh Foo"
```

CHAPTER 15

Clustering

The cluster feature allows 2 vyos routers to share IP adresses and various services.

VyOS supports multicast clustering.

Note: Please follow the process of the cluster function here. https://phabricator.vyos.net/T985

15.1 General cluster configuration

In the general cluster configuration the network interfaces used for monitoring and negotiation of the cluster health is defined. Additionally, the communication interval settings, multicast group (for sending/receiving heartbeat messages), and pre-shared secret used in this monitoring is defined.

```
vyos@vyos# set cluster
Possible completions:
  dead-interval            Interval after which a node is considered dead after␣
↪missing heartbeats (milliseconds)
+> group                   Name of resource group for clustering [REQUIRED]
+  interface               Interface(s) for sending/receiving heartbeat packets␣
↪[REQUIRED]
  keepalive-interval       Time interval between heartbeat packets (milliseconds)
  mcast-group              Multicast group for sending/receiving heartbeat packets
  monitor-dead-interval    Interval after which a monitor node is considered dead␣
↪(milliseconds)
  pre-shared-secret        Pre-shared secret for authentication between cluster nodes␣
↪[REQUIRED]
```

15.2 Cluster group configuration

For the cluster group configuration, the group name must be defined before the groups configuration can be set (See Example below). After the group name is defined, the specific service to be clustered between primary and secondary nodes is configured.

```
vyos@vyos# set cluster group GROUPNAME
Possible completions:
  auto-failback        Fail back to primary node if it recovers from failure
+  monitor             IP address(es) for monitoring connectivity
  primary              Host name of the primary node [REQUIRED]
+  secondary           Host name(s) of the secondary node(s) [REQUIRED]
+  service             IP address(es) or service name(s) in this resource group␣
↪[REQUIRED]
```

15.3 Review cluster status

```
vyos@vyos:~$ show cluster status
```

15.4 Example

In the example below SSH is clustered between two nodes.

```
cluster {
    dead-interval 20000
    group cluster {
        auto-failback false
        primary node1
        secondary node2
        service ssh
        service 192.168.0.123/24/eth0
    }
    interface eth0
    keepalive-interval 5000
    monitor-dead-interval 20000
    pre-shared-secret S3cr#t
}
```

CHAPTER 16

WAN load balancing

Outbound traffic can be balanced between two or more outbound interfaces. If a path fails, traffic is balanced across the remaining healthy paths, a recovered path is automatically added back to the routing table and used by the load balancer. The load balancer automatically adds routes for each path to the routing table and balances traffic across the configured interfaces, determined by interface health and weight.

In a minimal, configuration the following must be provided:

- a interface with a nexthop
- one rule with a LAN (inbound-interface) and the WAN (interface).

lets assume we have two dhcp WAN interfaces and one LAN (eth2)

```
set load-balancing wan interface-health eth0 nexthop 'dhcp'
set load-balancing wan interface-health eth1 nexthop 'dhcp'
set load-balancing wan rule 0 inbound-interface 'eth2'
set load-balancing wan rule 0 interface eth0
set load-balancing wan rule 0 interface eth1
```

16.1 Balancing Rules

Interfaces, their weight and the type of traffic to be balanced are defined in numbered balancing rule sets. The rule sets are executed in numerical order against outgoing packets. In case of a match the packet is sent through an interface specified in the matching rule. If a packet doesn't match any rule it is sent by using the system routing table. Rule numbers can't be changed.

Create a load balancing rule, rule can be a number between 0. . . 4294967295:

```
vyos@vyos# set load-balancing wan rule 0
Possible completions:
description             Description for this rule
> destination           Destination
exclude                 Exclude packets matching this rule from wan load balance
```

(continues on next page)

(continued from previous page)

```
failover                Enable failover for packets matching this rule from wan load␣
↪balance
inbound-interface       Inbound interface name (e.g., "eth0") [REQUIRED]
+> interface            Interface name [REQUIRED]
> limit                 Enable packet limit for this rule
per-packet-balancing    Option to match traffic per-packet instead of the default,␣
↪per-flow
protocol                Protocol to match
> source                Source information
```

16.1.1 Interface weight

Let's expand the example from above and add a weight to the interfaces. The bandwidth from eth0 is larger than eth1. Per default outbound traffic is distributed randomly across available interfaces. Weights can be assigned to interfaces to influence the balancing.

```
set load-balancing wan rule 0 interface eth0 weight 2
set load-balancing wan rule 0 interface eth1 weight 1
```

66% traffic is routed to eth0 and eth1 get 33% of traffic.

16.1.2 Rate limit

A packet rate limit can be set for a rule to apply the rule to traffic above or below a specified threshold. To configure the rate limiting use:

```
set load-balancing wan rule <rule> limit <parameter>
```

- `burst`: Number of packets allowed to overshoot the limit within `period`. Default 5.
- `period`: Time window for rate calculation. Possible values: `second` (one second), `minute` (one minute), `hour` (one hour). Default is `second`.
- `rate`: Number of packets. Default 5.
- `threshold`: `below` or `above` the specified rate limit.

16.1.3 Flow and packet-based balancing

Outgoing traffic is balanced in a flow-based manner. A connection tracking table is used to track flows by their source address, destination address and port. Each flow is assigned to an interface according to the defined balancing rules and subsequent packets are sent through the same interface. This has the advantage that packets always arrive in order if links with different speeds are in use.

Packet-based balancing can lead to a better balance across interfaces when out of order packets are no issue. Per-packet-based balancing can be set for a balancing rule with:

```
set load-balancing wan rule <rule> per-packet-balancing
```

16.1.4 Exclude traffic

To exclude traffic from load balancing, traffic matching an exclude rule is not balanced but routed through the system routing table instead:

```
set load-balancing wan rule <rule> exclude
```

16.2 Health checks

The health of interfaces and paths assigned to the load balancer is periodically checked by sending ICMP packets (ping) to remote destinations, a TTL test or the execution of a user defined script. If an interface fails the health check it is removed from the load balancer's pool of interfaces. To enable health checking for an interface:

```
vyos@vyos# set load-balancing wan interface-health <interface>
Possible completions:
failure-count    Failure count
nexthop          Outbound interface nexthop address. Can be 'dhcp or ip address'␣
↪[REQUIRED]
success-count    Success count
+> test          Rule number
```

Specify nexthop on the path to destination, `ipv4-address` can be set to `dhcp`

```
set load-balancing wan interface-health <interface> nexthop <ipv4-address>
```

Set the number of health check failures before an interface is marked as unavailable, range for number is 1 to 10, default 1. Or set the number of successful health checks before an interface is added back to the interface pool, range for number is 1 to 10, default 1.

```
set load-balancing wan interface-health <interface> failure-count <number>
set load-balancing wan interface-health <interface> success-count <number>
```

Each health check is configured in its own test, tests are numbered and processed in numeric order. For multi target health checking multiple tests can be defined:

```
vyos@vyos# set load-balancing wan interface-health eth1 test 0
Possible completions:
resp-time    Ping response time (seconds)
target       Health target address
test-script  Path to user defined script
ttl-limit    Ttl limit (hop count)
type         WLB test type
```

- `resp-time`: the maximum response time for ping in seconds. Range 1…30, default 5
- `target`: the target to be sent ICMP packets to, address can be an IPv4 address or hostname
- `test-script`: A user defined script must return 0 to be considered successful and non-zero to fail. Scripts are located in /config/scripts, for different locations the full path needs to be provided
- `ttl-limit`: For the UDP TTL limit test the hop count limit must be specified. The limit must be shorter than the path length, an ICMP time expired message is needed to be returned for a successful test. default 1
- `type`: Specify the type of test. type can be ping, ttl or a user defined script

16.3 Source NAT rules

Per default, interfaces used in a load balancing pool replace the source IP of each outgoing packet with its own address to ensure that replies arrive on the same interface. This works through automatically generated source NAT (SNAT) rules, these rules are only applied to balanced traffic. In cases where this behaviour is not desired, the automatic generation of SNAT rules can be disabled:

```
set load-balancing wan disable-source-nat
```

16.4 Sticky Connections

Upon reception of an incoming packet, when a response is sent, it might be desired to ensure that it leaves from the same interface as the inbound one. This can be achieved by enabling sticky connections in the load balancing:

```
set load-balancing wan sticky-connections inbound
```

16.5 Failover

In failover mode, one interface is set to be the primary interface and other interfaces are secondary or spare. Instead of balancing traffic across all healthy interfaces, only the primary interface is used and in case of failure, a secondary interface selected from the pool of available interfaces takes over. The primary interface is selected based on its weight and health, others become secondary interfaces. Secondary interfaces to take over a failed primary interface are chosen from the load balancer's interface pool, depending on their weight and health. Interface roles can also be selected based on rule order by including interfaces in balancing rules and ordering those rules accordingly. To put the load balancer in failover mode, create a failover rule:

```
set load-balancing wan rule <number> failover
```

Because existing sessions do not automatically fail over to a new path, the session table can be flushed on each connection state change:

```
set load-balancing wan flush-connections
```

> **Warning:** Flushing the session table will cause other connections to fall back from flow-based to packet-based balancing until each flow is reestablished.

16.6 Script execution

A script can be run when an interface state change occurs. Scripts are run from /config/scripts, for a different location specify the full path:

```
set load-balancing wan hook script-name
```

Two environment variables are available:

- `WLB_INTERFACE_NAME=[interfacename]`: Interface to be monitored
- `WLB_INTERFACE_STATE=[ACTIVE|FAILED]`: Interface state

Warning: Blocking call with no timeout. System will become unresponsive if script does not return!

16.7 Handling and monitoring

Show WAN load balancer information including test types and targets. A character at the start of each line depicts the state of the test

- + successful
- – failed
- a blank indicates that no test has been carried out

```
vyos@vyos:~$ show wan-load-balance
Interface:  eth0
Status:  failed
Last Status Change:  Tue Jun 11 20:12:19 2019
-Test:  ping  Target:
    Last Interface Success:  55s
    Last Interface Failure:  0s
    # Interface Failure(s):  5

Interface:  eth1
Status:  active
Last Status Change:  Tue Jun 11 20:06:42 2019
+Test:  ping  Target:
    Last Interface Success:  0s
    Last Interface Failure:  6m26s
    # Interface Failure(s):  0
```

Show connection data of load balanced traffic:

```
vyos@vyos:~$ show wan-load-balance connection
conntrack v1.4.2 (conntrack-tools): 3 flow entries have been shown.
Type    State           Src                     Dst                     Packets Bytes
tcp     TIME_WAIT       10.1.1.13:38040         203.0.113.2:80          203.0.113.2 ␣
↪192.168.188.71
udp                     10.1.1.13:41891         198.51.100.3:53         198.51.100.3␣
↪192.168.188.71
udp                     10.1.1.13:55437         198.51.100.3:53         198.51.100.3␣
↪192.168.188.71
```

16.7.1 Restart

```
restart wan-load-balance
```

CHAPTER 17

System Image Management

The VyOS image-based installation is implemented by creating a directory for each image on the storage device selected during the install process.

The directory structure of the boot device:

```
/
/boot
/boot/grub
/boot/1.2.0-rolling+201810021347
```

The image directory contains the system kernel, a compressed image of the root filesystem for the OS, and a directory for persistent storage, such as configuration.

On boot, the system will extract the OS image into memory and mount the appropriate live-rw sub-directories to provide persistent storage system configuration.

This process allows for a system to always boot to a known working state, as the OS image is fixed and non-persistent. It also allows for multiple releases of VyOS to be installed on the same storage device.

The image can be selected manually at boot if needed, but the system will otherwise boot the image configured to be the default.

The default boot image can be set using the `set system image default-boot` command in operational mode.

A list of available images can be shown using the `show system image` command in operational mode.

```
vyos@vyos:~$ show system image
The system currently has the following image(s) installed:

   1: 1.2.0-rolling+201810021347 (default boot)
   2: 1.2.0-rolling+201810021217
   3: 1.2.0-rolling+201809280337
   4: 1.2.0-rolling+201809252218
   5: 1.2.0-rolling+201809192034
   6: 1.2.0-rolling+201809191744
   7: 1.2.0-rolling+201809150337
```

(continues on next page)

(continued from previous page)

```
   8: 1.2.0-rolling+201809141130
   9: 1.2.0-rolling+201809140949
  10: 1.2.0-rolling+201809131722

vyos@vyos:~$
```

Images no longer needed can be removed using the `delete system image` command.

17.1 Update VyOS Installation

Finally, new system images can be added using the `add system image` command. The add image command will extract the image from the release ISO (either on the local filesystem or remotely if a URL is provided). The image install process will prompt you to use the current system configuration and SSH security keys, allowing for the new image to boot using the current configuration.

```
vyos@vyos:~$ add system image https://downloads.vyos.io/rolling/current/amd64/vyos-1.
↪2.0-rolling%2B201810030440-amd64.iso
Trying to fetch ISO file from https://downloads.vyos.io/rolling/current/amd64/vyos-1.
↪2.0-rolling%2B201810030440-amd64.iso
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  338M  100  338M    0     0  3837k      0  0:01:30  0:01:30 --:--:-- 3929k
ISO download succeeded.
Checking for digital signature file...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
curl: (22) The requested URL returned error: 404 Not Found

Unable to fetch digital signature file.
Do you want to continue without signature check? (yes/no) [yes]
Checking MD5 checksums of files on the ISO image...OK.
Done!

What would you like to name this image? [1.2.0-rolling+201810030440]:

OK.  This image will be named: 1.2.0-rolling+201810030440
We do not have enough disk space to install this image!
We need 344880 KB, but we only have 17480 KB.
Exiting...
```

Note: Rolling releases are not GPG signed, only the real release build will have a proper GPG signature.

Note: VyOS configuration is associated to each image, and each image has a unique copy of its configuration. This is different than a traditional network router where the configuration is shared across all images.

If you need some files from a previous images - take a look inside a `/live` directory.

After reboot you might want to verify the version you are running with `show version`

```
vyos@vyos:~$ show version
Version:          VyOS 1.2.0-rolling+201810030440
Built by:         autobuild@vyos.net
Built on:         Mon 10 Mar 2018 03:37 UTC
Build UUID:       2ed16684-875c-4a19-8a34-1b03099eed35
Build Commit ID:  3305dca496d814

Architecture:     x86_64
Boot via:         installed image
System type:      Microsoft Hyper-V guest

Hardware vendor:  Microsoft Corporation
Hardware model:   Virtual Machine
Hardware S/N:     9705-6585-6578-0429-1204-0427-62
Hardware UUID:    5260b1ce-4028-4d9c-bc5d-4f8425e5c056

Copyright:        VyOS maintainers and contributors
```

CHAPTER 18

Command scripting

VyOS supports executing configuration and operational commands non-interactively from shell scripts.

To include VyOS-specific functions and aliases you need to `source /opt/vyatta/etc/functions/script-template` files at the top of your script.

```
#!/bin/vbash
source /opt/vyatta/etc/functions/script-template

exit
```

18.1 Run configuration commands

Configuration commands are executed just like from a normal config session.

For example, if you want to disable a BGP peer on VRRP transition to backup:

```
#!/bin/vbash
source /opt/vyatta/etc/functions/script-template

configure

set protocols bgp 65536 neighbor 192.168.2.1 shutdown

commit

exit
```

18.2 Run operational commands

Unlike a normal configuration sessions, all operational commands must be prepended with `run`, even if you haven't created a session with configure.

