VNetBuild Reference

Version 3.0.1 (Built 05 Aug 2016)

Contents

1 VNetBuild Reference **2**
- 1.1 Who should read this manual . . . 2
- 1.2 Where to get help . . . 2
- 1.3 Installation . . . 2
- 1.4 Licence . . . 3

2 Running and Configuring VNetBuild **4**

3 Manual Pages in Alphabetical Order **4**
- 3.1 vnetbuild(1) . . . 4
  - 3.1.1 NAME . . . 4
  - 3.1.2 SYNOPSIS . . . 4
  - 3.1.3 DESCRIPTION . . . 4
  - 3.1.4 COMMANDS . . . 4
  - 3.1.5 RUNNING COMMANDS IN A NAMESPACE . . . 5
  - 3.1.6 SEE ALSO . . . 6
- 3.2 vnetbuild.conf(5) . . . 7
  - 3.2.1 NAME . . . 7
  - 3.2.2 SYNOPSIS . . . 7
  - 3.2.3 DESCRIPTION . . . 7
  - 3.2.4 NAMESPACE DEFINITIONS . . . 8
  - 3.2.5 CONFIGURATION STATEMENTS . . . 8
  - 3.2.6 COMMON CUSTOM COMMANDS . . . 10
  - 3.2.7 EXAMPLE . . . 11
  - 3.2.8 LIMITATIONS . . . 12
  - 3.2.9 SEE ALSO . . . 12

The latest version of this manual is available online as a PDF, as single page HTML and also as multiple pages within the website.

1 VNetBuild Reference

1.1 Who should read this manual

This is a reference guide with specific detailed information on commands and configuration syntax for the VNetBuild tool. The reference is unlikely to be suitable for newcomers to the tools, except as a means to look up more information on a particular command.

For tutorials and guides to using FireHOL and VNetBuild, please visit the website.

1.2 Where to get help

The FireHOL website.

The mailing lists and archives.

The package comes with a complete set of manpages, a README and a brief INSTALL guide.

1.3 Installation

You can download tar-file releases by visiting the FireHOL website download area.

Unpack and change directory with:

```
tar xfz firehol-version.tar.gz
cd firehol-version
```

Options for the configure program can be seen in the INSTALL file and by running:

```
./configure --help
```

To build and install taking the default options:

```
./configure && make && sudo make install
```

Alternatively, just copy the `sbin/vnetbuild.in` file to where you want it. All of the common SysVInit command line arguments are recognised which makes it easy to deploy the script as a startup service.

Packages are available for most distributions and you can use your distribution's standard commands (e.g. aptitude, yum, etc.) to install these.

> **Note**
>
> Distributions do not always offer the latest version. You can see what the latest release is on the FireHOL website.

1.4 Licence

2 Running and Configuring VNetBuild

- vnetbuild(1) - start and stop networks of network namespaces
- vnetbuild.conf(5) - configuration file

3 Manual Pages in Alphabetical Order

3.1 vnetbuild(1)

3.1.1 NAME

vnetbuild - an easy to use but powerful namespace setup tool

3.1.2 SYNOPSIS

sudo vnetbuild *CONFIGFILE* { start | stop | status }

vnetbuild *CONFIGFILE* graphviz *OUTFILE*.{gv|png|pdf|ps}

3.1.3 DESCRIPTION

VNetBuild is a program that helps you set up groups of interconnected network namespaces, to simulate networks of any complexity without resorting to using real or virtual machines.

This is ideal for testing complex multi-host configurations with a minimal amount of resources on a single machine:

- Each namespace can have its own network setup, including firewall and QOS configuration.
- Commands can be run in the namespace and will have that specific view of the network, including running standard network tools and daemons.

Run without any arguments, `vnetbuild` will present some help on usage.

3.1.4 COMMANDS

start Sets up a series of network namespaces as defined in *CONFIGFILE*. `vnetbuild` creates interconnected network devices as specified in the configuration, sets up routing and runs any custom commands that are given within the namespace.

stop Removes any devices from the namespaces defined in *CONFIGFILE* and kills any processes running with the namespaces, then removes the namespaces themselves.

status For each namespace defined in *CONFIGFILE*, shows if it is active and if so its network devices and their configuration.

graphviz *OUTFILE* Generates a graph of the network defined in *CONFIGFILE*. This does not need root access, nor does it require the namespaces to have been started.

OUTFILE can be `png` `pdf` or `ps`. If the extension `gv` is given the output is a graphviz(7) file which you can process separately.

