Proxmox VE Firewall

Proxmox Server Solutions GmbH
<support@proxmox.com>
version 8.2.2, Thu Apr 25 09:24:16 CEST 2024
↩Index

Table of Contents

Zones
Configuration Files
- Cluster Wide Setup
- Host Specific Configuration
- VM/Container Configuration

Firewall Rules
Security Groups
IP Aliases
- Standard IP Alias local_network

IP Sets
- Standard IP set management
- Standard IP set blacklist
- Standard IP set ipfilter-net*

Services and Commands
Default firewall rules
- Datacenter incoming/outgoing DROP/REJECT
- VM/CT incoming/outgoing DROP/REJECT

Logging of firewall rules
- Logging of user defined firewall rules

Tips and Tricks
- How to allow FTP
- Suricata IPS integration

Notes on IPv6
Ports used by Proxmox VE
nftables
- Installation and Usage
- Usage
- Helpful Commands

Proxmox VE Firewall provides an easy way to protect your IT infrastructure. You can setup firewall rules for all hosts inside a cluster, or define rules for virtual machines and containers. Features like firewall macros, security groups, IP sets and aliases help to make that task easier.

While all configuration is stored on the cluster file system, the `iptables`-based firewall service runs on each cluster node, and thus provides full isolation between virtual machines. The distributed nature of this system also provides much higher bandwidth than a central firewall solution.

The firewall has full support for IPv4 and IPv6. IPv6 support is fully transparent, and we filter traffic for both protocols by default. So there is no need to maintain a different set of rules for IPv6.

Zones

The Proxmox VE firewall groups the network into the following logical zones:

Host
: Traffic from/to a cluster node

VM
: Traffic from/to a specific VM

For each zone, you can define firewall rules for incoming and/or outgoing traffic.

Configuration Files

All firewall related configuration is stored on the proxmox cluster file system. So those files are automatically distributed to all cluster nodes, and the `pve-firewall` service updates the underlying `iptables` rules automatically on changes.

You can configure anything using the GUI (i.e. **Datacenter** → **Firewall**, or on a **Node** → **Firewall**), or you can edit the configuration files directly using your preferred editor.

Firewall configuration files contain sections of key-value pairs. Lines beginning with a `#` and blank lines are considered comments. Sections start with a header line containing the section name enclosed in `[` and `]`.

Cluster Wide Setup

The cluster-wide firewall configuration is stored at:

`/etc/pve/firewall/cluster.fw`

The configuration can contain the following sections:

`[OPTIONS]`
: This is used to set cluster-wide firewall options.

`ebtables: <boolean>` (*default* = `1`)
: Enable ebtables rules cluster wide.

`enable: <integer> (0 - N)`
: Enable or disable the firewall cluster wide.

`log_ratelimit: [enable=]<1|0> [,burst=<integer>] [,rate=<rate>]`
: Log ratelimiting settings

  `burst=<integer> (0 - N)` (*default* = `5`)
  : Initial burst of packages which will always get logged before the rate is applied

  `enable=<boolean>` (*default* = `1`)
  : Enable or disable log rate limiting

  `rate=<rate>` (*default* = `1/second`)
  : Frequency with which the burst bucket gets refilled

`policy_in: <ACCEPT | DROP | REJECT>`
: Input policy.

`policy_out: <ACCEPT | DROP | REJECT>`
: Output policy.

`[RULES]`
: This sections contains cluster-wide firewall rules for all nodes.

`[IPSET <name>]`
: Cluster wide IP set definitions.

`[GROUP <name>]`
: Cluster wide security group definitions.

`[ALIASES]`
: Cluster wide Alias definitions.

Enabling the Firewall

The firewall is completely disabled by default, so you need to set the enable option here:

```
[OPTIONS]
# enable firewall (cluster-wide
```

```
setting, default is disabled)
enable: 1
```

> If you enable the firewall, traffic to all hosts is blocked by default. Only exceptions is WebGUI(8006) and ssh(22) from your local network.

If you want to administrate your Proxmox VE hosts from remote, you need to create rules to allow traffic from those remote IPs to the web GUI (port 8006). You may also want to allow ssh (port 22), and maybe SPICE (port 3128).

> Please open a SSH connection to one of your Proxmox VE hosts before enabling the firewall. That way you still have access to the host if something goes wrong .

To simplify that task, you can instead create an IPSet called “management”, and add all remote IPs there. This creates all required firewall rules to access the GUI from remote.