```
#!/bin/vbash
source /opt/vyatta/etc/functions/script-template

run show interfaces

exit
```

18.3 Other script language

If you want to script the configs in a language other than bash you can have your script output commands and then source them in a bash script.

Here is a simple example:

```
#!/usr/bin/env python
print "delete firewall group address-group somehosts"
print "set firewall group address-group somehosts address '1.1.1.1'"
print "set firewall group address-group somehosts address '1.1.1.2'"
```

```
#!/bin/vbash
source /opt/vyatta/etc/functions/script-template

configure
source <(/config/scripts/setfirewallgroup.py)
commit
```

18.4 Executing Configuration Scripts

There is a pitfall when working with configuration scripts. It is tempting to call configuration scripts with "sudo" (i.e., temporary root permissions), because that's the common way on most Linux platforms to call system commands.

On VyOS this will cause the following problem: After modifying the configuration via script like this once, it is not possible to manually modify the config anymore:

```
sudo ./myscript.sh # Modifies config
configure
set ... # Any configuration parameter
```

This will result in the following error message: `Set failed`
If this happens, a reboot is required to be able to edit the config manually again.

To avoid these problems, the proper way is to call a script with the `vyattacfg` group, e.g., by using the `sg` (switch group) command:

```
sg vyattacfg -c ./myscript.sh
```

To make sure that a script is not accidentally called without the `vyattacfg` group, the script can be safeguarded like this:

```
if [ "$(id -g -n)" != 'vyattacfg' ] ; then
    exec sg vyattacfg -c "/bin/vbash $(readlink -f $0) $@"
fi
```

18.5 Postconfig on boot

The `/config/scripts/vyos-postconfig-bootup.script` script is called on boot after the VyOS configuration is fully applied.

Any modifications done to work around unfixed bugs and implement enhancements which are not complete in the VyOS system can be placed here.

The default file looks like this:

```
#!/bin/sh
# This script is executed at boot time after VyOS configuration is fully applied.
# Any modifications required to work around unfixed bugs
# or use services not available through the VyOS CLI system can be placed here.
```

Hint: For configuration/upgrade management issues, modification of this script should be the last option. Always try to find solutions based on CLI commands first.

CHAPTER 19

Release notes

19.1 1.2 (Crux)

19.1.1 1.2.3

1.2.3 is a maintenance and feature backport release made in September 2019.

New features

- HTTP API
- "set service dns forwarding allow-from <IPv4 net|IPv6 net>" option for limiting queries to specific client networks (T1524)
- Functions for checking if a commit is in progress (T1503)
- "set system contig-mangement commit-archive source-address" option (T1543)
- Intel NIC drivers now support receive side scaling and multiqueue (T1554)

Resolved issues

- OSPF max-metric values over 100 no longer causes commit errors (T1209)
- Fixes issue with DNS forwarding not performing recursive lookups on domain specific forwarders (T1333)
- Special characters in VRRP passwords are handled correctly (T1362)
- BGP weight is applied properly (T1377)
- Fixed permission for log files (T1420)
- Wireguard interfaces now support /31 addresses (T1425)
- Wireguard correctly handles firewall marks (T1428)

- DHCPv6 static mappings now work correctly (T1439)
- Flood ping commands now works correctly (T1450)
- Op mode "show firewall" commands now support counters longer than 8 digits (T1460)
- Fixed priority inversion in VTI commands (T1465)
- Fixed remote-as check in the BGP route-reflector-client option (T1468)
- It's now possible to re-create VRRP groups with RFC compatibility mode enabled (T1472)
- Fixed a typo in DHCPv6 server help strings (T1527)
- Unnumbered BGP peers now support VLAN interfaces (T1529)
- Fixed "set system syslog global archive file" command (T1530)
- Multiple fixes in cluster configuration scripts (T1531)
- Fixed missing help text for "service dns" (T1537)
- Fixed input validation in DHCPv6 relay options (T1541)
- It's now possible to create a QinQ interface and a firewall assigned to it in one commit (T1551)
- URL filtering now uses correct rule database path and works again (T1559)
- "show log vpn ipsec" command works again (T1579)
- "show arp interface <intf>" command works again (T1576)
- Fixed regression in L2TP/IPsec server (T1605)
- Netflow/sFlow captures IPv6 traffic correctly (T1613)
- "renew dhcpv6" command now works from op mode (T1616)
- BGP remove-private-as option iBGP vs eBGP check works correctly now (T1642)
- Multiple improvements in name servers and hosts configuration handling (T1540, T1360, T1264, T1623)

Internals

/etc/resolv.conf and /etc/hosts files are now managed by the vyos-hostsd service that listens on a ZMQ socket for update messages.

19.1.2 1.2.2

1.2.2 is a maintenance release made in July 2019.

New features

- Options for per-interface MSS clamping.
- BGP extended next-hop capability
- Relaxed BGP multipath option
- Internal and external options for "remote-as" (accept any AS as long as it's the same to this router or different, respectively)
- "Unnumbered" (interface-based) BGP peers

- BGP no-prepend option
- Additive BGP community option
- OSPFv3 network type option
- Custom arguments for VRRP scripts
- A script for querying values from config files

Resolved issues

- Linux kernel 4.19.54, including a fix for the TCP SACK vulnerability
- VRRP health-check scripts now can use arguments (T1371)
- DNS server addresses coming from a DHCP server are now correctly propagated to resolv.conf (T1497)
- Domain-specific name servers in DNS forwarding are now used for recursive queries (T1469)
- "run show dhcpv6 server leases" now display leases correctly (T1433)
- Deleting "firewall options" node no longer causes errors (T1461)
- Correct hostname is sent to remote syslog again (T1458)
- Board serial number from DMI is correctly displayed in "show version" (T1438)
- Multiple corrections in remote syslog config (T1358, T1355, T1294)
- Fixed missing newline in /etc/hosts (T1255)
- "system domain-name" is correctly included in /etc/resolv.conf (T1174)
- Fixed priority inversion in "interfaces vti vtiX ip" settings (T1465)
- Fixed errors when installing with RAID1 on UEFI machines (T1446)
- Fixed an error on disabling an interfaces that has no address (T1387)
- Fixed deleting VLAN interface with non-default MTU (T1367)
- vyos.config return_effective_values() function now correctly returns a list rather than a string (T1505)

19.1.3 1.2.1

VyOS 1.2.1 is a maintenance release made in April 2019.

Resolved issues

- Package updates: kernel 4.19.32, open-vm-tools 10.3, latest Intel NIC drivers.
- The kernel now includes drivers for various USB serial adapters, which allows people to add a serial console to a machine without onboard RS232, or connect to something else from the router (T1326).
- The collection of network card firmware is now much more extensive.
- VRRP now correctly uses a virtual rather than physical MAC addresses in the RFC-compliant mode (T1271).
- DHCP WPAD URL option works correctly again (T1330)
- Many to many NAT rules now can use source/destination and translation networks of non-matching size (T1312). If 1:1 network bits translation is desired, it's now user's responsibility to check if prefix length matches.

- IPv6 network prefix translation is fixed (T1290).
- Non-alphanumeric characters such as “>” can now be safely used in PPPoE passwords (T1308).
- “show | commands” no longer fails when a config section ends with a leaf node such as “timezone” in “show system | commands” (T1305).
- “show | commands” correctly works in config mode now (T1235).
- VTI is now compatible with the DHCP-interface IPsec option (T1298).
- “show dhcp server statistics” command was broken in latest Crux (T1277).
- An issue with TFTP server refusing to listen on addresses other than loopback was fixed (T1261).
- Template issue that might cause UDP broadcast relay fail to start is fixed (T1224).
- VXLAN value validation is improved (T1067).
- Blank hostnames in DHCP updates no longer can crash DNS forwarding (T1211).
- Correct configuration is now generated for DHCPv6 relays with more than one upstream interface (T1322).
- “relay-agents-packets” option works correctly now (T1234).
- Dynamic DNS data is now cleaned on configuration change (T1231).
- Remote Syslog can now use a fully qualified domain name (T1282).
- ACPI power off works again (T1279).
- Negation in WAN load balancing rules works again (T1247).
- FRR’s staticd now starts on boot correctly (T1218).
- The installer now correctly detects SD card devices (T1296).
- Wireguard peers can be disabled now (T1225).
- The issue with wireguard interfaces impossible to delete is fixed (T1217).
- Unintended IPv6 access is fixed in SNMP configuration (T1160).
- It’s now possible to exclude hosts from the transparent web proxy (T1060).
- An issue with rules impossible to delete from the zone-based firewall is fixed (T484).

19.2 Earlier releases

See the wiki.

CHAPTER 20

Troubleshooting

Sometimes things break or don't work as expected. This section describes several troubleshooting tools provided by VyOS that can help when something goes wrong.

20.1 Basic Connectivity Verification

Verifying connectivity can be done with the familiar *ping* and *traceroute* commands. The options for each are shown (the options for each command were displayed using the built-in help as described in the *Command-Line Interface* section and are omitted from the output here):

```
vyos@vyos:~$ ping
Possible completions:
  <hostname>    Send Internet Control Message Protocol (ICMP) echo request
  <x.x.x.x>
  <h:h:h:h:h:h:h:h>
```

Several options are available when more extensive troubleshooting is needed:

```
vyos@vyos:~$ ping 8.8.8.8
Possible completions:
  <Enter>       Execute the current command
  adaptive      Ping options
  allow-broadcast
  audible
  bypass-route
  count
  deadline
  flood
  interface
  interval
  mark
  no-loopback
  numeric
```

(continues on next page)

(continued from previous page)

```
  pattern
  quiet
  record-route
  size
  timestamp
  tos
  ttl
  verbose
```

```
vyos@vyos:~$ traceroute
Possible completions:
  <hostname>    Track network path to specified node
  <x.x.x.x>
  <h:h:h:h:h:h:h:h>
  ipv4          Track network path to <hostname|IPv4 address>
  ipv6          Track network path to <hostname|IPv6 address>
```

However, another tool, mtr, is available which combines ping and traceroute into a single tool. An example of its output is shown:

```
vyos@vyos:~$ mtr 10.62.212.12

                           My traceroute  [v0.85]
vyos (0.0.0.0)
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                 Packets               Pings
Host                            Loss%   Snt   Last   Avg  Best  Wrst StDev
1. 10.11.110.4                   0.0%    34    0.5   0.5   0.4   0.8   0.1
2. 10.62.255.184                 0.0%    34    1.1   1.0   0.9   1.4   0.1
3. 10.62.255.71                  0.0%    34    1.4   1.4   1.3   2.0   0.1
4. 10.62.212.12                  0.0%    34    1.6   1.6   1.6   1.7   0.0
```

Note: The output of `mtr` consumes the screen and will replace your command prompt.

Several options are available for changing the display output. Press *h* to invoke the built in help system. To quit, just press *q* and you'll be returned to the VyOS command prompt.

20.2 Monitoring

20.2.1 Network Interfaces

It's possible to monitor network traffic, either at the flow level or protocol level. This can be useful when troubleshooting a variety of protocols and configurations. The following interface types can be monitored:

```
vyos@vyos:~$ monitor interfaces
Possible completions:
  <Enter>       Execute the current command
  bonding       Monitor a bonding interface
  bridge        Monitor a bridge interface
  ethernet      Monitor a ethernet interface
  loopback      Monitor a loopback interface
```

(continues on next page)

(continued from previous page)

```
  openvpn       Monitor an openvpn interface
  pppoe         Monitor pppoe interface
  pseudo-ethernet
                Monitor a pseudo-ethernet interface
  tunnel        Monitor a tunnel interface
  vrrp          Monitor a vrrp interface
  vti           Monitor a vti interface
  wireless      Monitor wireless interface
```

To monitor traffic flows, issue the `monitor interfaces <type> <name> flow` command, replacing *<type>* and *<name>* with your desired interface type and name, respectively. Output looks like the following:

```
                    12.5Kb              25.0Kb              37.5Kb              50.0Kb␣
↪       62.5Kb
????????????????????????????????????????????????????????????????????????????????????????????????????
↪?????????????
10.11.111.255                        => 10.11.110.37                          0b    ␣
↪   0b      0b
                                    <=                                      624b    ␣
↪749b    749b
10.11.110.29                         => 10.62.200.11                          0b    ␣
↪198b    198b
                                    <=                                        0b    ␣
↪356b    356b
255.255.255.255                      => 10.11.110.47                          0b    ␣
↪   0b      0b
                                    <=                                      724b    ␣
↪145b    145b
10.11.111.255                        => 10.11.110.47                          0b    ␣
↪   0b      0b
                                    <=                                      724b    ␣
↪145b    145b
10.11.111.255                        => 10.11.110.255                         0b    ␣
↪   0b      0b
                                    <=                                      680b    ␣
↪136b    136b
????????????????????????????????????????????????????????????????????????????????????????????????????
↪?????????????
TX:             cumm:  26.7KB   peak:   40.6Kb                      rates:   23.2Kb ␣
↪21.4Kb  21.4Kb
RX:                    67.5KB           63.6Kb                              54.6Kb ␣
↪54.0Kb  54.0Kb
TOTAL:                 94.2KB            104Kb                              77.8Kb ␣
↪75.4Kb  75.4Kb
```

Several options are available for changing the display output. Press *h* to invoke the built in help system. To quit, just press *q* and you'll be returned to the VyOS command prompt.

To monitor interface traffic, issue the `monitor interfaces <type> <name> traffic` command, replacing *<type>* and *<name>* with your desired interface type and name, respectively. This command invokes the familiar tshark utility and the following options are available:

```
vyos@vyos:~$ monitor interfaces ethernet eth0 traffic
Possible completions:
  <Enter>       Execute the current command
  detail        Monitor detailed traffic for the specified ethernet interface
  filter        Monitor filtered traffic for the specified ethernet interface
```

(continues on next page)

(continued from previous page)

```
  save          Save monitored traffic to a file
  unlimited     Monitor traffic for the specified ethernet interface
```

To quit monitoring, press *Ctrl-c* and you'll be returned to the VyOS command prompt. The *detail* keyword provides verbose output of the traffic seen on the monitored interface. The *filter* keyword accepts valid PCAP filter expressions, enclosed in single or double quotes (e.g. "port 25" or "port 161 and udp"). The *save* keyword allows you to save the traffic dump to a file. The *unlimited* keyword is used to specify that an unlimited number of packets can be captured (by default, 1,000 packets are captured and you're returned to the VyOS command prompt).

20.2.2 Interface Bandwith

to take a quick view on the used bandwith of an interface use the `monitor bandwith` command

```
vyos@vyos:~$ monitor bandwidth interface eth0
```

show the following:

```
 eth0                                                                                                  ␣
↪                          bmon 3.5
Interfaces                      │ RX bps       pps     %│ TX bps       pps     %
 >eth0                          │    141B        2      │    272B        1
────────────────────────────────────────────────────────────────────────────────────────────────────────────
       B                       (RX Bytes/second)
  198.00 .|....|.................................................
  165.00 .|....|.................................................
  132.00 ||..|.|.................................................
   99.00 ||..|.|.................................................
   66.00 ||||||..................................................
   33.00 ||||||..................................................
         1   5   10   15   20   25   30   35   40   45   50   55   60
     KiB                       (TX Bytes/second)
    3.67 ......|...................................................
    3.06 ......|...................................................
    2.45 ......|...................................................
    1.84 ......|...................................................
    1.22 ......|...................................................
    0.61 :::::||...................................................
         1   5   10   15   20   25   30   35   40   45   50   55   60

──────────────────────────────────────── Press d to enable detailed statistics␣
↪────────────────────────────────────────
──────────────────────────────────────── Press i to enable additional information␣
↪────────────────────────────────────────
 Wed Apr  3 14:46:59 2019                                                                              ␣
↪                 Press ? for help
```

Press `d` for more detailed informations or `i` for additional information.
To exit press `q` and than `y`

20.2.3 Interface performance

To take a look on the network bandwith between two nodes, the `monitor bandwidth-test` command is used to run iperf.

```
vyos@vyos:~$ monitor bandwidth-test
Possible completions:
  accept        Wait for bandwidth test connections (port TCP/5001)
  initiate      Initiate a bandwidth test
```

The `accept` command open a listen iperf server on TCP Port 5001
The `initiate` command conncet to this server.

```
vyos@vyos:~$ monitor bandwidth-test initiate
Possible completions:
  <hostname>    Initiate a bandwidth test to specified host (port TCP/5001)
  <x.x.x.x>
  <h:h:h:h:h:h:h:h>
```

20.2.4 Monitor command

The `monitor command` command allows you to repeatedly run a command to view a continuously refreshed output. The command is run and output every 2 seconds, allowing you to monitor the output continuously without having to re-run the command. This can be useful to follow routing adjacency formation.

```
vyos@router:~$ monitor command "show interfaces"
```

Will clear the screen and show you the output of `show interfaces` every 2 seconds.

```
Every 2.0s: /opt/vyatta/bin/vyatta-op-cmd-wrapper s...  Sun Mar 26 02:49:46 2019

Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.1.1/24                    u/u
eth0.5           198.51.100.4/24                   u/u  WAN
lo               127.0.0.1/8                       u/u
                 ::1/128
vti0             172.32.254.2/30                   u/u
vti1             172.32.254.9/30                   u/u
```

20.3 Clear Command

Sometimes you need to clear counters or statistics to troubleshoot better.

To do this use the `clear` command in Operational mode.

to clear the console output

```
vyos@vyos:~$ clear console
```

to clear interface counters

```
# clear all interfaces
vyos@vyos:~$ clear interface ethernet counters
# clear specific interface
vyos@vyos:~$ clear interface ehternet eth0 counters
```

The command follow the same logic as the `set` command in configuration mode.