3.1.5 RUNNING COMMANDS IN A NAMESPACE

Once you have created a set of network namespaces, you can easily run any commands you want within them. If for instance you defined three hosts (`host_a` with IP `10.0.0.1`, `host_b` with IP `10.0.0.2` and `host_c` with IP `10.0.0.3`) connected via a common switch `sw0`:

```
# ping host_b and host_c from host_a
sudo ip netns exec host_a ping 10.0.0.2
sudo ip netns exec host_a ping 10.0.0.3

# use netcat to listen on host_a and send data from host_b
# (use two terminals to run the commands simultaneously)
sudo ip netns exec host_a nc -l -p 23
sudo ip netns exec host_b nc -q 0 10.0.0.1 23 < /etc/hosts

# capture traffic passing through the switch, then view it
sudo ip netns exec sw0 tcpdump -i switch -w capfile
wireshark capfile

# Use 'firehol panic' in host_b to block all traffic
# (you could equally load a full config etc.)
sudo ip netns exec host_b firehol panic

# this is now blocked
sudo ip netns exec host_a ping 10.0.0.2
```

```
# not blocked (host_b not involved)
sudo ip netns exec host_a ping 10.0.0.3

# obtain a shell for your regular user, only "in" host_c
sudo ip netns exec host_c sudo -i -u $USER
ip a | grep 10.0.0.3
```

3.1.6 SEE ALSO

- vnetbuild.conf(5) - VNetBuild configuration file
- firehol(1) - FireHOL program
- fireqos(1) - FireQOS program
- FireHOL Website
- VNetBuild Online PDF Manual
- VNetBuild Online Documentation

3.2 vnetbuild.conf(5)

3.2.1 NAME

vnetbuild.conf - VNetBuild configuration file

3.2.2 SYNOPSIS

```
host *ID*
    dev *DEVICE* [ *ID*/*PAIRDEV* ] [ *IP*/*MASK*... ]
    ...
    bridgedev *BRIDGE* [ *DEVICE*... ] [ *IP*/*MASK*... ]
    ...
    route *ROUTECMD*
    ...
    pre_up *DEVICE* *CUSTOMCMD*
    ...
    exec *CUSTOMCMD*
    ...

...

switch *ID*
    dev *DEVICE* [ *ID*/*PAIRDEV* ]
    ...
    pre_up *DEVICE* *CUSTOMCMD*
    ...
    exec *CUSTOMCMD*
    ...

...
```

3.2.3 DESCRIPTION

There is no default configuration file for vnetbuild(1); one must always be specified on the command line.

The configuration file defines a set of namespaces that will be operated on.

VNetBuild defines two types of namespace, a `host` and a `switch`. Any number of each may be specified, with any number of configuration statements in each.

Note The Linux kernel does not see any difference between a `host` and a `switch` namespace. VNetBuild provides the distinction to make it easy build full virtual networks.

3.2.4 NAMESPACE DEFINITIONS

Namespace definitions come in two types, `host` and `switch`. Simply provide a simple unique alphanumeric *ID*. Any subsequent statements apply to this namespace until the next `host` or `switch` statement.

A `host` definition is designed to work like a physical machine. It allows you to specify any number of `dev` entries for network interfaces, with their IP addresses. You can also define any number of Linux bridges with `bridgedev` to add your defined interfaces to.

A `host` also allows any number of custom `exec` commands for extensibility and provides a `route` statement to deal with the common case of wanting to add network routes to the host.

A `switch` definition is designed to work like a physical network switch. It allows you to add any number of `dev` entries (and also custom `exec` commands for extensibility) but nothing else.

In addition, `dev` entries in a `switch` may only specify device names, they cannot have an IP address associated. A `switch` has a bridge automatically created in it and all `dev` entries are automatically added to it.

3.2.5 CONFIGURATION STATEMENTS

dev *DEVICE ...* Define a virtual ethernet device, *DEVICE* in a `host` or `switch`.

Devices must exist in pairs. A `dev` must first be defined unpaired in a namespace, then some subsequent `dev` must define the pair:

```
host a
  dev veth0
host b
  dev vppp0 a/veth0
```

Any *DEVICE* name which is acceptable to the Linux kernel may be used. We recommend sticking to e.g. `veth0`, `vppp0` etc. to make it clear that they are virtual and also how you are thinking of the device in terms of your setup. Devices will be created as type `veth`, irrespective of what you call them.

Hosts may optionally specify one or more *IP/MASK* values which will be applied (along with the calculated broadcast address) automatically, e.g.:

```
host a
  dev veth0 10.0.0.1/8 192.168.1.2/24
host b
  dev vppp0 a/veth0 10.0.0.2/8 192.168.1.3/24
```

A `dev` may not specify an IP address if it is in a `switch`. Switches exist just to tie together multiple devices in hosts, just like a physical network switch.

bridgedev *BRIDGE ...* Define an ethernet bridge, *BRIDGE* in a `host`. These are setup automatically using ip(8) and shown with bridge(8).

A bridge can specify network devices from its own namespace to be automatically added, as well as its own IP address(es).

```
host a
  dev veth0
  dev veth1 otherns/vdev0
  bridgedev vbr0 veth0 veth1 10.0.0.3/8
```

Devices included in a bridge generally do not need their own IP address (although that is permitted).

Bridges cannot have a pair themselves, but any devices added to a bridge need a pair as usual.

route *ROUTECMD* Specify an additional network route for a `host`.

