Red Hat Enterprise Linux Release 9.2 Manual Pages on 'firewalld.direct.5' command

$ man firewalld.direct.5

```
FIREWALLD.DIRECT(5)          firewalld.direct          FIREWALLD.DIRECT(5)
```

NAME

firewalld.direct - firewalld direct configuration file

SYNOPSIS

/etc/firewalld/direct.xml

DEPRECATED

The direct interface has been deprecated. It will be removed in a future release. It is superseded by policies, see firewalld.policies(5).

DESCRIPTION

Direct configuration gives a more direct access to the firewall. It requires user to know basic ip(6)tables/ebtables concepts, i.e. table (filter/mangle/nat/...), chain (INPUT/OUTPUT/FORWARD/...), commands (-A/-D/-I/...), parameters (-p/-s/-d/-j/...) and targets (ACCEPT/DROP/REJECT/...). Direct configuration should be used only as a last resort when it's not possible to use firewalld.zone(5). See also Direct Options in firewall-cmd(1).

A firewalld direct configuration file contains information about permanent direct chains, rules and passthrough ...

This is the structure of a direct configuration file:

```
<?xml version="1.0" encoding="utf-8"?>
<direct>
  [ <chain ipv="ipv4|ipv6|eb" table="table" chain="chain"/> ]
```

```
      [ <rule ipv="ipv4|ipv6|eb" table="table" chain="chain" priority="priority"> args </rule> ]
      [ <passthrough ipv="ipv4|ipv6|eb"> args </passthrough> ]
    </direct>
```

direct

The mandatory direct start and end tag defines the direct. This tag can only be used once in a direct configuration file. There are no attributes for direct.

chain

Is an optional empty-element tag and can be used several times. It can be used to define names for additional chains. A chain entry has exactly three attributes:

ipv="ipv4|ipv6|eb"

The IP family where the chain will be created. This can be either ipv4, ipv6 or eb.

table="table"

The table name where the chain will be created. This can be one of the tables that can be used for iptables, ip6tables or ebtables. For the possible values, see TABLES section in the iptables, ip6tables or ebtables man pages.

chain="chain"

The name of the chain, that will be created. Please make sure that there is no other chain with this name already.

Please remember to add a rule or passthrough rule with an --jump or --goto option to connect the chain to another one.

rule

Is an optional element tag and can be used several times. It can be used to add rules to a built-in or added chain. A rule entry has exactly four attributes:

ipv="ipv4|ipv6|eb"

The IP family where the rule will be added. This can be either ipv4, ipv6 or eb.

table="table"

The table name where the rule will be added. This can be one of the tables that can be used for iptables, ip6tables or ebtables. For the possible values, see TABLES section in the iptables, ip6tables or ebtables man pages.

chain="chain"

The name of the chain where the rule will be added. This can be either a built-in chain or a chain that has been created with the chain tag. If the chain name is a built-in chain, then the rule will be added to chain_direct, else the supplied chain name is used. chain_direct is created internally for all built-in chains to make sure that the added rules do not conflict with the rules created by firewalld.

priority="priority"

The priority is used to order rules. Priority 0 means add rule on top of the chain, with a higher priority the rule will be added further down. Rules with the same priority are on the same level and the order of these rules is not fixed and may change. If you want to make sure that a rule will be added after another one, use a low priority for the first and a higher for the following.

The args can be any arguments of iptables or ip6tables, that do not conflict with the table or chain attributes.

passthrough

Is an optional element tag and can be used several times. It can be used to add rules to a built-in or added chain. A rule entry has exactly one attribute:

ipv="ipv4|ipv6|eb"

The IP family where the passthrough rule will be added. This can be either ipv4, ipv6 or eb.

The args can be any arguments of iptables or ip6tables.

The passthrough rule will be added to the chain directly. There is no mechanism like for the direct rule above. The user of the passthrough rule has to make sure that there will be no conflict with the rules created by firewalld.

CAVEATS

Depending on the value of FirewallBackend (see firewalld.conf(5)) direct rules behave differently in some scenarios.

Packet accept/drop precedence

Due to implementation details of netfilter inside the kernel, if FirewallBackend=nftables is used direct rules that ACCEPT packets don't actually cause the packets to be immediately accepted by the system. Those packets are still be subject to firewalld's nftables ruleset. This basically means there are two independent firewalls and packets must be accepted by both (iptables and nftables). As an aside, this scenario also occurs inside of nftables (again due to netfilter) if there are multiple chains attached to the same hook - it's not as simple as iptables vs nftables.

There are a handful of options to workaround the ACCEPT issue:

1. Rich Rules

   If a rich rule can be used, then they should always be preferred over direct rules. Rich Rules will be converted to the enabled FirewallBackend. See firewalld.richlanguage(5).

2. Blanket Accept

   Users can add an explicit accept to the nftables ruleset. This can be done by adding the interface or source to the trusted zone. This strategy is often employed by things that perform their own filtering such as: libvirt, podman, docker.

   Warning: This means firewalld will do no filtering on these packets. It must all be done via direct rules or out-of-band iptables rules.

3. Selective Accept

   Alternatively, enable only the relevant service, port, address, or otherwise in the appropriate zone.

4. Revert to the iptables backend

   A last resort is to revert to the iptables backend by setting FirewallBackend=iptables. Users should be aware that firewalld development focuses on the nftables backend.

For direct rules that DROP packets the packets are immediately dropped regardless of the value of FirewallBackend. As such, there is no special consideration needed.

Firewalld guarantees the above ACCEPT/DROP behavior by registering nftables hooks with a lower precedence than iptables hooks.

Direct interface precedence

With FirewallBackend=iptables firewalld's top-level internal rules apply before direct rules are executed. This includes rules to accept existing connections. In the past this has surprised users. As an example, if a user adds a direct rule to drop traffic on destination port 22 existing SSH sessions would continue to function, but new connections would be denied.

With FirewallBackend=nftables direct rules were deliberately given a higher precedence than all other firewalld rules. This includes rules to accept existing connections.

EXAMPLE

Denylisting of the networks 192.168.1.0/24 and 192.168.5.0/24 with logging and dropping early in the raw table:

```
<?xml version="1.0" encoding="utf-8"?>
<direct>
  <chain ipv="ipv4" table="raw" chain="denylist"/>
  <rule ipv="ipv4" table="raw" chain="PREROUTING" priority="0">-s 192.168.1.0/24 -j denylist</rule>
  <rule ipv="ipv4" table="raw" chain="PREROUTING" priority="1">-s 192.168.5.0/24 -j denylist</rule>
  <rule ipv="ipv4" table="raw" chain="denylist" priority="0">-m limit --limit 1/min -j LOG --log-prefix "denylisted: "</rule>
  <rule ipv="ipv4" table="raw" chain="denylist" priority="1">-j DROP</rule>
</direct>
```

SEE ALSO

firewall-applet(1), firewalld(1), firewall-cmd(1), firewall-config(1), firewalld.conf(5), firewalld.direct(5), firewalld.dbus(5), firewalld.icmptype(5), firewalld.lockdown-whitelist(5), firewall-offline-cmd(1), firewalld.richlanguage(5), firewalld.service(5), firewalld.zone(5), firewalld.zones(5), firewalld.policy(5), firewalld.policies(5), firewalld.ipset(5), firewalld.helper(5)

NOTES

firewalld home page:

http://firewalld.org

More documentation with examples:

http://fedoraproject.org/wiki/FirewallD

AUTHORS

Thomas Woerner <twoerner@redhat.com>

Developer

Jiri Popelka <jpopelka@redhat.com>

Developer

Eric Garver <eric@garver.life>

Developer