```
# clear all counters of a interface type
vyos@vyos:~$ clear interface <interface_type> counters
# clear counter of a interface in interface_type
vyos@vyos:~$ clear interface <interface_type> <interace_name> counters
```

to clear counters on firewall rulesets or single rules

```
vyos@vyos:~$ clear firewall name <ipv4 ruleset name> counters
vyos@vyos:~$ clear firewall name <ipv4 ruleset name> rule <rule#> counters

vyos@vyos:~$ clear firewall ipv6-name <ipv6 ruleset name> counters
vyos@vyos:~$ clear firewall ipv6-name <ipv6 ruleset name> rule <rule#> counters
```

20.4 Basic System Information

20.4.1 Boot steps

VyOS 1.2.0+ uses Debian Jessie as the base Linux operating system. Jessie was the first version of Debian that uses systemd as the default init system.

These are the boot steps for VyOS 1.2.0+

1. The BIOS loads Grub (or isolinux for the Live CD)
2. Grub then starts the Linux boot and loads the Linux Kernel `/boot/vmlinuz`
3. Kernel Launches Systemd `/lib/systemd/systemd`
4. Systemd loads the VyOS service file `/lib/systemd/system/vyos-router.service`
5. The service file launches the VyOS router init script `/usr/libexec/vyos/init/vyos-router` - this is part of the vyatta-cfg Debian package
    1. Starts FRR - successor to GNU Zebra and Quagga
    2. Initialises the boot configuration file - copies over `config.boot.default` if there is no configuration
    3. Runs the configuration migration, if the configuration is for an older version of VyOS
    4. Runs The pre-config script, if there is one `/config/scripts/vyos-preconfig-bootup.script`
    5. If the config file was upgraded, runs any post upgrade scripts `/config/scripts/post-upgrade.d`
    6. Starts **rl-system** and **firewall**
    7. Mounts the `/boot` partition

8. The boot configuration file is then applied by `/opt/vyatta/sbin/vyatta-boot-config-loader /opt/vyatta/etc/config/config.boot`
1. The config loader script writes log entries to `/var/log/vyatta-config-loader.log`
10. Runs `telinit q` to tell the init system to reload `/etc/inittab`
11. Finally it runs the post-config script `/config/scripts/vyos-postconfig-bootup.script`

CHAPTER 21

Configuration Examples

This chapter contains various configuration Examples

21.1 VyOS DMVPN Hub

General infomration can be found in the *DMVPN* chapter.

21.1.1 Configuration

```
set interfaces tunnel tun100 address '172.16.253.134/29'
set interfaces tunnel tun100 encapsulation 'gre'
set interfaces tunnel tun100 local-ip '11.22.33.44'
set interfaces tunnel tun100 multicast 'enable'
set interfaces tunnel tun100 parameters ip key '1'

set protocols nhrp tunnel tun100 cisco-authentication '<nhrp secret key>'
set protocols nhrp tunnel tun100 holding-time '300'
set protocols nhrp tunnel tun100 multicast 'dynamic'
set protocols nhrp tunnel tun100 redirect
set protocols nhrp tunnel tun100 shortcut

set vpn ipsec esp-group ESP-HUB compression 'disable'
set vpn ipsec esp-group ESP-HUB lifetime '1800'
set vpn ipsec esp-group ESP-HUB mode 'tunnel'
set vpn ipsec esp-group ESP-HUB pfs 'dh-group2'
set vpn ipsec esp-group ESP-HUB proposal 1 encryption 'aes256'
set vpn ipsec esp-group ESP-HUB proposal 1 hash 'sha1'
set vpn ipsec esp-group ESP-HUB proposal 2 encryption '3des'
set vpn ipsec esp-group ESP-HUB proposal 2 hash 'md5'
set vpn ipsec ike-group IKE-HUB ikev2-reauth 'no'
set vpn ipsec ike-group IKE-HUB key-exchange 'ikev1'
```

(continues on next page)

(continued from previous page)

```
set vpn ipsec ike-group IKE-HUB lifetime '3600'
set vpn ipsec ike-group IKE-HUB proposal 1 dh-group '2'
set vpn ipsec ike-group IKE-HUB proposal 1 encryption 'aes256'
set vpn ipsec ike-group IKE-HUB proposal 1 hash 'sha1'
set vpn ipsec ike-group IKE-HUB proposal 2 dh-group '2'
set vpn ipsec ike-group IKE-HUB proposal 2 encryption 'aes128'
set vpn ipsec ike-group IKE-HUB proposal 2 hash 'sha1'
set vpn ipsec ipsec-interfaces interface 'eth0'

set vpn ipsec profile NHRPVPN authentication mode 'pre-shared-secret'
set vpn ipsec profile NHRPVPN authentication pre-shared-secret '<secretkey>'
set vpn ipsec profile NHRPVPN bind tunnel 'tun100'
set vpn ipsec profile NHRPVPN esp-group 'ESP-HUB'
set vpn ipsec profile NHRPVPN ike-group 'IKE-HUB'
```

21.1.2 Cisco IOS Spoke

This example is verified with a Cisco 2811 platform running IOS 15.1(4)M9 and VyOS 1.1.7 (helium) up to VyOS 1.2 (Crux).

```
Cisco IOS Software, 2800 Software (C2800NM-ADVENTERPRISEK9-M), Version 15.1(4)M9,␣
↪RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2014 by Cisco Systems, Inc.
Compiled Fri 12-Sep-14 10:45 by prod_rel_team

ROM: System Bootstrap, Version 12.3(8r)T7, RELEASE SOFTWARE (fc1)
```

Use this configuration on your Cisco device:

```
crypto pki token default removal timeout 0
crypto keyring DMVPN
  pre-shared-key address 1.2.3.4 key <secretkey>
!
crypto isakmp policy 10
 encr aes 256
 authentication pre-share
 group 2
!
crypto isakmp invalid-spi-recovery
crypto isakmp keepalive 30 30 periodic
crypto isakmp profile DMVPN
   keyring DMVPN
   match identity address 11.22.33.44 255.255.255.255
!
crypto ipsec transform-set DMVPN-AES256 esp-aes 256 esp-sha-hmac
 mode transport
!
crypto ipsec profile DMVPN
 set security-association idle-time 720
 set transform-set DMVPN-AES256
 set isakmp-profile DMVPN
!
interface Tunnel10
 description Tunnel to DMVPN HUB
```

(continues on next page)

(continued from previous page)

```
 ip address 172.16.253.129 255.255.255.248
 no ip redirects
 ip nhrp authentication <nhrp secret key>
 ip nhrp map multicast 11.22.33.44
 ip nhrp map 172.16.253.134 11.22.33.44
 ip nhrp network-id 1
 ip nhrp holdtime 600
 ip nhrp nhs 172.16.253.134
 ip nhrp registration timeout 75
 tunnel source Dialer1
 tunnel mode gre multipoint
 tunnel key 1
```

21.2 Zone-Policy example

21.2.1 Native IPv4 and IPv6

We have three networks.

```
WAN - 172.16.10.0/24, 2001:0DB8:0:9999::0/64
LAN - 192.168.100.0/24, 2001:0DB8:0:AAAA::0/64
DMZ - 192.168.200.0/24, 2001:0DB8:0:BBBB::0/64
```

This specific example is for a router on a stick, but is very easily adapted for however many NICs you have.

[http://imgur.com/Alz1J.png Topology Image]

The VyOS interface is assigned the .1/:1 address of their respective networks. WAN is on VLAN 10, LAN on VLAN 20, and DMZ on VLAN 30.

It will look something like this:

```
interfaces {
    ethernet eth0 {
        duplex auto
        hw-id 00:0c:29:6e:2a:92
        smp_affinity auto
        speed auto
        vif 10 {
            address 172.16.10.1/24
            address 2001:db8:0:9999::1/64
        }
        vif 20 {
            address 192.168.100.1/24
            address 2001:db8:0:AAAA::1/64
        }
        vif 30 {
            address 192.168.200.1/24
            address 2001:db8:0:BBBB::1/64
        }
    }
    loopback lo {
    }
}
```

21.2.2 Zones Basics

Each interface is assigned to a zone. The interface can be physical or virtual such as tunnels (VPN, pptp, gre, etc) and are treated exactly the same.

Traffic flows from zone A to zone B. That flow is what I refer to as a zone-pair-direction. eg. A->B and B->A are two zone-pair-destinations.

Ruleset are created per zone-pair-direction.

I name rule sets to indicate which zone-pair-direction they represent. eg. ZoneA-ZoneB or ZoneB-ZoneA. LAN-DMZ, DMZ-LAN.

In VyOS, you have to have unique Ruleset names. In the event of overlap, I add a "-6" to the end of v6 rulesets. eg. LAN-DMZ, LAN-DMZ-6. This allows for each auto-completion and uniqueness.

In this example we have 4 zones. LAN, WAN, DMZ, Local. The local zone is the firewall itself.

If your computer is on the LAN and you need to SSH into your VyOS box, you would need a rule to allow it in the LAN-Local ruleset. If you want to access a webpage from your VyOS box, you need a rule to allow it in the Local-LAN ruleset.

In rules, it is good to keep them named consistently. As the number of rules you have grows, the more consistency you have, the easier your life will be.

```
Rule 1 - State Established, Related
Rule 2 - State Invalid
Rule 100 - ICMP
Rule 200 - Web
Rule 300 - FTP
Rule 400 - NTP
Rule 500 - SMTP
Rule 600 - DNS
Rule 700 - DHCP
Rule 800 - SSH
Rule 900 - IMAPS
```

The first two rules are to deal with the idiosyncrasies of VyOS and iptables.

Zones and Rulesets both have a default action statement. When using Zone-Policies, the default action is set by the zone-policy statement and is represented by rule 10000.

It is good practice to log both accepted and denied traffic. It can save you significant headaches when trying to troubleshoot a connectivity issue.

To add logging to the default rule, do:

```
set firewall name <ruleSet> enable-default-log
```

By default, iptables does not allow traffic for established session to return, so you must explicitly allow this. I do this by adding two rules to every ruleset. 1 allows established and related state packets through and rule 2 drops and logs invalid state packets. We place the established/related rule at the top because the vast majority of traffic on a network is established and the invalid rule to prevent invalid state packets from mistakenly being matched against other rules. Having the most matched rule listed first reduces CPU load in high volume environments. Note: I have filed a bug to have this added as a default action as well.

''It is important to note, that you do not want to add logging to the established state rule as you will be logging both the inbound and outbound packets for each session instead of just the initiation of the session. Your logs will be massive in a very short period of time.''

In VyOS you must have the interfaces created before you can apply it to the zone and the rulesets must be created prior to applying it to a zone-policy.

I create/configure the interfaces first. Build out the rulesets for each zone-pair-direction which includes at least the three state rules. Then I setup the zone-policies.

Zones do not allow for a default action of accept; either drop or reject. It is important to remember this because if you apply an interface to a zone and commit, any active connections will be dropped. Specifically, if you are SSH'd into VyOS and add local or the interface you are connecting through to a zone and do not have rulesets in place to allow SSH and established sessions, you will not be able to connect.

The following are the rules that were created for this example (may not be complete), both in IPv4 and IPv6. If there is no IP specified, then the source/destination address is not explicit.

```
WAN - DMZ:192.168.200.200 - tcp/80
WAN - DMZ:192.168.200.200 - tcp/443
WAN - DMZ:192.168.200.200 - tcp/25
WAN - DMZ:192.168.200.200 - tcp/53
WAN - DMZ:2001:0DB8:0:BBBB::200 - tcp/80
WAN - DMZ:2001:0DB8:0:BBBB::200 - tcp/443
WAN - DMZ:2001:0DB8:0:BBBB::200 - tcp/25
WAN - DMZ:2001:0DB8:0:BBBB::200 - tcp/53

DMZ - Local - tcp/53
DMZ - Local - tcp/123
DMZ - Local - tcp/67,68

LAN - Local - tcp/53
LAN - Local - tcp/123
LAN - Local - tcp/67,68
LAN:192.168.100.10 - Local - tcp/22
LAN:2001:0DB8:0:AAAA::10 - Local - tcp/22

LAN - WAN - tcp/80
LAN - WAN - tcp/443
LAN - WAN - tcp/22
LAN - WAN - tcp/20,21

DMZ - WAN - tcp/80
DMZ - WAN - tcp/443
DMZ - WAN - tcp/22
DMZ - WAN - tcp/20,21
DMZ - WAN - tcp/53
DMZ - WAN - udp/53

Local - WAN - tcp/80
Local - WAN - tcp/443
Local - WAN - tcp/20,21

Local - DMZ - tcp/25
Local - DMZ - tcp/67,68
Local - DMZ - tcp/53
Local - DMZ - udp/53

Local - LAN - tcp/67,68

LAN - DMZ - tcp/80
LAN - DMZ - tcp/443
LAN - DMZ - tcp/993
```

(continues on next page)

(continued from previous page)

```
LAN:2001:0DB8:0:AAAA::10 - DMZ:2001:0DB8:0:BBBB::200 - tcp/22
LAN:192.168.100.10 - DMZ:192.168.200.200 - tcp/22
```

Since we have 4 zones, we need to setup the following rulesets.

```
Lan-wan
Lan-local
Lan-dmz
Wan-lan
Wan-local
Wan-dmz
Local-lan
Local-wan
Local-dmz
Dmz-lan
Dmz-wan
Dmz-local
```

Even if the two zones will never communicate, it is a good idea to create the zone-pair-direction rulesets and set enable-default-log. This will allow you to log attempts to access the networks. Without it, you will never see the connection attempts.

This is an example of the three base rules.

```
name wan-lan {
  default-action drop
  enable-default-log
  rule 1 {
    action accept
    state {
      established enable
      related enable
    }
  }
  rule 2 {
    action drop
    log enable
    state {
      invalid enable
    }
  }
}
```

Here is an example of an IPv6 DMZ-WAN ruleset.

```
ipv6-name dmz-wan-6 {
  default-action drop
  enable-default-log
  rule 1 {
    action accept
    state {
      established enable
      related enable
    }
  }
  rule 2 {
    action drop
```

(continues on next page)

(continued from previous page)

```
    log enable
    state {
      invalid enable
  }
  rule 100 {
    action accept
    log enable
    protocol ipv6-icmp
  }
  rule 200 {
    action accept
    destination {
      port 80,443
    }
    log enable
    protocol tcp
  }
  rule 300 {
    action accept
    destination {
      port 20,21
    }
    log enable
    protocol tcp
  }
  rule 500 {
    action accept
    destination {
      port 25
    }
    log enable
    protocol tcp
    source {
      address 2001:db8:0:BBBB::200
    }
  }
  rule 600 {
    action accept
    destination {
      port 53
    }
    log enable
    protocol tcp_udp
    source {
      address 2001:db8:0:BBBB::200
    }
  }
  rule 800 {
    action accept
    destination {
    port 22
    }
    log enable
    protocol tcp
  }
}
```

Once you have all of your rulesets built, then you need to create your zone-policy.

Start by setting the interface and default action for each zone.

```
set zone-policy zone dmz default-action drop
set zone-policy zone dmz interface eth0.30
```

In this case, we are setting the v6 ruleset that represents traffic sourced from the LAN, destined for the DMZ. Because the zone-policy firewall syntax is a little awkward, I keep it straight by thinking of it backwards.

> set zone-policy zone dmz from lan firewall ipv6-name lan-dmz-6

dmz-lan policy is lan-dmz. You can get a rhythm to it when you build out a bunch at one time.

In the end, you will end up with something like this config. I took out everything but the Firewall, Interfaces, and zone-policy sections. It is long enough as is. == IPv6 Tunnel ==

If you are using a IPv6 tunnel from HE.net or someone else, the basis is the same except you have two WAN interface. One for v4 and one for v6.

You would have 5 zones instead of just 4 and you would configure your v6 ruleset between your tunnel interface and your LAN/DMZ zones instead of to the WAN.

LAN, WAN, DMZ, local and TUN (tunnel)

v6 pairs would be:

```
lan-tun
lan-local
lan-dmz
tun-lan
tun-local
tun-dmz
local-lan
local-tun
local-dmz
dmz-lan
dmz-tun
dmz-local
```

Notice, none go to WAN since WAN wouldn't have a v6 address on it.

You would have to add a couple of rules on your wan-local ruleset to allow protocol 41 in.

Something like:

```
rule 400 {
  action accept
  destination {
    address 172.16.10.1
  }
  log enable
  protocol 41
  source {
    address ip.of.tunnel.broker
  }
}
```

21.3 VyOS BGP ipv6 unnumbered with extended nexthop

General infomration can be found in the *Border Gateway Protocol (BGP)* chapter.

21.3.1 Configuration

- Router A:

```
set protocols bgp 65020 address-family ipv4-unicast redistribute connected
set protocols bgp 65020 address-family ipv6-unicast redistribute connected
set protocols bgp 65020 neighbor eth1 interface v6only
set protocols bgp 65020 neighbor eth1 interface v6only peer-group 'fabric'
set protocols bgp 65020 neighbor eth2 interface v6only
set protocols bgp 65020 neighbor eth2 interface v6only peer-group 'fabric'
set protocols bgp 65020 parameters bestpath as-path multipath-relax
set protocols bgp 65020 parameters bestpath compare-routerid
set protocols bgp 65020 parameters default no-ipv4-unicast
set protocols bgp 65020 parameters router-id '192.168.0.1'
set protocols bgp 65020 peer-group fabric address-family ipv4-unicast
set protocols bgp 65020 peer-group fabric address-family ipv6-unicast
set protocols bgp 65020 peer-group fabric capability extended-nexthop
set protocols bgp 65020 peer-group fabric remote-as 'external'
```

- Router B:

```
set protocols bgp 65021 address-family ipv4-unicast redistribute connected
set protocols bgp 65021 address-family ipv6-unicast redistribute connected
set protocols bgp 65021 neighbor eth1 interface v6only
set protocols bgp 65021 neighbor eth1 interface v6only peer-group 'fabric'
set protocols bgp 65021 neighbor eth2 interface v6only
set protocols bgp 65021 neighbor eth2 interface v6only peer-group 'fabric'
set protocols bgp 65021 parameters bestpath as-path multipath-relax
set protocols bgp 65021 parameters bestpath compare-routerid
set protocols bgp 65021 parameters default no-ipv4-unicast
set protocols bgp 65021 parameters router-id '192.168.0.2'
set protocols bgp 65021 peer-group fabric address-family ipv4-unicast
set protocols bgp 65021 peer-group fabric address-family ipv6-unicast
set protocols bgp 65021 peer-group fabric capability extended-nexthop
set protocols bgp 65021 peer-group fabric remote-as 'external'
```

21.3.2 Results

- Router A:

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             62.104.XXX.XXX/24                 u/u
eth1             -                                 u/u
eth2             -                                 u/u
lo               127.0.0.1/8                       u/u
                 192.168.0.1/32
                 ::1/128
```

```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route

S>* 0.0.0.0/0 [210/0] via 62.104.XXX.XXX, eth0, 03:21:53
C>* 62.104.56.0/24 is directly connected, eth0, 03:21:53
C>* 192.168.0.1/32 is directly connected, lo, 03:21:56
B>* 192.168.0.2/32 [20/0] via fe80::a00:27ff:fe3b:7ed2, eth2, 00:05:07
  *                       via fe80::a00:27ff:fe7b:4000, eth1, 00:05:07
```

```
vyos@vyos:~$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.575 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.628 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.581 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=0.682 ms
64 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=0.597 ms

--- 192.168.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4086ms
rtt min/avg/max/mdev = 0.575/0.612/0.682/0.047 ms
```

```
vyos@vyos:~$ show ip bgp summary

IPv4 Unicast Summary:
BGP router identifier 192.168.0.1, local AS number 65020 vrf-id 0
BGP table version 4
RIB entries 5, using 800 bytes of memory
Peers 2, using 41 KiB of memory
Peer groups 1, using 64 bytes of memory

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
eth1            4      65021      13      13        0    0    0 00:05:33            2
eth2            4      65021      13      14        0    0    0 00:05:29            2

Total number of neighbors 2
```

- Router B:

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             62.104.XXX.XXX/24                 u/u
eth1             -                                 u/u
eth2             -                                 u/u
lo               127.0.0.1/8                       u/u
                 192.168.0.2/32
                 ::1/128
```

```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
```

(continues on next page)

(continued from previous page)

```
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route

S>* 0.0.0.0/0 [210/0] via 62.104.XXX.XXX, eth0, 00:44:08
C>* 62.104.56.0/24 is directly connected, eth0, 00:44:09
B>* 192.168.0.1/32 [20/0] via fe80::a00:27ff:fe2d:205d, eth1, 00:06:18
  *                       via fe80::a00:27ff:fe93:e142, eth2, 00:06:18
C>* 192.168.0.2/32 is directly connected, lo, 00:44:11
```

```
vyos@vyos:~$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.427 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.471 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.782 ms
64 bytes from 192.168.0.1: icmp_seq=4 ttl=64 time=0.715 ms

--- 192.168.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.427/0.598/0.782/0.155 ms
```

```
vyos@vyos:~$ show ip bgp summary
IPv4 Unicast Summary:
BGP router identifier 192.168.0.2, local AS number 65021 vrf-id 0
BGP table version 4
RIB entries 5, using 800 bytes of memory
Peers 2, using 41 KiB of memory
Peer groups 1, using 64 bytes of memory

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
eth1            4      65020      14      14        0    0    0 00:06:40            2
eth2            4      65020      14      14        0    0    0 00:06:37            2

Total number of neighbors 2
```

21.4 VyOS OSPF unnumbered with ecmp

General infomration can be found in the *Open Shortest Path First (OSPF)* chapter.

21.4.1 Configuration

- Router A:

```
set interfaces ethernet eth0 address '10.0.0.1/24'
set interfaces ethernet eth1 address '192.168.0.1/32'
set interfaces ethernet eth1 ip ospf authentication md5 key-id 1 md5-key 'yourpassword
↪'
set interfaces ethernet eth1 ip ospf network 'point-to-point'
set interfaces ethernet eth2 address '192.168.0.1/32'
set interfaces ethernet eth2 ip ospf authentication md5 key-id 1 md5-key 'yourpassword
↪'
set interfaces ethernet eth2 ip ospf network 'point-to-point'
```

(continues on next page)

(continued from previous page)

```
set interfaces loopback lo address '192.168.0.1/32'
set protocols ospf area 0.0.0.0 authentication 'md5'
set protocols ospf area 0.0.0.0 network '192.168.0.1/32'
set protocols ospf parameters router-id '192.168.0.1'
set protocols ospf redistribute connected
```

- Router B:

```
set interfaces ethernet eth0 address '10.0.0.2/24'
set interfaces ethernet eth1 address '192.168.0.2/32'
set interfaces ethernet eth1 ip ospf authentication md5 key-id 1 md5-key 'yourpassword
↪'
set interfaces ethernet eth1 ip ospf network 'point-to-point'
set interfaces ethernet eth2 address '192.168.0.2/32'
set interfaces ethernet eth2 ip ospf authentication md5 key-id 1 md5-key 'yourpassword
↪'
set interfaces ethernet eth2 ip ospf network 'point-to-point'
set interfaces loopback lo address '192.168.0.2/32'
set protocols ospf area 0.0.0.0 authentication 'md5'
set protocols ospf area 0.0.0.0 network '192.168.0.2/32'
set protocols ospf parameters router-id '192.168.0.2'
set protocols ospf redistribute connected
```

21.4.2 Results

- Router A:

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             10.0.0.1/24                       u/u
eth1             192.168.0.1/32                    u/u
eth2             192.168.0.1/32                    u/u
lo               127.0.0.1/8                       u/u
                 192.168.0.1/32
                 ::1/128
vyos@vyos:~$
```

```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

S>* 0.0.0.0/0 [210/0] via 10.0.0.254, eth0, 00:57:34
O   10.0.0.0/24 [110/20] via 192.168.0.2, eth1 onlink, 00:13:21
                           via 192.168.0.2, eth2 onlink, 00:13:21
C>* 10.0.0.0/24 is directly connected, eth0, 00:57:35
O   192.168.0.1/32 [110/0] is directly connected, lo, 00:48:53
C * 192.168.0.1/32 is directly connected, eth2, 00:56:31
C * 192.168.0.1/32 is directly connected, eth1, 00:56:31
C>* 192.168.0.1/32 is directly connected, lo, 00:57:36
```

(continues on next page)

(continued from previous page)

```
O>* 192.168.0.2/32 [110/1] via 192.168.0.2, eth1 onlink, 00:29:03
  *                        via 192.168.0.2, eth2 onlink, 00:29:03
vyos@vyos:~$
```

- Router B:

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             10.0.0.2/24                       u/u
eth1             192.168.0.2/32                    u/u
eth2             192.168.0.2/32                    u/u
lo               127.0.0.1/8                       u/u
                 192.168.0.2/32
                 ::1/128
vyos@vyos:~$
```

```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

S>* 0.0.0.0/0 [210/0] via 10.0.0.254, eth0, 00:57:34
O   10.0.0.0/24 [110/20] via 192.168.0.1, eth1 onlink, 00:13:21
                          via 192.168.0.1, eth2 onlink, 00:13:21
C>* 10.0.0.0/24 is directly connected, eth0, 00:57:35
O   192.168.0.2/32 [110/0] is directly connected, lo, 00:48:53
C * 192.168.0.2/32 is directly connected, eth2, 00:56:31
C * 192.168.0.2/32 is directly connected, eth1, 00:56:31
C>* 192.168.0.2/32 is directly connected, lo, 00:57:36
O>* 192.168.0.1/32 [110/1] via 192.168.0.1, eth1 onlink, 00:29:03
  *                        via 192.168.0.1, eth2 onlink, 00:29:03
vyos@vyos:~$
```

21.5 Route-Based Site-to-Site VPN to Azure (BGP over IKEv2/IPsec)

This guide shows an example of a route-based IKEv2 site-to-site VPN to Azure using VTI and BGP for dynamic routing updates.

21.5.1 Prerequisites

- A pair of Azure VNet Gateways deployed in active-passive configuration with BGP enabled.
- A local network gateway deployed in Azure representing the Vyos device, matching the below Vyos settings except for address space, which only requires the Vyos private IP, in this example 10.10.0.5/32
- A connection resource deployed in Azure linking the Azure VNet gateway and the local network gateway representing the Vyos device.

21.5.2 Example

WAN Interface	eth0
On-premises address space	10.10.0.0/16
Azure address space	10.0.0.0/16
Vyos public IP	198.51.100.3
Vyos private IP	10.10.0.5
Azure VNet Gateway public IP	203.0.113.2
Azure VNet Gateway BGP IP	10.0.0.4
Pre-shared key	ch00s3-4-s3cur3-psk
Vyos ASN	64499
Azure ASN	65540

21.5.3 Vyos configuration

- Configure the IKE and ESP settings to match a subset of those supported by Azure:

```
set vpn ipsec esp-group AZURE compression 'disable'
set vpn ipsec esp-group AZURE lifetime '3600'
set vpn ipsec esp-group AZURE mode 'tunnel'
set vpn ipsec esp-group AZURE pfs 'dh-group2'
set vpn ipsec esp-group AZURE proposal 1 encryption 'aes256'
set vpn ipsec esp-group AZURE proposal 1 hash 'sha1'

set vpn ipsec ike-group AZURE dead-peer-detection action 'restart'
set vpn ipsec ike-group AZURE dead-peer-detection interval '15'
set vpn ipsec ike-group AZURE dead-peer-detection timeout '30'
set vpn ipsec ike-group AZURE ikev2-reauth 'yes'
set vpn ipsec ike-group AZURE key-exchange 'ikev2'
set vpn ipsec ike-group AZURE lifetime '28800'
set vpn ipsec ike-group AZURE proposal 1 dh-group '2'
set vpn ipsec ike-group AZURE proposal 1 encryption 'aes256'
set vpn ipsec ike-group AZURE proposal 1 hash 'sha1'
```

- Enable IPsec on eth0

```
set vpn ipsec ipsec-interfaces interface 'eth0'
```

- Configure a VTI with a dummy IP address

```
set interfaces vti vti1 address '10.10.1.5/32'
set interfaces vti vti1 description 'Azure Tunnel'
```

- Clamp the VTI's MSS to 1350 to avoid PMTU blackholes.

```
set firewall options interface vti1 adjust-mss 1350
```

- Configure the VPN tunnel

```
set vpn ipsec site-to-site peer 203.0.113.2 authentication id '198.51.100.3'
set vpn ipsec site-to-site peer 203.0.113.2 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 203.0.113.2 authentication pre-shared-secret 'ch00s3-
↪4-s3cur3-psk'
set vpn ipsec site-to-site peer 203.0.113.2 authentication remote-id '203.0.113.2'
```

(continues on next page)

(continued from previous page)

```
set vpn ipsec site-to-site peer 203.0.113.2 connection-type 'respond'
set vpn ipsec site-to-site peer 203.0.113.2 description 'AZURE PRIMARY TUNNEL'
set vpn ipsec site-to-site peer 203.0.113.2 ike-group 'AZURE'
set vpn ipsec site-to-site peer 203.0.113.2 ikev2-reauth 'inherit'
set vpn ipsec site-to-site peer 203.0.113.2 local-address '10.10.0.5'
set vpn ipsec site-to-site peer 203.0.113.2 vti bind 'vti1'
set vpn ipsec site-to-site peer 203.0.113.2 vti esp-group 'AZURE'
```

- **Important**: Add an interface route to reach Azure's BGP listener

```
set protocols static interface-route 10.0.0.4/32 next-hop-interface vti1
```

- Configure your BGP settings

```
set protocols bgp 64499 neighbor 10.0.0.4 remote-as '65540'
set protocols bgp 64499 neighbor 10.0.0.4 address-family ipv4-unicast soft-
↪reconfiguration 'inbound'
set protocols bgp 64499 neighbor 10.0.0.4 timers holdtime '30'
set protocols bgp 64499 neighbor 10.0.0.4 timers keepalive '10'
```

- **Important**: Disable connected check

```
set protocols bgp 64499 neighbor 10.0.0.4 disable-connected-check
```

21.6 Route-Based Redundant Site-to-Site VPN to Azure (BGP over IKEv2/IPsec)

This guide shows an example of a redundant (active-active) route-based IKEv2 site-to-site VPN to Azure using VTI and BGP for dynamic routing updates.

21.6.1 Prerequisites

- A pair of Azure VNet Gateways deployed in active-passive configuration with BGP enabled.
- A local network gateway deployed in Azure representing the Vyos device, matching the below Vyos settings except for address space, which only requires the Vyos private IP, in this example 10.10.0.5/32
- A connection resource deployed in Azure linking the Azure VNet gateway and the local network gateway representing the Vyos device.

21.6.2 Example

WAN Interface	eth0
On-premises address space	10.10.0.0/16
Azure address space	10.0.0.0/16
Vyos public IP	198.51.100.3
Vyos private IP	10.10.0.5
Azure VNet Gateway 1 public IP	203.0.113.2
Azure VNet Gateway 2 public IP	203.0.113.3
Azure VNet Gateway BGP IP	10.0.0.4,10.0.0.5
Pre-shared key	ch00s3-4-s3cur3-psk
Vyos ASN	64499
Azure ASN	65540

21.6.3 Vyos configuration

- Configure the IKE and ESP settings to match a subset of those supported by Azure:

```
set vpn ipsec esp-group AZURE compression 'disable'
set vpn ipsec esp-group AZURE lifetime '3600'
set vpn ipsec esp-group AZURE mode 'tunnel'
set vpn ipsec esp-group AZURE pfs 'dh-group2'
set vpn ipsec esp-group AZURE proposal 1 encryption 'aes256'
set vpn ipsec esp-group AZURE proposal 1 hash 'sha1'

set vpn ipsec ike-group AZURE dead-peer-detection action 'restart'
set vpn ipsec ike-group AZURE dead-peer-detection interval '15'
set vpn ipsec ike-group AZURE dead-peer-detection timeout '30'
set vpn ipsec ike-group AZURE ikev2-reauth 'yes'
set vpn ipsec ike-group AZURE key-exchange 'ikev2'
set vpn ipsec ike-group AZURE lifetime '28800'
set vpn ipsec ike-group AZURE proposal 1 dh-group '2'
set vpn ipsec ike-group AZURE proposal 1 encryption 'aes256'
set vpn ipsec ike-group AZURE proposal 1 hash 'sha1'
```

- Enable IPsec on eth0

```
set vpn ipsec ipsec-interfaces interface 'eth0'
```

- Configure two VTIs with a dummy IP address each

```
set interfaces vti vti1 address '10.10.1.5/32'
set interfaces vti vti1 description 'Azure Primary Tunnel'

set interfaces vti vti2 address '10.10.1.6/32'
set interfaces vti vti2 description 'Azure Secondary Tunnel'
```

- Clamp the VTI's MSS to 1350 to avoid PMTU blackholes.

```
set firewall options interface vti1 adjust-mss 1350
set firewall options interface vti2 adjust-mss 1350
```

- Configure the VPN tunnels

```
set vpn ipsec site-to-site peer 203.0.113.2 authentication id '198.51.100.3'
set vpn ipsec site-to-site peer 203.0.113.2 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 203.0.113.2 authentication pre-shared-secret 'ch00s3-
↪4-s3cur3-psk'
set vpn ipsec site-to-site peer 203.0.113.2 authentication remote-id '203.0.113.2'
set vpn ipsec site-to-site peer 203.0.113.2 connection-type 'respond'
set vpn ipsec site-to-site peer 203.0.113.2 description 'AZURE PRIMARY TUNNEL'
set vpn ipsec site-to-site peer 203.0.113.2 ike-group 'AZURE'
set vpn ipsec site-to-site peer 203.0.113.2 ikev2-reauth 'inherit'
set vpn ipsec site-to-site peer 203.0.113.2 local-address '10.10.0.5'
set vpn ipsec site-to-site peer 203.0.113.2 vti bind 'vti1'
set vpn ipsec site-to-site peer 203.0.113.2 vti esp-group 'AZURE'

set vpn ipsec site-to-site peer 203.0.113.3 authentication id '198.51.100.3'
set vpn ipsec site-to-site peer 203.0.113.3 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 203.0.113.3 authentication pre-shared-secret 'ch00s3-
↪4-s3cur3-psk'
set vpn ipsec site-to-site peer 203.0.113.3 authentication remote-id '203.0.113.3'
set vpn ipsec site-to-site peer 203.0.113.3 connection-type 'respond'
set vpn ipsec site-to-site peer 203.0.113.3 description 'AZURE SECONDARY TUNNEL'
set vpn ipsec site-to-site peer 203.0.113.3 ike-group 'AZURE'
set vpn ipsec site-to-site peer 203.0.113.3 ikev2-reauth 'inherit'
set vpn ipsec site-to-site peer 203.0.113.3 local-address '10.10.0.5'
set vpn ipsec site-to-site peer 203.0.113.3 vti bind 'vti2'
set vpn ipsec site-to-site peer 203.0.113.3 vti esp-group 'AZURE'
```

- **Important**: Add an interface route to reach both Azure's BGP listeners

```
set protocols static interface-route 10.0.0.4/32 next-hop-interface vti1
set protocols static interface-route 10.0.0.5/32 next-hop-interface vti2
```

- Configure your BGP settings

```
set protocols bgp 64499 neighbor 10.0.0.4 remote-as '65540'
set protocols bgp 64499 neighbor 10.0.0.4 address-family ipv4-unicast soft-
↪reconfiguration 'inbound'
set protocols bgp 64499 neighbor 10.0.0.4 timers holdtime '30'
set protocols bgp 64499 neighbor 10.0.0.4 timers keepalive '10'

set protocols bgp 64499 neighbor 10.0.0.5 remote-as '65540'
set protocols bgp 64499 neighbor 10.0.0.5 address-family ipv4-unicast soft-
↪reconfiguration 'inbound'
set protocols bgp 64499 neighbor 10.0.0.5 timers holdtime '30'
set protocols bgp 64499 neighbor 10.0.0.5 timers keepalive '10'
```

- **Important**: Disable connected check, otherwise the routes learned from Azure will not be imported into the routing table.