Host Specific Configuration

Host related configuration is read from:

`/etc/pve/nodes/<nodename>/host.fw`

This is useful if you want to overwrite rules from `cluster.fw` config. You can also increase log verbosity, and set netfilter related options. The configuration can contain the following sections:

`[OPTIONS]`
: This is used to set host related firewall options.

`enable: <boolean>`
: Enable host firewall rules.

`log_level_in: <alert | crit | debug | emerg | err | info | nolog | notice | warning>`
: Log level for incoming traffic.

`log_level_out: <alert | crit | debug | emerg | err | info | nolog | notice | warning>`

Log level for outgoing traffic.

`log_nf_conntrack`: `<boolean>` (*default* = `0`)

Enable logging of conntrack information.

`ndp`: `<boolean>` (*default* = `0`)

Enable NDP (Neighbor Discovery Protocol).

`nf_conntrack_allow_invalid`: `<boolean>` (*default* = `0`)

Allow invalid packets on connection tracking.

`nf_conntrack_helpers`: `<string>` (*default* = ``)

Enable conntrack helpers for specific protocols. Supported protocols: amanda, ftp, irc, netbios-ns, pptp, sane, sip, snmp, tftp

`nf_conntrack_max`: `<integer> (32768 - N)` (*default* = `262144`)

Maximum number of tracked connections.

`nf_conntrack_tcp_timeout_established`: `<integer> (7875 - N)` (*default* = `432000`)

Conntrack established timeout.

`nf_conntrack_tcp_timeout_syn_recv`: `<integer> (30 - 60)` (*default* = `60`)

Conntrack syn recv timeout.

`nftables`: `<boolean>` (*default* = `0`)

Enable nftables based firewall (tech preview)

`nosmurfs`: `<boolean>`

Enable SMURFS filter.

`protection_synflood`: `<boolean>` (*default* = `0`)

Enable synflood protection

`protection_synflood_burst`: `<integer>` (*default* = `1000`)

Synflood protection rate burst by ip src.

`protection_synflood_rate`: `<integer>` (*default* = `200`)

Synflood protection rate syn/sec by ip src.

`smurf_log_level`: `<alert | crit | debug | emerg | err | info | nolog | notice | warning>`

Log level for SMURFS filter.

`tcp_flags_log_level`: `<alert | crit | debug | emerg | err | info | nolog | notice | warning>`
: Log level for illegal tcp flags filter.

`tcpflags`: `<boolean>` (*default* = `0`)
: Filter illegal combinations of TCP flags.

`[RULES]`
: This sections contains host specific firewall rules.

VM/Container Configuration

VM firewall configuration is read from:

```
/etc/pve/firewall/<VMID>.fw
```

and contains the following data:

`[OPTIONS]`
: This is used to set VM/Container related firewall options.

`dhcp`: `<boolean>` (*default* = `0`)
: Enable DHCP.

`enable`: `<boolean>` (*default* = `0`)
: Enable/disable firewall rules.

`ipfilter`: `<boolean>`
: Enable default IP filters. This is equivalent to adding an empty ipfilter-net<id> ipset for every interface. Such ipsets implicitly contain sane default restrictions such as restricting IPv6 link local addresses to the one derived from the interface's MAC address. For containers the configured IP addresses will be implicitly added.

`log_level_in`: `<alert | crit | debug | emerg | err | info | nolog | notice | warning>`
: Log level for incoming traffic.

`log_level_out`: `<alert | crit | debug | emerg | err | info | nolog | notice | warning>`
: Log level for outgoing traffic.

`macfilter`: `<boolean>` (*default* = `1`)
: Enable/disable MAC address filter.

`ndp`: `<boolean>` (*default* = `0`)
: Enable NDP (Neighbor Discovery Protocol).

`policy_in`: `<ACCEPT | DROP | REJECT>`

Input policy.

`policy_out: <ACCEPT | DROP | REJECT>`
: Output policy.

`radv: <boolean>`
: Allow sending Router Advertisement.

`[RULES]`
: This sections contains VM/Container firewall rules.

`[IPSET <name>]`
: IP set definitions.

`[ALIASES]`
: IP Alias definitions.

Enabling the Firewall for VMs and Containers

Each virtual network device has its own firewall enable flag. So you can selectively enable the firewall for each interface. This is required in addition to the general firewall `enable` option.