Most commonly to add a default route from hosts on a "LAN" to the machine that acts as a gateway, e.g.:

```
route default via 10.0.0.254
```

The syntax of *ROUTECMD* is anything that can fit this pattern:

```
ip route add ROUTECMD
```

See ip(8) and ip-route(8) for help adding routes. If you want to do anything more complex than simply adding routes, use the `exec` configuration statement.

pre_up *DEVICE CUSTOMCMD* Execute custom commands in a `host` or `switch` just before bringing up the specified device. All of the `pre_up` statements for a device are combined and executed in the namespace.

In addition to any explicitly defined interfaces, switches have an implicit bridge device called `switch` which can also be used in `pre_up` commands.

Bridges always start after other devices, so to run a command after all everything has been created but before any interfaces are up, you can make use of `pre_up` on the first defined `dev`.

See below for some common uses for custom `pre_up` and `exec` commands.

exec ***CUSTOMCMD*** Execute a custom command in a `host` or `switch` once the rest of the namespace setup is complete.

Once all the namespaces are created, the final step in setting each one up is to have its `exec` statements combined and executed.

It is roughly the equivalent to writing your own script and executing it after `vnetbuild start` has finished:

```
sudo ip netns exec myns ./myscript.sh
```

See below for some common uses for custom `pre_up` and `exec` commands.

3.2.6 COMMON CUSTOM COMMANDS

For the most part it doesn't matter whether these commands are used in `pre_up` or `exec` operations - the only difference is when they will execute, so e.g. if you want a firewall in place before any interfaces come up then start it from the `pre_up` of the first device. If you only want the firewall after all devices are up, put it in `exec`, e.g.:

```
host myfirewall
    ...
    exec firehol myfirewall.conf start
```

Forwarding is not enabled by the Linux kernel when a namespace is first created. This can be easily done for any hosts that need to forward traffic:

```
host mygateway
    ...
    exec echo 1 > /proc/sys/net/ipv4/ip_forward
```

The `exec` operates in the `mygateway` namespace so your host is not affected.

Bridges are created without STP being enabled. To enable STP to ensure loops are not created, the following can be done:

```
host myhost
    bridgedev vbr0 ...
    ...
    pre_up vbr0 echo 2 > /sys/class/net/vbr0/bridge/stp_state

switch myswitch
    ...
    pre_up switch echo 2 > /sys/class/net/vbr0/bridge/stp_state
```

You could also use `brctl stp vbr0 on` and `brctl stp switch on` instead of setting the values directly. To disable multicast snooping you can use exactly the same method e.g.:

```
switch myswitch
    ...
    pre_up switch echo 0 > /sys/class/net/switch/bridge/multicast_snooping
```

It is possible to run firehol within a namespace to set up custom

Logs from network namespaces are not included in the normal system logs. To enable iptables logging you must start an instance of ulogd(8) in the namespace and use *ULOG* or *NFLOG* logging. For FireHOL, that means set `FIREHOL_LOG_MODE=ULOG` or `FIREHOL_LOG_MODE=NFLOG`. Note that *NFLOG* only works with ulogd version 2.

The default configuration for ulogd(8) is `/etc/ulogd.conf`. Assuming the default place it will write iptables logs to is `/var/log/ulog/syslogemu.log` (otherwise change the `sed` command as required), it is simple to set up per-namespace logging:

```
host mygateway
  ...
  exec sed 's:/var/log/ulog/syslogemu.log:/var/log/ulog/mygateway.log:' /etc/ulogd.conf > $N
  exec /usr/sbin/ulogd -d -c $NSTMP/ulogd.conf
```

The `-d` flag to ulogd(8) makes it become a daemon; when `vnetbuild stop` executes it will automatically kill any programs running in the namespaces is is stopping, which includes the logging daemon.

The configuration file will get cleaned as soon as `vnetbuild start` is finished. To be able to access such files you need to write them to a location not under `$NSTMP` or create them outside the `vnetbuild` configuration altogether.

3.2.7 EXAMPLE

A simple LAN arrangement with two hosts, one of which is a gateway to third host:

```
host host01
  dev veth0 10.0.0.1/8
  dev vppp0 192.168.0.1/24
  exec echo 1 > /proc/sys/net/ipv4/ip_forward
  route default via 192.168.0.1

host host02
  dev veth0 10.0.0.2/8
```

```
  route default via 10.0.0.1

switch lan
  dev d01 host01/veth0
  dev d02 host02/veth0

host extern01
  dev veth0 host01/vppp0 192.168.0.254/24
  route default via 192.168.0.1
  exec echo 1 > /proc/sys/net/ipv4/ip_forward
```

3.2.8 LIMITATIONS

When created, the namespaces setup by `vnetbuild` are completely disconnected from any real network. There is no way of defining such a connection in the `vnetbuild` configuration as allowing it would lead to conflicts with the normal network setup tools and configuration files in most distributions.

It is possible to arrange your network so you can connect real devices into one or more network namespaces. For the general approach see this mailing list post.

3.2.9 SEE ALSO

- vnetbuild(1) - VNetBuild program
- FireHOL Website
- VNetBuild Online PDF Manual
- VNetBuild Online Documentation
- ip(8) - show/manipulate network devices
- ip-route(8) - routing table management
- bridge(8) - routing table management
- ulogd(8) - netfilter/iptables logging daemon