```
set protocols bgp 64499 neighbor 10.0.0.4 disable-connected-check
set protocols bgp 64499 neighbor 10.0.0.5 disable-connected-check
```

21.7 VyOS Tunnelbroker.net IPv6

This guides walks through the setup of Tunnelbroker.net for an IPv6 Tunnel.

21.7.1 Prerequisites

- A public IP address. This does not necessarily need to be static, but you will need to update the tunnel endpoint when/if your IP address changes, which can be done with a script and a scheduled task.
- An account at Tunnelbroker.net.
- Requested a "Regular Tunnel". You want to choose a location that is closest to your physical location for the best response time.

21.7.2 Setting up the initial tunnel

- Set up the initial IPv6 tunnel. Replace the field below from the fields on the Tunnelbroker.net tunnel information page.

```
conf
set interfaces tunnel tun0 address Client_IPv6_from_Tunnelbroker    # This will be␣
↪your VyOS install's public IPv6 address
set interfaces tunnel tun0 description 'HE.NET IPv6 Tunnel'
set interfaces tunnel tun0 encapsulation 'sit'
set interfaces tunnel tun0 local-ip Client_IPv4_from_Tunnelbroker   # This is your␣
↪public IP
set interfaces tunnel tun0 mtu '1472'
set interfaces tunnel tun0 multicast 'disable'
set interfaces tunnel tun0 remote-ip Server_IPv4_from_Tunnelbroker  # This is the IP␣
↪of the Tunnelbroker server
set protocols static interface-route6 ::/0 next-hop-interface tun0  # Tell all␣
↪traffic to go over this tunnel
commit
```

- If your WAN connection is over PPPoE, you may need to set the MTU on the above tunnel lower than 1472.
- At this point you should be able to ping an IPv6 address. Try pinging Google:

```
ping6 -c2 2001:4860:4860::8888

64 bytes from 2001:4860:4860::8888: icmp_seq=1 ttl=57 time=21.7 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=2 ttl=57 time=21.1 ms

--- 2001:4860:4860::8888 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 21.193/21.459/21.726/0.304 ms
```

- Assuming the pings are successful, you need to add some DNS servers. Some options:

```
set system name-server 2001:4860:4860::8888  # Google
set system name-server 2001:4860:4860::8844  # Google
set system name-server 2606:4700:4700::1111  # Cloudflare
set system name-server 2606:4700:4700::1001  # Cloudflare
commit
```

- You should now be able to ping something by IPv6 DNS name:

```
# ping6 -c2 one.one.one.one
PING one.one.one.one(one.one.one.one) 56 data bytes
64 bytes from one.one.one.one: icmp_seq=1 ttl=58 time=16.8 ms
64 bytes from one.one.one.one: icmp_seq=2 ttl=58 time=17.4 ms
```

(continues on next page)

(continued from previous page)

```
--- one.one.one.one ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 16.880/17.153/17.426/0.273 ms
```

- Assuming everything works, you can proceed to client configuration

21.7.3 LAN Configuration

At this point your VyOS install should have full IPv6, but now your LAN devices need access.

With Tunnelbroker.net, you have two options:

- Routed /64. This is the default assignment. In IPv6-land, it's good for a single "LAN", and is somewhat equivalent to a /24. Example: *2001:470:xxxx:xxxx::/64*
- Routed /48. This is something you can request by clicking the "Assign /48" link in the Tunnelbroker.net tunnel config. It allows you to have up to 65k LANs. Example: *2001:470:xxxx::/48*

Unlike IPv4, IPv6 is really not designed to be broken up smaller than /64. So if you ever want to have multiple LANs, VLANs, DMZ, etc, you'll want to ignore the assigned /64, and request the /48 and use that.

21.7.4 Single LAN Setup

Single LAN setup where eth1 is your LAN interface. Use the /64 (all the xxxx should be replaced with the information from your *Routed /64* tunnel):

```
set interfaces ethernet eth1 address '2001:470:xxxx:xxxx::1/64'
set interfaces ethernet eth1 ipv6 router-advert name-server '2001:4860:4860::8888'
set interfaces ethernet eth1 ipv6 router-advert name-server '2001:4860:4860::8844'
set interfaces ethernet eth1 ipv6 router-advert prefix 2001:470:xxxx:xxxx::/64␣
↪autonomous-flag 'true'
set interfaces ethernet eth1 ipv6 router-advert prefix 2001:470:xxxx:xxxx::/64 on-
↪link-flag 'true'
set interfaces ethernet eth1 ipv6 router-advert prefix 2001:470:xxxx:xxxx::/64 valid-
↪lifetime '2592000'
```

- This accomplishes a few things:
  - Sets your LAN interface's IP address
  - Enables router advertisements. This is an IPv6 alternative for DHCP (though DHCPv6 can still be used). With RAs, Your devices will automatically find the information they need for routing and DNS.

21.7.5 Multiple LAN/DMZ Setup

In this, you use the *Routed /48* information. This allows you to assign a different /64 to every interface, LAN, or even device. Or you could break your network into smaller chunks like /56 or /60.

The format of these addresses:

- *2001:470:xxxx::/48*: The whole subnet. xxxx should come from Tunnelbroker.
- *2001:470:xxxx:1::/64*: A subnet suitable for a LAN
- *2001:470:xxxx:2::/64*: Another subnet

- *2001:470:xxxx:ffff:/64*: The last usable /64 subnet.

In the above examples, 1,2,ffff are all chosen by you. You can use 1-ffff (1-65535).

So, when your LAN is eth1, your DMZ is eth2, your cameras live on eth3, etc:

```
set interfaces ethernet eth1 address '2001:470:xxxx:1::1/64'
set interfaces ethernet eth1 ipv6 router-advert name-server '2001:4860:4860::8888'
set interfaces ethernet eth1 ipv6 router-advert name-server '2001:4860:4860::8844'
set interfaces ethernet eth1 ipv6 router-advert prefix 2001:470:xxxx:1::/64␣
↪autonomous-flag 'true'
set interfaces ethernet eth1 ipv6 router-advert prefix 2001:470:xxxx:1::/64 on-link-
↪flag 'true'
set interfaces ethernet eth1 ipv6 router-advert prefix 2001:470:xxxx:1::/64 valid-
↪lifetime '2592000'

set interfaces ethernet eth2 address '2001:470:xxxx:2::1/64'
set interfaces ethernet eth2 ipv6 router-advert name-server '2001:4860:4860::8888'
set interfaces ethernet eth2 ipv6 router-advert name-server '2001:4860:4860::8844'
set interfaces ethernet eth2 ipv6 router-advert prefix 2001:470:xxxx:2::/64␣
↪autonomous-flag 'true'
set interfaces ethernet eth2 ipv6 router-advert prefix 2001:470:xxxx:2::/64 on-link-
↪flag 'true'
set interfaces ethernet eth2 ipv6 router-advert prefix 2001:470:xxxx:2::/64 valid-
↪lifetime '2592000'

set interfaces ethernet eth3 address '2001:470:xxxx:3::1/64'
set interfaces ethernet eth3 ipv6 router-advert name-server '2001:4860:4860::8888'
set interfaces ethernet eth3 ipv6 router-advert name-server '2001:4860:4860::8844'
set interfaces ethernet eth3 ipv6 router-advert prefix 2001:470:xxxx:3::/64␣
↪autonomous-flag 'true'
set interfaces ethernet eth3 ipv6 router-advert prefix 2001:470:xxxx:3::/64 on-link-
↪flag 'true'
set interfaces ethernet eth3 ipv6 router-advert prefix 2001:470:xxxx:3::/64 valid-
↪lifetime '2592000'
```

21.7.6 Firewall

Finally, don't forget the *Firewall*. The usage is identical, except for instead of *set firewall name NAME*, you would use *set firewall ipv6-name NAME*.

Similarly, to attach the firewall, you would use *set interfaces ethernet eth0 firewall in ipv6-name* or *set zone-policy zone LOCAL from WAN firewall ipv6-name*

CHAPTER 22

Command tree

See the the full Command tree in Operational mode and Configuration mode

22.1 Operational mode

Operational mode allows for commands to perform operational system tasks and view system and service status. After this is the first view after the login. Please see *Command-Line Interface* for navigation in the CLI

```
vyos@vyos:~$ [tab]
Possible completions:
  add               Add an object to a service
  clear             Clear system information
  clone             Clone an object
  configure         Enter configure mode
  connect           Establish a connection
  copy              Copy an object
  delete            Delete an object
  disconnect        Take down a connection
  force             Force an operation
  format            Format a device
  generate          Generate an object
  install           Install a new system
  monitor           Monitor system information
  ping              Send IPv4 or IPv6 ICMP (Internet Control Message Protocol) echo␣
↪requests
  poweroff          Poweroff the system
  reboot            Reboot the system
  release           Release specified variable
  rename            Rename an object
  renew             Renew specified variable
  reset             Reset a service
  restart           Restart a service
  set               Set operational options
```

(continues on next page)

(continued from previous page)

```
  show              Show system information
  telnet            Telnet to a node
  traceroute        Track network path to node
  update            Update data for a service
```

22.1.1 Add

```
raid         Add a RAID set element
system       Add an item to a system facility
```

22.1.2 Clear

```
console            Clear screen
firewall           Clear firewall statistics
flow-accounting    Clear flow accounting
interfaces         Clear interface information
ip                 Clear Internet Protocol (IP) statistics or status
ipv6               Clear Internet Protocol (IPv6) statistics or status
nat                Clear network address translation (NAT) tables
policy             Clear policy statistics
```

22.1.3 Clone

The `clone` command allows you to clone a configuration from a system image to another one, or from the running config to another system image. To clone the running config to a system image:

```
clone system config <system-image> from running
```

To clone from system image A to system image B:

```
clone system config <system-image-B> from <system-image-A>
```

22.1.4 Configure

The `configure` command allows you to enter configuration mode.

```
vyos@vyos:~$ configure
[edit]
vyos@vyos#
```

22.1.5 Connect

The `connect` command allows you to bring up a connection oriented interface, like a pppoe interface.

```
connect interface <interface>
```

22.1.6 Copy

The `copy` command allows you to copy a file to your running config or over images.

It can look like this example:

```
vyos@vyos:~$  copy file [tab]
Possible completions:
  http://<user>:<passwd>@<host>/<file>
                Copy files from specified source
  scp://<user>:<passwd>@<host>/<file>
  ftp://<user>:<passwd>@<host>/<file>
  tftp://<host>/<file>
  1.2.0://config/
  1.2.0-rolling+201902251818://config/
  1.2.0-rolling+201902201040://config/
  1.2.0-rolling+201902080337://config/
  1.2.0-H4://config/
  running://config/
```

To copy from file A to file B:

```
copy <file A> to <file B>
```

22.1.7 Delete

```
conntrack     Delete Conntrack entries
file          Delete files in a particular image
log           Delete a log file
raid          Remove a RAID set element
system        Delete system objects
```

22.1.8 Disconnect

The `disconnect` command allows you to take down a connection oriented interface, like a pppoe interface.

```
disconnect interface <interface>
```

22.1.9 Force

```
arp           Send gratuitous ARP request or reply
cluster       Force a cluster state transition
```

22.1.10 Format

The `format` command allows you to format a disk the same way as another one.

```
format disk <target> like <source>
```

22.1.11 Generate

```
openvpn       OpenVPN key generation tool
ssh-server-key
              Regenerate the host SSH keys and restart the SSH server
tech-support  Generate tech-support archive
vpn           VPN key generation utility
wireguard     wireguard key generation utility
```

22.1.12 Install

The `install` command allows you to install the system image on the disk.

```
install image
```

22.1.13 Monitor

`monitor` can be used to continually view what is happening on the router.

```
bandwidth     Monitor interface bandwidth in real time
bandwidth-test
              Initiate or wait for bandwidth test
cluster       Monitor clustering service
command       Monitor an operational mode command (refreshes every 2 seconds)
conntrack-sync
              Monitor conntrack-sync
content-inspection
              Monitor Content-Inspection
dhcp          Monitor Dynamic Host Control Protocol (DHCP)
dns           Monitor a Domain Name Service (DNS) daemon
firewall      Monitor Firewall
https         Monitor the Secure Hypertext Transfer Protocol (HTTPS) service
lldp          Monitor Link Layer Discovery Protocol (LLDP) daemon
log           Monitor last lines of messages file
nat           Monitor network address translation (NAT)
openvpn       Monitor OpenVPN
protocol      Monitor routing protocols
snmp          Monitor Simple Network Management Protocol (SNMP) daemon
stop-all      Stop all current background monitoring processes
traceroute    Monitor the path to a destination in realtime
traffic       Monitor traffic dumps
vpn           Monitor VPN
vrrp          Monitor Virtual Router Redundancy Protocol (VRRP)
webproxy      Monitor Webproxy service
```

22.1.14 Ping

The `ping` command allows you to send an ICMP-EchoRequest packet and display the ICMP-EchoReply received.

```
<hostname>    Send Internet Control Message Protocol (ICMP) echo request
<x.x.x.x>
<h:h:h:h:h:h:h:h>
```

22.1.15 Poweroff

The `poweroff` command allows you to properly shut down the VyOS instance. Without any modifier, the command is executed immediately.

```
<Enter>       Execute the current command
at            Poweroff at a specific time
cancel        Cancel a pending poweroff
in            Poweroff in X minutes
now           Poweroff the system without confirmation
```

22.1.16 Reboot

The `reboot` command allows you to properly restart the VyOS instance. Without any modifier, the command is executed immediately.

```
<Enter>       Execute the current command
at            Poweroff at a specific time
cancel        Cancel a pending poweroff
in            Poweroff in X minutes
now           Poweroff the system without confirmation
```

22.1.17 Release

The `release` command allows you to release a DHCP or DHCPv6 lease.

```
vyos@vyos:~$ release dhcp interface <int>
vyos@vyos:~$ release dhcpv6 interface <int>
```

22.1.18 Rename

The `rename` command allows you to rename a system image.

```
rename system image <currentname> <newname>
```

22.1.19 Renew

The `renew` command allows you to renew a DHCP or DHCPv6 lease.

```
vyos@vyos:~$ renew dhcp interface <int>
vyos@vyos:~$ renew dhcpv6 interface <int>
```

22.1.20 Reset

```
conntrack     Reset all currently tracked connections
conntrack-sync
              Reset connection syncing parameters
dns           Reset a DNS service state
firewall      reset a firewall group
```

(continues on next page)

(continued from previous page)

```
ip            Reset Internet Protocol (IP) parameters
ipv6          Reset Internet Protocol version 6 (IPv6) parameters
nhrp          Clear/Purge NHRP entries
openvpn       Reset OpenVPN
terminal      Reset terminal
vpn           Reset Virtual Private Network (VPN) information
```

22.1.21 Restart

```
cluster       Restart cluster node
conntrack-sync
              Restart connection tracking synchronization service
dhcp          Restart DHCP processes
dhcpv6        Restart DHCPv6 processes
dns           Restart a DNS service
flow-accounting
              Restart flow-accounting service
https         Restart https server
vpn           Restart IPsec VPN
vrrp          Restart the VRRP (Virtual Router Redundancy Protocol) process
wan-load-balance
              Restart WAN load balancing
webproxy      Restart webproxy service
```

22.1.22 Set

```
<OPTION>      Bash builtin set command
console       Control console behaviors
date          Set system date and time
system        Set system operational parameters
terminal      Control terminal behaviors
```

22.1.23 Show

```
arp           Show Address Resolution Protocol (ARP) information
bridge        Show bridging information
cluster       Show clustering information
configuration Show available saved configurations
conntrack     Show conntrack entries in the conntrack table
conntrack-sync
              Show connection syncing information
date          Show system time and date
dhcp          Show DHCP (Dynamic Host Configuration Protocol) information
dhcpv6        Show DHCPv6 (IPv6 Dynamic Host Configuration Protocol) information
disk          Show status of disk device
dns           Show DNS information
file          Show files for a particular image
firewall      Show firewall information
flow-accounting
              Show flow accounting statistics
```

(continues on next page)

(continued from previous page)

```
hardware      Show system hardware details
history       show command history
host          Show host information
incoming      Show ethernet input-policy information
interfaces    Show network interface information
ip            Show IPv4 routing information
ipv6          Show IPv6 routing information
license       Show VyOS license information
lldp          Show lldp
log           Show contents of current master log file
login         Show current login credentials
monitoring    Show currently monitored services
nat           Show Network Address Translation (NAT) information
nhrp          Show NHRP info
ntp           Show peer status of NTP daemon
openvpn       Show OpenVPN information
policy        Show policy information
poweroff      Show scheduled poweroff
pppoe-server  show pppoe-server status
queueing      Show ethernet queueing information
raid          Show statis of RAID set
reboot        Show scheduled reboot
remote-config Show remote side config
route-map     Show route-map information
snmp          Show status of SNMP on localhost
system        Show system information
system-integrity
              checks the integrity of the system
table         Show routing table
tech-support  Show consolidated tech-support report (private information removed)
users         Show user information
version       Show system version information
vpn           Show Virtual Private Network (VPN) information
vrrp          Show VRRP (Virtual Router Redundancy Protocol) information
wan-load-balance
              Show Wide Area Network (WAN) load-balancing information
webproxy      Show webproxy information
wireguard     Show wireguard properties
zone-policy   Show summary of zone policy for a specific zone
```

22.1.24 Telnet

In the past the `telnet` command allowed you to connect remotely to another device using the telnet protocol. Telnet is unencrypted and should not use anymore. But its nice to test if an TCP Port to a host is open.