Firewall Rules

Firewall rules consists of a direction (`IN` or `OUT`) and an action (`ACCEPT`, `DENY`, `REJECT`). You can also specify a macro name. Macros contain predefined sets of rules and options. Rules can be disabled by prefixing them with `|`.

Firewall rules syntax

```
[RULES]

DIRECTION ACTION [OPTIONS]
|DIRECTION ACTION [OPTIONS] #
disabled rule

DIRECTION MACRO(ACTION)
[OPTIONS] # use predefined macro
```

The following options can be used to refine rule matches.

`--dest <string>`
: Restrict packet destination address. This can refer to a single IP address, an IP set (*+ipsetname*) or an IP alias definition.

You can also specify an address range like *20.34.101.207-201.3.9.99*, or a list of IP addresses and networks (entries are separated by comma). Please do not mix IPv4 and IPv6 addresses inside such lists.

`--dport <string>`

Restrict TCP/UDP destination port. You can use service names or simple numbers (0-65535), as defined in */etc/services*. Port ranges can be specified with *\d+:\d+*, for example *80:85*, and you can use comma separated list to match several ports or ranges.

`--icmp-type <string>`

Specify icmp-type. Only valid if proto equals *icmp* or *icmpv6*/*ipv6-icmp*.

`--iface <string>`

Network interface name. You have to use network configuration key names for VMs and containers (*net\d+*). Host related rules can use arbitrary strings.

`--log <alert | crit | debug | emerg | err | info | nolog | notice | warning>`

Log level for firewall rule.

`--proto <string>`

IP protocol. You can use protocol names (*tcp*/*udp*) or simple numbers, as defined in */etc/protocols*.

`--source <string>`

Restrict packet source address. This can refer to a single IP address, an IP set (*+ipsetname*) or an IP alias definition. You can also specify an address range like *20.34.101.207-201.3.9.99*, or a list of IP addresses and networks (entries are separated by comma). Please do not mix IPv4 and IPv6 addresses inside such lists.

`--sport <string>`

Restrict TCP/UDP source port. You can use service names or simple numbers (0-65535), as defined in */etc/services*. Port ranges can be specified with *\d+:\d+*, for example *80:85*, and you can use comma separated list to match several ports or ranges.

Here are some examples:

```
[RULES]
IN SSH(ACCEPT) -i net0
IN SSH(ACCEPT) -i net0 # a
comment
IN SSH(ACCEPT) -i net0 -source
192.168.2.192 # only allow SSH
from 192.168.2.192
IN SSH(ACCEPT) -i net0 -source
10.0.0.1-10.0.0.10 # accept SSH
for IP range
IN SSH(ACCEPT) -i net0 -source
10.0.0.1,10.0.0.2,10.0.0.3
#accept ssh for IP list
IN SSH(ACCEPT) -i net0 -source
+mynetgroup # accept ssh for
ipset mynetgroup
IN SSH(ACCEPT) -i net0 -source
myserveralias #accept ssh for
alias myserveralias

|IN SSH(ACCEPT) -i net0 #
disabled rule

IN  DROP # drop all incoming
packages
OUT ACCEPT # accept all outgoing
packages
```

Security Groups

A security group is a collection of rules, defined at cluster level, which can be used in all VMs' rules. For example you can define a group named “webserver” with rules to open the *http* and *https* ports.

```
# /etc/pve/firewall/cluster.fw

[group webserver]
IN  ACCEPT -p tcp -dport 80
IN  ACCEPT -p tcp -dport 443
```

Then, you can add this group to a VM’s firewall

```
# /etc/pve/firewall/<VMID>.fw

[RULES]
GROUP webserver
```

IP Aliases

IP Aliases allow you to associate IP addresses of networks with a name. You can then refer to those names:

- inside IP set definitions
- in `source` and `dest` properties of firewall rules

Standard IP Alias `local_network`

This alias is automatically defined. Please use the following command to see assigned values:

```
# pve-firewall localnet
local hostname: example
local IP address: 192.168.2.100
network auto detect:
192.168.0.0/20
using detected local_network:
192.168.0.0/20
```

The firewall automatically sets up rules to allow everything needed for cluster communication (corosync, API, SSH) using this alias.

The user can overwrite these values in the `cluster.fw` alias section. If you use a single host on a public network, it is better to explicitly assign the local IP address

```
#  /etc/pve/firewall/cluster.fw
[ALIASES]
local_network 1.2.3.4 # use the
single IP address
```

IP Sets

IP sets can be used to define groups of networks and hosts. You can refer to them with '+name` in the firewall rules' `source` and `dest` properties.

The following example allows HTTP traffic from the `management` IP set.

```
IN HTTP(ACCEPT) -source
+management
```

Standard IP set `management`

This IP set applies only to host firewalls (not VM firewalls). Those IPs are allowed to do normal management tasks (Proxmox VE GUI, VNC, SPICE, SSH).

The local cluster network is automatically added to this IP set (alias `cluster_network`), to enable inter-host cluster communication. (multicast,ssh,...)

```
# /etc/pve/firewall/cluster.fw

[IPSET management]
192.168.2.10
192.168.2.10/24
```

Standard IP set `blacklist`

Traffic from these IPs is dropped by every host's and VM's firewall.

```
# /etc/pve/firewall/cluster.fw

[IPSET blacklist]
77.240.159.182
213.87.123.0/24
```

Standard IP set `ipfilter-net*`

These filters belong to a VM's network interface and are mainly used to prevent IP spoofing. If such a set exists for an interface then any outgoing traffic with a source IP not matching its interface's corresponding ipfilter set will be dropped.

For containers with configured IP addresses these sets, if they exist (or are activated via the general `IP Filter` option in the VM's firewall's **options** tab), implicitly contain the associated IP addresses.

For both virtual machines and containers they also implicitly contain the standard MAC-derived IPv6 link-local address in order to allow the neighbor discovery protocol to work.

```
/etc/pve/firewall/<VMID>.fw

[IPSET ipfilter-net0] # only
```

```
allow specified IPs on net0
192.168.2.10
```

Services and Commands

The firewall runs two service daemons on each node:

- pvefw-logger: NFLOG daemon (ulogd replacement).
- pve-firewall: updates iptables rules

There is also a CLI command named `pve-firewall`, which can be used to start and stop the firewall service:

```
# pve-firewall start
# pve-firewall stop
```

To get the status use:

```
# pve-firewall status
```

The above command reads and compiles all firewall rules, so you will see warnings if your firewall configuration contains any errors.

If you want to see the generated iptables rules you can use:

```
# iptables-save
```

Default firewall rules

The following traffic is filtered by the default firewall configuration:

Datacenter incoming/outgoing DROP/REJECT

If the input or output policy for the firewall is set to DROP or REJECT, the following traffic is still allowed for all Proxmox VE hosts in the cluster:

- traffic over the loopback interface
- already established connections
- traffic using the IGMP protocol
- TCP traffic from management hosts to port 8006 in order to allow access to the web interface

- TCP traffic from management hosts to the port range 5900 to 5999 allowing traffic for the VNC web console
- TCP traffic from management hosts to port 3128 for connections to the SPICE proxy
- TCP traffic from management hosts to port 22 to allow ssh access
- UDP traffic in the cluster network to ports 5405-5412 for corosync
- UDP multicast traffic in the cluster network
- ICMP traffic type 3 (Destination Unreachable), 4 (congestion control) or 11 (Time Exceeded)

The following traffic is dropped, but not logged even with logging enabled:

- TCP connections with invalid connection state
- Broadcast, multicast and anycast traffic not related to corosync, i.e., not coming through ports 5405-5412
- TCP traffic to port 43
- UDP traffic to ports 135 and 445
- UDP traffic to the port range 137 to 139
- UDP traffic form source port 137 to port range 1024 to 65535
- UDP traffic to port 1900
- TCP traffic to port 135, 139 and 445
- UDP traffic originating from source port 53

The rest of the traffic is dropped or rejected, respectively, and also logged. This may vary depending on the additional options enabled in **Firewall** → **Options**, such as NDP, SMURFS and TCP flag filtering.

Please inspect the output of the

```
# iptables-save
```

system command to see the firewall chains and rules active on your system. This output is also included in a `System Report`, accessible over a node's subscription tab in the web GUI, or through the `pvereport` command-line tool.

VM/CT incoming/outgoing DROP/REJECT

This drops or rejects all the traffic to the VMs, with some exceptions for DHCP, NDP, Router Advertisement, MAC and IP filtering depending on the set configuration. The same rules for dropping/rejecting packets are inherited from the datacenter, while the exceptions for accepted incoming/outgoing traffic of the host do not apply.

Again, you can use iptables-save (see above) to inspect all rules and chains applied.