```
vyos@vyos:~$ telnet 192.168.1.3 443
Trying 192.168.1.3...
telnet: Unable to connect to remote host: Network is unreachable

vyos@vyos:~$ telnet 192.168.1.4 443
Trying 192.168.1.4...
Connected to 192.168.1.4.
Escape character is '^]'.
```

22.1.25 Traceroute

The `traceroute` command allows you to trace the path taken to a particular device.

```
<hostname>    Track network path to specified node
<x.x.x.x>
<h:h:h:h:h:h:h:h>
ipv4          Track network path to <hostname|IPv4 address>
ipv6          Track network path to <hostname|IPv6 address>
```

22.1.26 Update

```
dns           Update DNS information
webproxy      Update webproxy
```

22.2 Configuration mode

```
confirm           Confirm prior commit-confirm
comment           Add comment to this configuration element
commit            Commit the current set of changes
commit-confirm    Commit the current set of changes with 'confirm' required
compare           Compare configuration revisions
copy              Copy a configuration element
delete            Delete a configuration element
discard           Discard uncommitted changes
edit              Edit a sub-element
exit              Exit from this configuration level
load              Load configuration from a file and replace running configuration
loadkey           Load user SSH key from a file
merge             Load configuration from a file and merge running configuration
rename            Rename a configuration element
rollback          Rollback to a prior config revision (requires reboot)
run               Run an operational-mode command
save              Save configuration to a file
set               Set the value of a parameter or create a new element
show              Show the configuration (default values may be suppressed)
```

22.2.1 Confirm

The `confirm` command confirms the prior `commit-confirm`.

22.2.2 Comment

The `comment` commands allow you to insert a comment above the current configuration section. The command cannot be used at the top of the configuration hierarchy, only on subsections. Comments needs to be commited, just like other config changes.

To add a comment to a section, while being already at the proper section level:

```
[edit <section>]
vyos@vyos# comment "Type Comment Here"
```

To add a comment directly to a section, from the top or a higher section:

```
[edit]
vyos@vyos# comment <section> "Type Comment Here"
```

To remove a comment, add a blank comment to overwrite:

```
[edit <section>]
vyos@vyos# comment ""
```

Examples

To add a comment to the "interfaces" section:

```
[edit]
vyos@vyos# edit interfaces
[edit interfaces]
vyos@vyos# comment "Here is a comment"
[edit interfaces]
vyos@vyos# commit
```

The comment would then appear like this:

```
[edit]
vyos@vyos# show
 /* Here is a comment */
  interfaces {
      ethernet eth0 {
  [...]
```

An important thing to note is that since the comment is added on top of the section, it will not appear if the `show <section>` command is used. With the above example, the `show interfaces` command would return starting after the "interfaces {" line, hiding the comment:

```
[edit]
vyos@vyos# show interfaces
 ethernet eth0 {
[...]
```

To add a comment to the interfaces section from the top:

```
[edit]
vyos@vyos# comment interfaces "test"
```

The comment can be added to any node that already exists, even if it's multiple levels lower:

```
[edit]
vyos@vyos# comment interfaces ethernet eth0 vif 222 address "Far down comment"
```

22.2.3 Commit

The `commit` command commits the proposed changes to the configuration file. Every changes done in the configuration session is only applied when the configuration is committed. To view the changes that will be applied, use the show command. To discard the changes without committing, use the `discard` command. The `commit` command doesn't save the configuration, you need to manually use the `save` command.

The confirm keyword can be added, see `commit-confirm`. A comment can be entered, it will appear in the commit log.

```
[edit]
vyos@vyos# commit
Possible completions:
  <Enter>       Commit working configuration
  comment       Comment for commit log
```

22.2.4 Commit-confirm

The `commit-confirm` command commits the proposed changes to the configuration file and starts a timer. If the `confirm` command is not entered before the timer expiration, the configuration will be rolled back and VyOS will reboot. The default timer value is 10 minutes, but a custom value can be entered.

```
[edit]
vyos@vyos# commit-confirm
Possible completions:
  <Enter>       Commit, rollback/reboot in 10 minutes if no confirm
  <N>           Commit, rollback/reboot in N minutes if no confirm
  comment       Comment for commit log
```

22.2.5 Compare

VyOS maintains backups of previous configurations. To compare configuration revisions in configuration mode, use the compare command:

```
[edit]
vyos@vyos# compare
Possible completions:
  <Enter>     Compare working & active configurations
  saved               Compare working & saved configurations
  <N>         Compare working with revision N
  <N> <M>     Compare revision N with M

  Revisions:
    0 2019-03-20 20:57:22 root by boot-config-loader
    1 2019-03-15 20:00:04 root by boot-config-loader
    2 2019-03-05 01:58:39 vyos by cli
    3 2019-03-05 01:54:59 vyos by cli
    4 2019-03-05 01:53:08 vyos by cli
    5 2019-03-05 01:52:21 vyos by cli
    6 2019-02-24 21:01:24 root by boot-config-loader
    7 2019-02-21 22:00:12 vyos by cli
    8 2019-02-21 21:56:49 vyos by cli
```

22.2.6 Copy

The `copy` command allows you to copy a configuration object.

Copy the configuration entrys from a firewall name WAN rule 1 to rule 2.

```
[edit firewall name WAN]
vyos@vyos# show
 rule 1 {
     action accept
     source {
         address 10.1.0.0/24
     }
 }
[edit firewall name WAN]
vyos@vyos# copy rule 1 to rule 2
[edit firewall name WAN]
vyos@vyos# show
 rule 1 {
     action accept
     source {
         address 10.1.0.0/24
     }
 }
+rule 2 {
+    action accept
+    source {
+        address 10.1.0.0/24
+    }
+}
```

22.2.7 Delete

The `delte` command is to delete a configuration entry.

This Example delete the hole `service tftp-server` section.

```
delete service tftp-server
```

22.2.8 Discard

The `discard` command removes all pending configuration changes.

```
[edit]
vyos@vyos# discard

  Changes have been discarded
```

22.2.9 Edit

The `edit` command allows you to navigate down into the configuration tree. To get back to an upper level, use the `up` command or use the `top` command to get back to the upper most level. The `[edit]` text displays where the user is located in the configuration tree.

```
[edit]
vyos@vyos# edit interfaces
[edit interfaces]
vyos@vyos# edit ethernet eth0
[edit interfaces ethernet eth0]
```

22.2.10 Exit

The `exit` command exits the current configuration mode. If the current configuration level isn't the top-most, then the configuration level is put back to the top-most level. If the configuration level is at the top-most level, then it exits the configuration mode and returns to operational mode. The `exit` command cannot be used if uncommitted changes exists in the configuration file. To exit with uncommitted changes, you either need to use the `exit discard` command or you need to commit the changes before exiting. The `exit` command doesn't save the configuration, only the `save` command does. A warning will be given when exiting with unsaved changes. Using the `exit` command in operational mode will logout the session.

Exiting from a configuration level:

```
[edit interfaces ethernet eth0]
vyos@vyos# exit
[edit]
vyos@vyos#
```

Exiting from configuration mode:

```
[edit]
vyos@vyos# exit
exit
vyos@vyos:~$
```

Exiting from operational mode:

```
vyos@vyos:~$ exit
logout
```

Error message when trying to exit with uncommitted changes:

```
vyos@vyos# exit
Cannot exit: configuration modified.
Use 'exit discard' to discard the changes and exit.
[edit]
vyos@vyos#
```

Warning message when exiting with unsaved changes:

```
[edit]
vyos@vyos# exit
Warning: configuration changes have not been saved.
exit
vyos@vyos:~$
```

22.2.11 Load

The `load` command load a configuration from a local or remote file. You have to be use `commit` to make the change active

```
<Enter>                                Load from system config file
<file>                                 Load from file on local machine
scp://<user>:<passwd>@<host>/<file>    Load from file on remote machine
sftp://<user>:<passwd>@<host>/<file>   Load from file on remote machine
ftp://<user>:<passwd>@<host>/<file>    Load from file on remote machine
http://<host>/<file>                   Load from file on remote machine
```

(continues on next page)

(continued from previous page)

```
https://<host>/<file>                 Load from file on remote machine
tftp://<host>/<file>                  Load from file on remote machine
```

```
[edit]
vyos@vyos# load
Loading configuration from '/config/config.boot'...

Load complete.  Use 'commit' to make changes active.
```

22.2.12 Loadkey

Copies the content of a public key to the ~/.ssh/authorized_keys file.

```
loadkey <username> [tab]

<file>                      Load from file on local machine
scp://<user>@<host>/<file>  Load from file on remote machine
sftp://<user>@<host>/<file> Load from file on remote machine
ftp://<user>@<host>/<file>  Load from file on remote machine
http://<host>/<file>        Load from file on remote machine
tftp://<host>/<file>        Load from file on remote machine
```

22.2.13 Merge

The `merge` command merge the config from a local or remote file with the running config.

In the example below exist a `default-firewall.config` file with some common firewall rules you saved earlier.

```
[edit]
vyos@vyos# show firewall
Configuration under specified path is empty
[edit]
vyos@vyos# merge default-firewall.config
Loading configuration from '/config/default-firewall.config'...

Merge complete.  Use 'commit' to make changes active.
[edit]
vyos@vyos#

vyos@vyos# show firewall
+all-ping enable
+broadcast-ping disable
+config-trap disable
+ipv6-receive-redirects disable
+ipv6-src-route disable
+ip-src-route disable
+log-martians enable
+name WAN {
+    default-action drop
+    rule 1 {
+        action accept
+        source {
+            address 10.1.0.0/24
```

(continues on next page)

(continued from previous page)

```
+        }
+    }
+    rule 2 {
+        action accept
+        source {
+            address 10.1.0.0/24
+        }
......
```

22.2.14 Rename

The `rename` command allows you to rename or move a configuration object.

See here how to move the configuration entrys from vlanid 3 to 2

```
[edit interfaces ethernet eth1]
vyos@vyos# show
 duplex auto
 hw-id 08:00:27:81:c6:59
 smp-affinity auto
 speed auto
 vif 3 {
     address 10.4.4.4/32
 }
[edit interfaces ethernet eth1]
vyos@vyos# rename vif 3 to vif 2
[edit interfaces ethernet eth1]
vyos@vyos# show
 duplex auto
 hw-id 08:00:27:81:c6:59
 smp-affinity auto
 speed auto
+vif 2 {
+    address 10.4.4.4/32
+}
-vif 3 {
-    address 10.4.4.4/32
-}
[edit interfaces ethernet eth1]
vyos@vyos#
```

22.2.15 Rollback

You can `rollback` configuration using the rollback command, however this command will currently trigger a system reboot. Use the compare command to verify the configuration you want to rollback to.

```
vyos@vyos# compare 1
[edit system]
>host-name vyos-1
[edit]
vyos@vyos# rollback 1
Proceed with reboot? [confirm][y]
```

(continues on next page)

(continued from previous page)

```
Broadcast message from root@vyos-1 (pts/0) (Tue Dec 17 21:07:45 2018):

The system is going down for reboot NOW!
[edit]
vyos@vyos#
```

22.2.16 Run

The `run` command allows you to execute any operational mode commands without exiting the configuration session.

```
[edit]
vyos@vyos# run show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             10.1.1.1/24                        u/u
```

22.2.17 Save

The `save` command saves the current configuration to non-volatile storage. VyOS also supports saving and loading configuration remotely using SCP, FTP, or TFTP.

```
<Enter>                                Save to system config file
<file>                                 Save to file on local machine
scp://<user>:<passwd>@<host>/<file>    Save to file on remote machine
sftp://<user>:<passwd>@<host>/<file>   Save to file on remote machine
ftp://<user>:<passwd>@<host>/<file>    Save to file on remote machine
tftp://<host>/<file>                   Save to file on remote machine
```

22.2.18 Set

The `set` command create all configuration entrys

```
[edit]
vyos@vyos# set protocols static route 0.0.0.0/0 next-hop 192.168.1.1
```

22.2.19 Show

The `show` command in the configuration mode displays the configuration and show uncommitted changes.

Show the hole config, the address and description of eth1 is moving to vlan 2 if you commit the changes.

```
[edit]
vyos@vyos# show
 interfaces {
     dummy dum0 {
         address 10.3.3.3/24
     }
     ethernet eth0 {
         address dhcp
```

(continues on next page)

(continued from previous page)

```
         duplex auto
         hw-id 08:00:27:2b:c0:0b
         smp-affinity auto
         speed auto
     }
     ethernet eth1 {
-        address 10.1.1.1/32
-        description "MGMT Interface"
         duplex auto
         hw-id 08:00:27:81:c6:59
         smp-affinity auto
         speed auto
+        vif 2 {
+            address 10.1.1.1/32
+            description "MGMT Interface"
+        }
     }
     loopback lo {
     }
 }
 service {
     ssh {
         port 22
......
```

CHAPTER 23

Running on Bare Metal

23.1 Intel Atom C3000

I opted to get one of the new Intel Atom C3000 CPUs to spawn VyOS on it. Running VyOS on an UEFI only device is supported as of VyOS release 1.2.

23.1.1 Shopping Cart

- 1x Supermicro CSE-505-203B (19" 1U chassis, inkl. 200W PSU)
- 1x Supermicro MCP-260-00085-0B (I/O Shield for A2SDi-2C-HLN4F)
- 1x Supermicro A2SDi-2C-HLN4F (Intel Atom C3338, 2C/2T, 4MB cache, Quad LAN with Intel C3000 SoC 1GbE)
- 1x Crucial CT4G4DFS824A (4GB DDR4 RAM 2400 MT/s, PC4-19200)
- 1x SanDisk Ultra Fit 32GB (USB-A 3.0 SDCZ43-032G-G46 mass storage for OS)
- 1x Supermicro MCP-320-81302-0B (optional FAN tray)

23.1.2 Optional (10GE)

If you wan't to get additional ethernet ports or even 10GE connectivity the following optional parts will be required:

- 1x Supermicro RSC-RR1U-E8 (Riser Card)
- 1x Supermicro MCP-120-00063-0N (Riser Card Bracket)

Latest VyOS rolling releases boot without any problem on this board. You also receive a nice IPMI interface realized with an ASPEED AST2400 BMC (no information about [OpenBMC](https://www.openbmc.org/)) so far on this motherboard.

23.1.3 Pictures

23.2 PC Engines APU4

As this platform seems to be quiet common in terms of noise, cost, power and performance it makes sense to write a small installation manual.

This guide was developed using an APU4C4 board with the following specs:

- AMD Embedded G series GX-412TC, 1 GHz quad Jaguar core with 64 bit and AES-NI support, 32K data + 32K instruction cache per core, shared 2MB L2 cache.
- 4 GB DDR3-1333 DRAM, with optional ECC support
- About 6 to 10W of 12V DC power depending on CPU load
- 2 miniPCI express (one with SIM socket for 3G modem).
- 4 Gigabit Ethernet channels using Intel i211AT NICs

The board can be powered via 12V from the front or via a 5V onboard connector.

23.2.1 Shopping Cart

- 1x apu4c4 = 4 i211AT LAN / AMD GX-412TC CPU / 4 GB DRAM / dual SIM
- 1x Kingston SUV500MS/120G

CABLE
3.5" HDD/ 2.5" HDD
3.5" HDD/ 2.5" HDD
CABLE
3.5" HDD
CABLE
2X2.5" HDD

CABLE
2X2.5" HDD
CABLE
3.5" HDD
3.5" HDD/ 2.5" HDD
3.5" HDD/ 2.5" HDD
CABLE
2X2.5" HDD
CABLE
3.5" HDD
SUPERMICRO
INPUT RATING: 100~240V~, 50/60Hz, 2.5A Max
80 PLUS
OK

CABLE
3.5" HDD
CABLE
3.5" HDD
ASPEED
AST2400
AMI Aptio
SUPERMICRO
E139336
IS 13252(Part 1)/
IEC 60950-1
R-41067938
OUTPUT
Total output power 200W max
+3.3V & +5V combined power 20W max
+3.3V & +5V combined power 20W max
Adjustment
Made in China
REV: 1.1
80 PLUS GOLD
HI-POT
OK

SUPERMICRO
E139336
IS 13252(Part 1)/
IEC 60950-1
R-41067938
RATINGS
100~240V~ 50/60Hz
2.6A Max
Adjustment
Made in China
REV: 1.1
80 PLUS GOLD
HI-POT
OK
3.5" HDD/ 2.5" HDD
2X2.5" HDD

SUPERMICRO
80 PLUS GOLD
OK
HDD 2.5"HDD

CABLE
2X2.5" HDD
3.5" HDD/ 2.5" HDD

- 1x VARIA Group Item 326745 10" dual rack rack for APU4

The 19" enclosure can accomodate two APU4 boards - there is a single and dual cover.

23.2.2 VyOS 1.2 (crux)

Depending on the VyOS versions you intend to install there is a difference in the serial port settings (T1327).

Create a bootable USB pendrive using e.g. Rufus on a Windows machine.

Connect serial port to a PC through null modem cable (RXD / TXD crossed over). Set terminal emulator to 115200 8N1.