Logging of firewall rules

By default, all logging of traffic filtered by the firewall rules is disabled. To enable logging, the `loglevel` for incoming and/or outgoing traffic has to be set in **Firewall → Options**. This can be done for the host as well as for the VM/CT firewall individually. By this, logging of Proxmox VE's standard firewall rules is enabled and the output can be observed in **Firewall → Log**. Further, only some dropped or rejected packets are logged for the standard rules (see default firewall rules).

`loglevel` does not affect how much of the filtered traffic is logged. It changes a `LOGID` appended as prefix to the log output for easier filtering and post-processing.

`loglevel` is one of the following flags:

loglevel	LOGID
nolog	—
emerg	0
alert	1
crit	2
err	3
warning	4
notice	5
info	6

loglevel	LOGID
debug	7

A typical firewall log output looks like this:

```
VMID LOGID CHAIN TIMESTAMP
POLICY: PACKET_DETAILS
```

In case of the host firewall, `VMID` is equal to 0.

Logging of user defined firewall rules

In order to log packets filtered by user-defined firewall rules, it is possible to set a log-level parameter for each rule individually. This allows to log in a fine grained manner and independent of the log-level defined for the standard rules in **Firewall → Options**.

While the `loglevel` for each individual rule can be defined or changed easily in the web UI during creation or modification of the rule, it is possible to set this also via the corresponding `pvesh` API calls.

Further, the log-level can also be set via the firewall configuration file by appending a `-log <loglevel>` to the selected rule (see possible log-levels).

For example, the following two are identical:

```
IN REJECT -p icmp -log nolog
IN REJECT -p icmp
```

whereas

```
IN REJECT -p icmp -log debug
```

produces a log output flagged with the `debug` level.

Tips and Tricks

How to allow FTP

FTP is an old style protocol which uses port 21 and several other dynamic ports. So you need a rule to accept port 21. In addition, you need to load the `ip_conntrack_ftp` module. So please run:

```
modprobe ip_conntrack_ftp
```

and add `ip_conntrack_ftp` to `/etc/modules` (so that it works after a reboot).

Suricata IPS integration

If you want to use the Suricata IPS (Intrusion Prevention System), it's possible.

Packets will be forwarded to the IPS only after the firewall ACCEPTed them.

Rejected/Dropped firewall packets don't go to the IPS.

Install suricata on proxmox host:

```
# apt-get install suricata
# modprobe nfnetlink_queue
```

Don't forget to add `nfnetlink_queue` to `/etc/modules` for next reboot.

Then, enable IPS for a specific VM with:

```
# /etc/pve/firewall/<VMID>.fw

[OPTIONS]
ips: 1
ips_queues: 0
```

`ips_queues` will bind a specific cpu queue for this VM.

Available queues are defined in

```
# /etc/default/suricata
NFQUEUE=0
```

Notes on IPv6

The firewall contains a few IPv6 specific options. One thing to note is that IPv6 does not use the ARP protocol anymore, and instead uses NDP (Neighbor Discovery

Protocol) which works on IP level and thus needs IP addresses to succeed. For this purpose link-local addresses derived from the interface's MAC address are used. By default the `NDP` option is enabled on both host and VM level to allow neighbor discovery (NDP) packets to be sent and received.

Beside neighbor discovery NDP is also used for a couple of other things, like auto-configuration and advertising routers.

By default VMs are allowed to send out router solicitation messages (to query for a router), and to receive router advertisement packets. This allows them to use stateless auto configuration. On the other hand VMs cannot advertise themselves as routers unless the "Allow Router Advertisement" (`radv: 1`) option is set.

As for the link local addresses required for NDP, there's also an "IP Filter" (`ipfilter: 1`) option which can be enabled which has the same effect as adding an `ipfilter-net*` ipset for each of the VM's network interfaces containing the corresponding link local addresses. (See the Standard IP set `ipfilter-net*` section for details.)

Ports used by Proxmox VE

- Web interface: 8006 (TCP, HTTP/1.1 over TLS)
- VNC Web console: 5900-5999 (TCP, WebSocket)
- SPICE proxy: 3128 (TCP)
- sshd (used for cluster actions): 22 (TCP)
- rpcbind: 111 (UDP)
- sendmail: 25 (TCP, outgoing)
- corosync cluster traffic: 5405-5412 UDP
- live migration (VM memory and local-disk data): 60000-60050 (TCP)

nftables

As an alternative to `pve-firewall` we offer `proxmox-firewall`, which is an implementation of the Proxmox VE firewall

based on the newer nftables rather than iptables.

> `proxmox-firewall` is currently in tech preview. There might be bugs or incompatibilies with the original firewall. It is currently not suited for production use.

This implementation uses the same configuration files and configuration format, so you can use your old configuration when switching. It provides the exact same functionality with a few exceptions:

- REJECT is currently not possible for guest traffic (traffic will instead be dropped).
- Using the `NDP`, `Router Advertisement` or `DHCP` options will **always** create firewall rules, irregardless of your default policy.
- firewall rules for guests are evaluated even for connections that have conntrack table entries.

Installation and Usage

Install the `proxmox-firewall` package:

```
apt install proxmox-firewall
```

Enable the nftables backend via the Web UI on your hosts (Host > Firewall > Options > nftables), or by enabling it in the configuration file for your hosts (`/etc/pve/nodes/<node_name>/host.fw`):

```
[OPTIONS]

nftables: 1
```

> After enabling/disabling `proxmox-firewall`, all running VMs and containers need to be restarted for the old/new firewall to work properly.

After setting the `nftables` configuration key, the new `proxmox-firewall` service will take over. You can check if the new service is working by checking the systemctl status of `proxmox-firewall`:

```
systemctl status proxmox-firewall
```

You can also examine the generated ruleset. You can find more information about this in the section Helpful Commands. You should also check whether `pve-firewall` is no longer generating iptables rules, you can find the respective commands in the Services and Commands section.

Switching back to the old firewall can be done by simply setting the configuration value back to 0 / No.

Usage

`proxmox-firewall` will create two tables that are managed by the `proxmox-firewall` service: `proxmox-firewall` and `proxmox-firewall-guests`. If you want to create custom rules that live outside the Proxmox VE firewall configuration you can create your own tables to manage your custom firewall rules. `proxmox-firewall` will only touch the tables it generates, so you can easily extend and modify the behavior of the `proxmox-firewall` by adding your own tables.

Instead of using the `pve-firewall` command, the nftables-based firewall uses `proxmox-firewall`. It is a systemd service, so you can start and stop it via `systemctl`:

```
systemctl start proxmox-firewall
systemctl stop proxmox-firewall
```

Stopping the firewall service will remove all generated rules.

To query the status of the firewall, you can query the status of the systemctl service:

```
systemctl status proxmox-firewall
```

Helpful Commands

You can check the generated ruleset via the following command:

```
nft list ruleset
```

If you want to debug `proxmox-firewall` you can simply run the daemon in foreground with the `RUST_LOG` environment variable set to `trace`. This should provide you with detailed debugging output:

```
RUST_LOG=trace
/usr/libexec/proxmox/proxmox-
firewall
```

You can also edit the systemctl service if you want to have detailed output for your firewall daemon:

```
systemctl edit proxmox-firewall
```

Then you need to add the override for the `RUST_LOG` environment variable:

```
[Service]
Environment="RUST_LOG=trace"
```

This will generate a large amount of logs very quickly, so only use this for debugging purposes. Other, less verbose, log levels are `info` and `debug`.

Running in foreground writes the log output to STDERR, so you can redirect it with the following command (e.g. for submitting logs to the community forum):

```
RUST_LOG=trace
/usr/libexec/proxmox/proxmox-
firewall 2>
firewall_log_$(hostname).txt
```

It can be helpful to trace packet flow through the different chains in order to debug firewall rules. This can be achieved by setting `nftrace` to 1 for packets that you want to track. It is advisable that you do not set this

flag for **all** packets, in the example below we only examine ICMP packets.

```
#!/usr/sbin/nft -f
table bridge tracebridge
delete table bridge tracebridge

table bridge tracebridge {
    chain trace {
        meta l4proto icmp meta
nftrace set 1
    }

    chain prerouting {
        type filter hook
prerouting priority -350; policy
accept;
        jump trace
    }

    chain postrouting {
        type filter hook
postrouting priority -350;
policy accept;
        jump trace
    }
}
```

Saving this file, making it executable, and then running it once will create the respective tracing chains. You can then inspect the tracing output via the Proxmox VE Web UI (Firewall > Log) or via `nft monitor trace`.

The above example traces traffic on all bridges, which is usually where guest traffic flows through. If you want to examine host traffic, create those chains in the `inet` table instead of the `bridge` table.

> Be aware that this can generate a **lot** of log spam and slow down the performance of your networking stack significantly.

You can remove the tracing rules via running the following command:

```
nft delete table bridge
tracebridge
```

Version 8.2.2
Last updated Thu Apr 25 09:24:16 CEST 2024