```
PC Engines apu4
coreboot build 20171130
BIOS version v4.6.4
4080 MB ECC DRAM
SeaBIOS (version rel-1.11.0.1-0-g90da88d)

Press F10 key now for boot menu:

Select boot device:

1. ata0-0: KINGSTON SUV500MS120G ATA-11 Hard-Disk (111 GiBytes)
2. USB MSC Drive Generic Flash Disk 8.07
3. Payload [memtest]
4. Payload [setup]
```

Now boot from the `USB MSC Drive Generic Flash Disk 8.07` media by pressing `2`, the VyOS boot menu will appear, just wait 10 seconds or press `Enter` to continue.

```
lqqq
```

```
set system console device ttyS0 speed 115200
```

Note: Once you `commit` the above changes access to the serial interface is lost until you set your terminal emulator to 115200 8N1 again.

```
vyos@vyos# show system console
 device ttyS0 {
   speed 115200
 }
```

23.2.3 VyOS 1.2 (rolling)

Installing the rolling release on an APU2 board does not require any change on the serial console from your host side as T1327 was successfully implemented.

Simply proceed with a regular image installation as described in *Installation*.

23.2.4 Pictures

Note: Both device types operate without any moving parts and emit zero noise.

Rack Mount

Kingston
120GB
ETH0
ETH1
ETH2
ETH3
10/100/1000

SATA
48RSRD02.0GAAPC
SD
I2C
SD CARD
RTC clr
SPI
USB
LPC
mSATA / mPCIe 3
mPCIe 2
mPCIe 1
Kingston
120GB
2R2
1847
4R7
1843

SD CARD
mSATA / mPCI 3
mPCIe 2
mPCIe 1
USB
LPC

Kingston
120GB

Desktop

23.3 Qotom Q355G4

The install on this Q355G4 box is pretty much plug and play. The port numbering the OS does might differ from the labels on the outside, but the UEFI firmware has a port blink test built in with MAC adresses so you can very quickly identify which is which. MAC labels are on the inside as well, and this test can be done from VyOS or plain Linux too. Default settings in the UEFI will make it boot, but depending on your installation wishes (i.e. storage type, boot type, console type) you might want to adjust them. This Qotom company seems to be the real OEM/ODM for many other relabelling companies like Protectli.

23.3.1 Hardware

There are a number of other options, but they all seem to be close to Intel reference designs, with added features like more serial ports, more network interfaces and the likes. Because they don't deviate too much from standard designs all the hardware is well-supported by mainline. It accepts one LPDDR3 SO-DIMM, but chances are that if you need more than that, you'll also want something even beefier than an i5. There are options for antenna holes, and SIM slots, so you could in theory add an LTE/Cell modem (not tested so far).

The chassis is a U-shaped alu extrusion with removable I/O plates and removable bottom plate. Cooling is completely passive with a heatsink on the SoC with internal and external fins, a flat interface surface, thermal pad on top of that, which then directly attaches to the chassis, which has fins as well. It comes with mounting hardware and rubber feet, so you could place it like a desktop model or mount it on a VESA mount, or even wall mount it with the provided mounting plate. The closing plate doubles as internal 2.5" mounting place for an HDD or SSD, and comes supplied with a small SATA cable and SATA power cable.

Power supply is a 12VDC barrel jack, and included switching power supply, which is why SATA power regulation is on-board. Internally it has a NUC-board-style on-board 12V input header as well, the molex locking style.

There are WDT options and auto-boot on power enable, which is great for remote setups. Firmware is reasonably secure (no backdoors found, BootGuard is enabled in enforcement mode, which is good but also means no coreboot option), yet has most options available to configure (so it's not locked out like most firmwares are).

An external RS232 serial port is available, internally a GPIO header as well. It does have Realtek based audio on board for some reason, but you can disable that. Booting works on both USB2 and USB3 ports. Switching between serial BIOS mode and HDMI BIOS mode depends on what is connected at startup; it goes into serial mode if you disconnect HDMI and plug in serial, in all other cases it's HDMI mode.

23.4 Partaker i5

I believe this is actually the same hardware as the Protectli. I purchased it from Amazon in June 2018. It came pre-loaded with pfSense.

Manufacturer product page.

23.4.1 Installation

- Write VyOS ISO to USB drive of some sort
- Plug in VGA, power, USB keyboard, and USB drive
- Press "SW" button on the front (this is the power button; I don't know what "SW" is supposed to mean).

- Begin rapidly pressing delete on the keyboard. The boot prompt is very quick, but with a few tries you should be able to get into the BIOS.
- Chipset > South Bridge > USB Configuration: set XHCI to Disabled and USB 2.0 (EHCI) to Enabled. Without doing this, the USB drive won't boot.
- Boot to the VyOS installer and install as usual.

Warning the interface labels on my device are backwards; the left-most "LAN4" port is eth0 and the right-most "LAN1" port is eth3.

23.5 Acrosser AND-J190N1

11/22/2016. This microbox network appliance was build to create OpenVPN bridges. It can saturate a 100Mbps link.

It is a small (serial console only) PC with 6 Gb LAN http://www.acrosser.com/upload/AND-J190_J180N1-2.pdf

You may have to add your own RAM and HDD/SSD. There is no VGA connector. But Acrosser provides a DB25 adapter for the VGA header on the motherboard (not used).

23.5.1 BIOS Settings:

First thing you want to do is getting a more user friendly console to configure BIOS. Default VT100 brings a lot of issues. Configure VT100+ instead.

For practical issues change speed from 115200 to 9600. 9600 is the default speed at which both linux kernel and VyOS will reconfigure the serial port when loading.

Connect to serial (115200bps). Power on the appliance and press Del in the console when requested to enter BIOS settings.

Advanced > Serial Port Console Redirection > Console Redirection Settings:

- Terminal Type : VT100+
- Bits per second : 9600

Save, reboot and change serial speed to 9600 on your client.

Some options have to be changed for VyOS to boot correctly. With XHCI enabled the installer can't access the USB key. Enable EHCI instead.

Reboot into BIOS, Chipset > South Bridge > USB Configuration:

- Disable XHCI
- Enable USB 2.0 (EHCI) Support

23.5.2 Install VyOS:

Create a VyOS bootable USB key. I Used the 64bits iso (VyOS 1.1.7) and live USB installer (http://www.linuxliveusb.com/)

I'm not sure if it helps the process but I changed default option to live-serial (line "default xxxx") on the USB key under syslinux/syslinux.cfg.

I connected the key to one black USB port on the back and powered on. The first VyOS screen has some readability issues. Press enter to continue.

Then VyOS should boot and you can perform the `install image`

CHAPTER 24

Migrate from Vyatta Core

VyOS 1.x.x line aims to preserve backward compatibility and provide a safe upgrade ath for existing Vyatta Core users. You may think of 1.0.0 as VC7.0.

24.1 Vyatta release compatiblity

Vyatta Core releases from **6.5** to **6.6** should be 100% compatible.

Vyatta Core **6.4** and earlier may have incompatibilities. In **6.5** the "modify" firewall was removed and replaced with `set policy route` command family, and old config cannot be automatically converted. You will have to adapt it to post-6.5 syntax manually.

Note: Also, in **6.5** remote access VPN interfaces were renamed from pppX to l2tpX and pptpX, so if you are using zone-policy in pre-6.5 versions, make sure to change interface names in rules for remote access VPN.

24.2 Upgrade procedure

You just use `add system image`, as if it was a new VC release. The only thing is that is you want to verify image digital signature, you will have to add the public key.

```
vyatta@vyatta:~$ '''wget http://vyos.net/so3group_maintainers.key'''
Connecting to vyos.net (x.x.x.x:80)
so3group_maintainers 100% |******************************|  3125  --:--:-- ETA
vyatta@vyatta:~$ '''sudo apt-key add so3group_maintainers.key'''
OK
vyatta@vyatta:~$
```

Next, we can add the VyOS image.

Note: Vyatta doesn't support HTTP redirects for `add system image` and http://mirror.vyos.net HTTP load-balancer links will not work. Instead, choose one of the mirrors and get a direct link.

This example uses 1.0.0 image, however, it's better to install the latest release.

```
vyatta@vyatta:~$ show system image
The system currently has the following image(s) installed:
  1: VC6.6R1 (default boot) (running image)

vyatta@vyatta:~$ add system image http://0.uk.mirrors.vyos.net/iso/release/1.0.0/vyos-
↪1.0.0-amd64.iso
 Trying to fetch ISO file from http://0.uk.mirrors.vyos.net/iso/release/1.0.0/vyos-1.
↪0.0-amd64.iso
   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                  Dload  Upload   Total   Spent    Left  Speed
 100  223M  100  223M    0     0   960k      0  0:03:57  0:03:57 --:--:--  657k
 ISO download succeeded.
 Checking for digital signature file...
   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                  Dload  Upload   Total   Spent    Left  Speed
 100   836  100   836    0     0   4197      0 --:--:-- --:--:-- --:--:--  4287
 Found it.  Checking digital signature...
 gpg: directory `/root/.gnupg' created
 gpg: new configuration file `/root/.gnupg/gpg.conf' created
 gpg: WARNING: options in `/root/.gnupg/gpg.conf' are not yet active during this run
 gpg: keyring `/root/.gnupg/pubring.gpg' created
 gpg: Signature made Sun Dec 22 16:51:42 2013 GMT using RSA key ID A442E6E9
 gpg: /root/.gnupg/trustdb.gpg: trustdb created
 gpg: Good signature from "SO3 Group Maintainers <maintainers@so3group.net>"
 gpg: WARNING: This key is not certified with a trusted signature!
 gpg:          There is no indication that the signature belongs to the owner.
 Primary key fingerprint: DD5B B405 35E7 F6E3 4278  1ABF B744 E25A A442 E6E9
 Digital signature is valid.
 Checking MD5 checksums of files on the ISO image...OK.
 Done!
 What would you like to name this image? [1.0.0]: '''[return]'''
 OK.  This image will be named: 1.0.0
 Installing "1.0.0" image.
 Copying new release files...
 Would you like to save the current configuration
 directory and config file? (Yes/No) [Yes]: '''[return]'''
 Copying current configuration...
 Would you like to save the SSH host keys from your
 current configuration? (Yes/No) [Yes]: '''[return]'''
 Copying SSH keys...
 Setting up grub configuration...
 Done.
 vyatta@vyatta:~$ '''show system image'''
 The system currently has the following image(s) installed:

    1: 1.0.0 (default boot)
    2: VC6.6R1 (running image)

 vyatta@vyatta:~$
```

Upon reboot, you should have a working installation of VyOS.

You can go back to your Vyatta install using the `set system image default-boot` command and selecting the your previous Vyatta image.

Note: Future releases of VyOS will break the direct upgrade path from Vyatta core. Please upgrade through an intermediate VyOS version e.g. VyOS 1.2.x.

CHAPTER 25

Building VyOS using Docker

This will guide you though the process of building a VyOS ISO yourself using Docker and works best on a fresh installation of Debain 8 (Jessie).

Note: Starting with VyOS 1.2 the developers have decided to change their release model. VyOS is now **free as in speech, but not as in beer**, meaning that while VyOS is still an open source project, the release ISO's are no longer free and can only be obtained via subscription, or by contributing to the community. Since the source code is still public you can build your own ISO using the process described here.

Installing Docker and it's prequisites

```
root@build:~$ apt update
root@build:~$ apt install apt-transport-https ca-certificates curl gnupg2 software-
↪properties-common
root@build:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key␣
↪add -
root@build:~$ add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/
↪debian $(lsb_release -cs) stable"
root@build:~$ apt update
root@build:~$ apt install docker-ce
```

Adding you user to the docker group to be able to execute the `docker` command without sudo.

```
root@build:~$ usermod -aG docker user
```

Note: It is recommended to use a non-root user from here on out.

Note: The build process needs to be built on a local file system, building on SMB or NFS shares is not supported!

Cloning the vyos-build crux branch and creating the docker container

```
user@build:~$ git clone -b crux --single-branch https://github.com/vyos/vyos-build.git
user@build:~$ cd vyos-build
user@build:~/vyos-build$ docker build -t vyos-builder docker
```

Running the container and building the ISO

```
user@build:~$ docker run --rm -it --privileged -v $(pwd):/vyos -w /vyos vyos-builder␣
↪bash
vyos_bld@d4220bb519a0:/vyos# ./configure --architecture amd64 --build-by "your@email.
↪tld" --build-type release --version 1.2.0
vyos_bld@d4220bb519a0:/vyos# sudo make iso
```

You may use these options to customize you ISO code-block:: sh

-h, --help show this help message and exit

--architecture ARCHITECTURE Image target architecture (amd64 or i586 or armhf)

--build-by BUILD_BY Builder identifier (e.g. jrandomhacker@example.net)

--custom-package CUSTOM_PACKAGES Custom packages to install from repositories

--build-type BUILD_TYPE Build type, release or development

--debian-security-mirror DEBIAN_SECURITY_MIRROR Debian security updated mirror

--version VERSION Version number (release builds only)

--debian-mirror DEBIAN_MIRROR Debian repository mirror for ISO build

--vyos-mirror VYOS_MIRROR VyOS package mirror

--pbuilder-debian-mirror PBUILDER_DEBIAN_MIRROR Debian repository mirror for pbuilder env bootstrap

--debug Enable debug output

--custom-apt-entry CUSTOM_APT_ENTRY Custom APT entry

--custom-apt-key CUSTOM_APT_KEY Custom APT key file

Your freshly built ISO should now be in the build directory. Good luck!

CHAPTER 26

Issues and Feature requests

26.1 Bug Report / Issue

Issues or bugs are found in any software project. VyOS is not an exception.

All issues should be reported to the developers. This lets the developers know what is not working properly. Without this sort of feedback every developer will believe that everything is working correctly.

26.1.1 I have found a bug, what should I do?

When you believe you have found a bug, it is always a good idea to verify the issue prior to opening a bug request.

- Consult the documentation to ensure that you have configured your system correctly.
- Get community support slack or forum.

26.1.2 Ensure that the problem is reproducible

When you are able to verify that it is actually a bug, spend some time to document how to reproduce the issue. This documentation can be invaluable. When you wish to have a developer fix a bug that you found, helping them reproduce the issue benefits everyone. Be sure to include information about the hardware you are using, commands that you were running, any other activities that you may have been doing at the time. This additional information can be very useful.

- What were you attempting to achieve?
- What was the configuration prior to the change?
- What commands did you use?

26.1.3 Include output

The output you get when you find a bug can provide lots of information. If you get an error message on the screen, copy it exactly. Having the exact message can provide detail that the developers can use. Like wise if you have any

log messages that also are from the time of the issue, include those. They may also contain information that is helpful for the development team.

26.1.4 Report a Bug

Create an account on the VyOS Phabricator. VyOS Phabricator is at https://phabricator.vyos.net. To create a Bugreport use the quick link in the left side under the specific project.

- provide as much information as you can
- which version you use
- what is to do to reproduce the bug

26.2 Feature Request

To send a Feature Request please search Phabricator if there is already a feature request targeting your request. You can enhance it or if you don't find one create a new one by use the quick link in the left side under the specific project.

CHAPTER 27

Development

The source code is hosted on GitHub under VyOS organization github.com/vyos

The code is split into modules. VyOS is composed of multiple individual packages, some of them are periodically synced with upstream, so keeping the whole source under a single repository would be very inconvenient and slow. There is now an ongoing effort to consolidate all VyOS-specific framework/config packages into vyos-1x package, but the basic structure is going to stay the same, just with fewer packages.

The repository that contains the ISO build script is vyos-build. The README will guide you to use the this top level repository.

27.1 Submit a patch

Patches are always welcome. You should follow some procedures and guidelines though.

27.1.1 Before you make a patch

In a big system, such as VyOS, that is comprised of multiple components, it's impossible to keep track of all the changes and bugs/feature requests in one's head. We use a bugtracker for it ("issue tracker" would be a better term, but this one stuck).

This information is used in two ways:

- Keep track of the progress (what we've already done in this branch and what we still need to do).
- Prepare release notes.

To make this approach work, every change must be associated with a bug number and componant. If there is no bug/enhancement request for the changes you are going to make, you must create a bugtracker entry first. Once there is a bugtracker entry about it, you should reference in your commit message, as in:

```
[vyos build] T1327: add serial console (115200,8n1) to ISO kernel command-line
[vyos config] T1397: Rewrite the config merge script
```

If there is no reference to an item in our bugtracker the pull request will be rejected.

Patch limits:

- If there is a bug that has multiple tasks than it is ok to reference multiple items in a commit and pull request.
- Multiple bugs can not be fixed in one patch if they have no reference to each other.
- We only accept bugfixes in packages other than vyos-1x.
- No new functionality should use old style templates and perl/shell code, use python.

27.1.2 How to make a patch

We only accept patches in git format, because traditional UNIX patch is only good if the code it's going to be applied to doesn't change. If the code changes, merge will fail and the patch will need manual editing, even if there are no real conflicting changes.

Git keeps more information and uses more sophisticated merge algorithms, which makes a fake conflict a rare occurrence. For the same reason you should always make a patch against the latest current branch.

You can make a pull request on github.com/vyos.

27.1.3 Find the package

Suppose you want to make a change in the webproxy script.

You can find its package with "dpkg -S":

```
# dpkg -S /opt/vyatta/sbin/vyatta-update-webproxy.pl
vyatta-webproxy: /opt/vyatta/sbin/vyatta-update-webproxy.pl
```

This means it's in vyatta-webproxy package.

27.1.4 Make a patch using a fork

Fork the repository in github and make a clone.

```
git clone https://github.com/<user>/vyatta-webproxy.git
```

Set your name and email in the git config:

```
git config user.name "J. Random Hacker"
git config user.email "jrhacker@example.net"
```

Make your changes and save them. Then do the following for all changes files:

```
git add myfile

# Or, for a whole dir:
git add somedir/*
```

Commit the changes:

```
git commit
```

Please use meaningful commit descriptions and don't forget to reference the bug number there! Now submit the patch, push and make a pull request.

27.1.5 Make a patch to mail or attach to an item

clone the repository.

```
git clone https://github.com/vyos/vyatta-webproxy.git
```

Set your name and email in the git config:

```
git config user.name "J. Random Hacker"
git config user.email "jrhacker@example.net"
```

Make your changes and save them. Then do the following for all changes files:

```
git add myfile

# Or, for a whole dir:
git add somedir/*
```

Commit the changes:

```
git commit
```

Please use meaningful commit descriptions and don't forget to reference the bug number there! Export the patch and send it to maintainers@vyos.net or attach to the bug.

```
git format-patch

# Or, for multiple commits, suppose you made two:
git format-patch -2
```

27.1.6 Make a patch using a feature branch (maintainers only)

checkout the current branch and make sure it is up to date.

```
git clone https://github.com/vyos/vyatta-webproxy.git
git checkout current
git pull origin current
```

Set your name and email in the git config:

```
git config user.name "J. Random Hacker"
git config user.email "jrhacker@example.net"
```

Create the feature branch:

```
git checkout -b <feature branch name>
```

Make your changes and save them. Then do the following for all changes files:

```
git add myfile

# Or, for a whole dir:
git add somedir/*
```

Commit the changes:

```
git commit
```

Please use meaningful commit descriptions and don't forget to reference the bug number there!

Rebase on the current repo if needed and push your branch

```
git checkout current
git pull origin current
git checkout <feature branch name>
git rebase current
git push -u origin <feature branch name>
```

Now you create a pull request.

CHAPTER 28

VyOS CLI

The bash completion in VyOS is defined in *templates*. Templates are text files stored in a directory tree, where directory names define command names, and template files define command behaviour. Befor VyOS 1.2.x this files were created by hand. After a complex redesing process the new style template are in XML.

XML interface definitions for VyOS come with a RelaxNG schema and are located in the vyos-1x module. This schema is a slightly modified schema from VyConf alias VyOS 2.0 So VyOS 1.2.x interface definitions will be reusable in Nextgen VyOS Versions with very minimal changes.

The great thing about schemas is not only that people can know the complete grammar for certain, but also that it can be automatically verified. The *scripts/build-command-templates* script that converts the XML definitions to old style templates also verifies them against the schema, so a bad definition will cause the package build to fail. I do agree that the format is verbose, but there is no other format now that would allow this. Besides, a specialized XML editor can alleviate the issue with verbosity.

28.1 Example XML File

```
<?xml version="1.0"?>
<!-- Cron configuration -->
<interfaceDefinition>
  <node name="system">
    <children>
      <node name="task-scheduler">
        <properties>
          <help>Task scheduler settings</help>
        </properties>
        <children>
          <tagNode name="task" owner="${vyos_conf_scripts_dir}/task_scheduler.py">
            <properties>
              <help>Scheduled task</help>
              <valueHelp>
                <format>&lt;string&gt;</format>
                <description>Task name</description>
```

(continues on next page)

(continued from previous page)

```
            </valueHelp>
            <priority>999</priority>
          </properties>
          <children>
            <leafNode name="crontab-spec">
              <properties>
                <help>UNIX crontab time specification string</help>
              </properties>
            </leafNode>
            <leafNode name="interval">
              <properties>
                <help>Execution interval</help>
                <valueHelp>
                  <format>&lt;minutes&gt;</format>
                  <description>Execution interval in minutes</description>
                </valueHelp>
                <valueHelp>
                  <format>&lt;minutes&gt;m</format>
                  <description>Execution interval in minutes</description>
                </valueHelp>
                <valueHelp>
                  <format>&lt;hours&gt;h</format>
                  <description>Execution interval in hours</description>
                </valueHelp>
                <valueHelp>
                  <format>&lt;days&gt;d</format>
                  <description>Execution interval in days</description>
                </valueHelp>
                <constraint>
                  <regex>[1-9]([0-9]*)([mhd]{0,1})</regex>
                </constraint>
              </properties>
            </leafNode>
            <node name="executable">
              <properties>
                <help>Executable path and arguments</help>
              </properties>
              <children>
                <leafNode name="path">
                  <properties>
                    <help>Path to executable</help>
                  </properties>
                </leafNode>
                <leafNode name="arguments">
                  <properties>
                    <help>Arguments passed to the executable</help>
                  </properties>
                </leafNode>
              </children>
            </node>
          </children>
        </tagNode>
      </children>
    </node>
  </children>
</node>
</interfaceDefinition>
```

28.2 Configuration mode command definitions

Command definitions are purely declarative, and cannot contain any logic. All logic for generating config files for target applications, restarting services and so on is implemented in configuration scripts instead.

28.2.1 Command syntax guidelines

Use of numbers

Use of numbers in command names **should** be avoided unless a number is a part of a protocol name or similar. Thus, `protocols ospfv3` is perfectly fine, but something like `server-1` is questionable at best.

28.2.2 Help string guidelines

To ensure uniform look and feel, and improve readability, we should follow a set of guidelines consistently.

Capitalization and punctuation

The first word of every help string **must** be capitalized. There **must not** be a period at the end of help strings.

Rationale: this seems to be the unwritten standard in network device CLIs, and a good aesthetic compromise.

Examples:

- Good: "Frobnication algorithm"
- Bad: "frobnication algorithm"
- Bad: "Frobnication algorithm."
- Horrible: "frobnication algorithm."

Use of abbreviations and acronyms

Abbreviations and acronyms **must** be capitalized.

Examples:

- Good: "TCP connection timeout"
- Bad: "tcp connection timeout"
- Horrible: "Tcp connectin timeout"

Acronyms also **must** be capitalized to visually distinguish them from normal words:

Examples:

- Good: RADIUS (as in remote authentication for dial-in user services)
- Bad: radius (unless it's about the distance between a center of a circle and any of its points)

Some abbreviations are traditionally written in mixed case. Generally, if it contains words "over" or "version", the letter **should** be lowercase. If there's an accepted spelling (especially if defined by an RFC or another standard), it **must** be followed.

Examples:

- Good: PPPoE, IPsec
- Bad: PPPOE, IPSEC
- Bad: pppoe, ipsec

Use of verbs

Verbs **should** be avoided. If a verb can be omitted, omit it.

Examples:

- Good: "TCP connection timeout"
- Bad: "Set TCP connection timeout"

If a verb is essential, keep it. For example, in the help text of *set system ipv6 disable-forwarding*, "Disable IPv6 forwarding on all interfaces" is a perfectly justified wording.

Prefer infinitives

Verbs, when they are necessary, **should** be in their infinitive form.

Examples:

- Good: "Disable IPv6 forwarding"
- Bad: "Disables IPv6 forwarding"

28.3 Mapping old node.def style to new XML definitions

Old concept/syntax	New syntax	Notes
mynode/node.def	<node name="mynode"> </node>	Leaf nodes (nodes with values) use <leafNode> tag instead
mynode/node.tag , tag:	<tagNode name="mynode> </node>	
help: My node	<properties> <help>My node</help>	
val_help: <format>; some string	<properties> <valueHelp> <format> format </format> <description> some string </description>	Do not add angle brackets around the format, they will be inserted automatically
syntax:expression: pattern	<properties> <constraint> <regex> . . .	<constraintErrorMessage> will be displayed on failure
syntax:expression: $VAR(@) in "foo", "bar", "baz"	None	Use regex
syntax:expression: exec . . .	<properties> <constraint> <validator> <name ="foo" argument="bar">	"${vyos_libexecdir}/validators/foo bar $VAR(@)" will be executed, <constraintErrorMessage> will be displayed on failure
syntax:expression: (arithmetic expression)	None	External arithmetic validator may be added if there's demand, complex validation is better left to commit-time scripts
priority: 999	<properties> <priority>999</priority>	Please leave a comment explaining why the priority was chosen (e.g. "after interfaces are configured")
multi:	<properties> <multi/>	Only applicable to leaf nodes
allowed: echo foo bar	<properties> <completionHelp> <list> foo bar </list>	
allowed: cli-shell-api listNodes vpn ipsec esp-group	<properties> <completionHelp> <path> vpn ipsec esp-group </path> . . .	
allowed: /path/to/script	<properties> <completionHelp> <script> /path/to/script </script> . . .	
default:	None	Move default values to scripts
commit:expression:	None	All commit time checks should be in the verify() function of the script
begin:/create:/delete:	None	All logic should be in the scripts

CHAPTER 29

Python Coding Guidelines

The switch to the Python programming language for new code is not merely a change of the language, but a chance to rethink and improve the programming approach.

Let's face it: VyOS is full of spaghetti code where logic for reading the VyOS config, generating daemon configs, and restarting processes is all mixed up.

Python (or any other language, for that matter) does not provide automatic protection from bad design, so we need to also devise design guidelines and follow them to keep the system extensible and maintainable.

29.1 Configuration script structure and behaviour

```
import sys

from vyos.config import Config
from vyos.util import ConfigError

def get_config():
    vc = Config()
    # Convert the VyOS config to an abstract internal representation
    config = ...
    return config

def verify(config):
    # Verify that configuration is valid
    if invalid:
        raise ConfigError("Descriptive message")
    return True

def generate(config):
    # Generate daemon configs
    pass
```

(continues on next page)

(continued from previous page)

```
def apply(config):
    # Apply the generated configs to the live system
    pass

try:
    config = get_config()
    verify(config)
except ConfigError as e:
    print(e)
    sys.exit(1)
```

The **get_config()** function must convert the VyOS config to an abstract internal representation. No other function is allowed to call `vyos.config.Config` object methods directly. The rationale for it is that when config reads are mixed with other logic, it's very hard to change the config syntax since you need to weed out every occurence of the old syntax. If syntax-specific code is confined to a single function, the rest of the code can be left untouched as long as the internal representation remains compatible.

Another advantage is testability of the code. Mocking the entire config subsystem is hard, while constructing an internal representation by hand is way simpler.

The **verify()** function takes an internal representation of the config and checks if it's valid, otherwise it must raise `VyOSError` with an error message that describes the problem and possibly suggests how to fix it. It must not make any changes to the system. The rationale for it is again testability and, in the future when the config backend is ready and every script is rewritten in this fashion, ability to execute commit dry run ("commit test" like in JunOS) and abort commit before making any changes to the system if an error is found in any component.

The **generate()** function generates config files for system components.

The **apply()** function applies the generated configuration to the live system. It should use non-disruptive reload whenever possible. It may execute disruptive operations such as daemon process restart if a particular component does not support non-disruptive reload, or when the expected service degradation is minimal (for example, in case of auxillary services such as LLDPd). In case of high impact services such as VPN daemon and routing protocols, when non-disruptive reload is supported for some but not all types of configuration changes, scripts authors should make effort to determine if a configuration change can be done in a non-disruptive way and only resort to disruptive restart if it cannot be avoided.

Unless absolutely necessary, configuration scripts should not modify the active configuration of system components directly. Whenever at all possible, scripts should generate a configuration file or files that can be applied with a single command such as reloading a service through systemd init. Inserting statements one by one is particularly discouraged, for example, when configuring netfilter rules, saving them to a file and loading it with iptables-restore should always be preferred to executing iptables directly.

The **apply()** and **generate()** functions may `raise ConfigError` if, for example, the daemon failed to start with the updated config. It shouldn't be a substitute for proper config checking in the **verify()** function. All reasonable effort should be made to verify that generated configuration is valid and will be accepted by the daemon, including, when necessary, cross-checks with other VyOS configuration subtrees.

Exceptions, including `VyOSError` (which is raised by `vyos.config.Config` on improper config operations, such as trying to use `list_nodes()` on a non-tag node) should not be silenced or caught and re-raised as config error. Sure this will not look pretty on user's screen, but it will make way better bug reports, and help users (and most VyOS users are IT professionals) do their own debugging as well.

29.2 Coding guidelines

29.2.1 Language

Python 3 **shall** be used. How long can we keep Python 2 alive anyway?

No considerations for Python 2 compatibility **should** be taken.

29.2.2 Formatting

Tabs **shall not** be used. Every indentation level should be 4 spaces.

29.2.3 Text generation

Template processor **should** be used for generating config files. Built-in string formatting **may** be used for simple line-oriented formats where every line is self-contained, such as iptables rules. Template processor **must** be used for structured, multi-line formats such as those used by ISC DHCPd.

The default template processor for VyOS code is jinja2.

29.3 Code policy

When modifying the source code, remember these rules of the legacy elimination campaign:

- No new features in Perl
- No old style command definitions
- No code incompatible with Python3

CHAPTER 30

Upstream packages

Many base system packages are pulled straight from Debian's main and contrib repositories, but there are exceptions.

30.1 vyos-netplug

Due to issues in the upstream version that sometimes set interfaces down, a modified version is used.

The source is at https://github.com/vyos/vyos-netplug

In the future, we may switch to using systemd infrastructure instead.

Building it doesn't require a special procedure.

30.2 keepalived

Keepalived normally isn't updated to newer feature releases between Debian versions, so we are building it from source.

Debian does keep their package in git, but it's upstream tarball imported into git without its original commit history. To be able to merge new tags in, we keep a fork of the upstream repository with packaging files imported from Debian at http://github.com/vyos/keepalived-upstream

30.3 strongswan

Our StrongSWAN build differs from the upstream:

- strongswan-nm package build is disabled since we don't use NetworkManager
- Patches for DMVPN are merged in

The source is at https://github.com/vyos/vyos-strongswan

DMVPN patches are added by this commit: https://github.com/vyos/vyos-strongswan/commit/1cf12b0f2f921bfc51affa3b81226d4a3e9138e7

Our op mode scripts use the python-vici module, which is not included in Debian's build, and isn't quite easy to integrate in that build. For this reason we debianize that module by hand now, using this procedure:

0. Install https://pypi.org/project/stdeb/
1. *cd vyos-strongswan*
2. *./configure –enable-python-eggs*
3. *cd src/libcharon/plugins/vici/python*
4. *make*
5. *python3 setup.py –command-packages=stdeb.command bdist_deb*

The package ends up in deb_dist dir.

30.4 ppp

Properly renaming PPTP and L2TP interfaces to pptpX and l2tpX from generic and non-informative pppX requires a patch that is neither in the upstream nor in Debian.

We keep a fork of Debian's repo at https://github.com/vyos/ppp-debian

The patches for pre-up renaming are:

- https://github.com/vyos/ppp-debian/commit/e728180026a051d2a96396276e7e4ae022899e2d
- https://github.com/vyos/ppp-debian/commit/f29ba8d9ebb043335a096d70bcd07e9635bba2e3

Additionally, there's a patch for reopening the log file to better support logging to files, even though it's less essential: https://github.com/vyos/ppp-debian/commit/dd2ebd5cdcddb40230dc4cc43d374055ff374711

The patches were written by Stephen Hemminger back in the Vyatta times.

30.5 mdns-repeater

This package doesn't exist in Debian. A debianized fork is kept at https://github.com/vyos/mdns-repeater

No special build procedure is required.

30.6 udp-broadcast-relay

This package doesn't exist in Debian. A debianized fork is kept at https://github.com/vyos/udp-broadcast-relay

No special build procedure is required.

30.7 Linux kernel

TBD

30.8 Linux firmware

TBD

30.9 Intel drivers

TBD

30.10 accel-ppp

accel-ppp has been packaged for the use with vyos, due to the kernel dependencies for its modules.

- https://github.com/vyos/vyos-accel-ppp

Build instructions are being kept up to date on the repos Readme.

30.11 hvinfo

A fork with packaging changes for VyOS is kept at https://github.com/vyos/hvinfo

The original repo is at https://github.com/dmbaturin/hvinfo

It's an Ada program and requires GNAT and gprbuild for building, dependencies are properly specified so just follow debuild's suggestions.

30.12 Per-file modifications

vyos-replace package replaces the upstream dhclient-script with a modified version that is aware of the VyOS config.