# Filter

*Document revision 2.8 (February 11, 2008, 4:14 GMT)*
This document applies to V3.0

# Table of Contents

Table of Contents
  Summary
  Quick Setup Guide
  Specifications
Firewall Filter
  Description
  Property Description
  Notes
Filter Applications
  Protect your RouterOS router
  Protecting the Customer's Network

# General Information

## Summary

The firewall implements packet filtering and thereby provides security functions that are used to manage data flow to, from and through the router. Along with the Network Address Translation it serve as a tool for preventing unauthorized access to directly attached networks and the router itself as well as a filter for outgoing traffic.

## Quick Setup Guide

- To add a firewall rule which drops all **TCP** packets that are destined to port **135** and going through the router, use the following command:

```
/ip firewall filter add chain=forward dst-port=135 protocol=tcp action=drop
```

- To deny acces to the router via Telnet (protocol TCP, port 23), type the following command:

```
/ip firewall filter add chain=input protocol=tcp dst-port=23 action=drop
```

- To only allow not more than 5 simultaneous connections from each of the clients, do the following:

```
/ip firewall filter add chain=forward protocol=tcp tcp-flags=syn connection-limit=6,32
action=drop
```

## Specifications

Packages required: ***system***
License required: ***level1 (P2P filters limited to 1), level3***
Home menu level: ***/ip firewall filter***
Standards and Technologies: ***IP***, ***RFC2113***

Hardware usage: *Increases with filtering rules count*

# Firewall Filter

Home menu level: ***/ip firewall filter***

## Description

Network firewalls keep outside threats away from sensitive data available inside the network. Whenever different networks are joined together, there is always a threat that someone from outside of your network will break into your LAN. Such break-ins may result in private data being stolen and distributed, valuable data being altered or destroyed, or entire hard drives being erased. Firewalls are used as a means of preventing or minimizing the security risks inherent in connecting to other networks. Properly configured firewall plays a key role in efficient and secure network infrastrure deployment.

MikroTik RouterOS has very powerful firewall implementation with features including:

- stateful packet inspection
- Layer-7 protocol detection
- peer-to-peer protocols filtering
- traffic classification by:
  - source MAC address
  - IP addresses (network or list) and address types (broadcast, local, multicast, unicast)
  - port or port range
  - IP protocols
  - protocol options (ICMP type and code fields, TCP flags, IP options and MSS)
  - interface the packet arrived from or left through
  - internal flow and connection marks
  - DSCP byte
  - packet content
  - rate at which packets arrive and sequence numbers
  - packet size
  - packet arrival time
  - and much more!

### General Filtering Principles

The firewall operates by means of firewall rules. A rule is a definitive form expression that tells the router what to do with a particular IP packet. Each rule consists of two parts that are the matcher which matches traffic flow against given conditions and the action which defines what to do with the mathched packets. Rules are organized in chains for better management.

The filter facility has three default chains: **input**, **forward** and **output** that are responsible for traffic coming from, throurh and to the router, respectively. New user-defined chains can be added, as necessary. Since these chains have no default traffic to match, rules with **action=jump** and relevant **jump-target** should be added to one or more of the three default chains.

## Filter Chains

As mentioned before, the firewall filtering rules are grouped together in chains. It allows a packet to be matched against one common criterion in one chain, and then passed over for processing against some other common criteria to another chain. For example a packet should be matched against the **IP address:port** pair. Of course, it could be achieved by adding as many rules with **IP address:port** match as required to the **forward** chain, but a better way could be to add one rule that matches traffic from a particular IP address, e.g.: `/ip firewall filter add src-address=1.1.1.2/32 jump-target="mychain"` and in case of successfull match passes control over the IP packet to some other chain, *id est* **mychain** in this example. Then rules that perform matching against separate ports can be added to **mychain** chain without specifying the IP addresses.

- **input** - used to process packets entering the router through one of the interfaces with the destination IP address which is one of the router's addresses. Packets passing through the router are not processed against the rules of the input chain
- **forward** - used to process packets passing through the router
- **output** - used to process packets originated from the router and leaving it through one of the interfaces. Packets passing through the router are not processed against the rules of the output chain

There are three predefined chains, which cannot be deleted:

When processing a chain, rules are taken from the chain in the order they are listed there from top to bottom. If a packet matches the criteria of the rule, then the specified action is performed on it, and no more rules are processed in that chain (the exception is the **passthrough** action). If a packet has not matched any rule within the chain, then it is accepted.

## Property Description

**action** (*accept* | *add-dst-to-address-list* | *add-src-to-address-list* | *drop* | *jump* | *log* | *passthrough* | *reject* | *return* | *tarpit*; default: **accept**) - action to undertake if the packet matches the rule

- **accept** - accept the packet. No action is taken, i.e. the packet is passed through and no more rules are applied to it
- **add-dst-to-address-list** - adds destination address of an IP packet to the address list specified by address-list parameter
- **add-src-to-address-list** - adds source address of an IP packet to the address list specified by address-list parameter
- **drop** - silently drop the packet (without sending the ICMP reject message)
- **jump** - jump to the chain specified by the value of the jump-target parameter
- **log** - each match with this action will add a message to the system log
- **passthrough** - ignores this rule and goes on to the next one
- **reject** - reject the packet and send an ICMP reject message

- **return** - passes control back to the chain from where the jump took place
- **tarpit** - captures and holds incoming TCP connections (replies with SYN/ACK to the inbound TCP SYN packet)

**address-list** (*name*) - specifies the name of the address list to collect IP addresses from rules having action=add-dst-to-address-list or action=add-src-to-address-list actions. These address lists could be later used for packet matching

**address-list-timeout** (*time*; default: **00:00:00**) - time interval after which the address will be removed from the address list specified by address-list parameter. Used in conjunction with add-dst-to-address-list or add-src-to-address-list actions

- **00:00:00** - leave the address in the address list forever

**chain** (*forward* | *input* | *outputname*) - specifies the chain to put a particular rule into. As the different traffic is passed through different chains, always be careful in choosing the right chain for a new rule. If the input does not match the name of an already defined chain, a new chain will be created

**comment** (*text*) - a descriptive comment for the rule. A comment can be used to identify rules form scripts

**connection-bytes** (*integerinteger*) - matches packets only if a given amount of bytes has been transfered through the particular connection

- **0** - means infinity, exempli gratia: connection-bytes=2000000-0 means that the rule matches if more than 2MB has been transfered through the relevant connection

**connection-limit** (*integernetmask*) - restrict connection limit per address or address block

**connection-mark** (*name*) - matches packets marked via mangle facility with particular connection mark

**connection-state** (*estabilished* | *invalid* | *new* | *related*) - interprets the connection tracking analysis data for a particular packet

- **estabilished** - a packet which belongs to an existing connection, exempli gratia a reply packet or a packet which belongs to already replied connection
- **invalid** - a packet which could not be identified for some reason. This includes out of memory condition and ICMP errors which do not correspond to any known connection. It is generally advised to drop these packets
- **new** - a packet which begins a new TCP connection
- **related** - a packet which is related to, but not part of an existing connection, such as ICMP errors or a packet which begins FTP data connection (the later requires enabled FTP connection tracking helper under /ip firewall service-port)

**connection-type** (*ftp* | *gre* | *h323* | *irc* | *mms* | *pptp* | *quake3* | *tftp*) - matches packets from related connections based on information from their connection tracking helpers. A relevant connection helper must be enabled under /ip firewall service-port

**content** (*text*) - the text packets should contain in order to match the rule

**dscp** (*integer*: 0..63) - DSCP (ex-ToS) IP header field value

**dst-address** (*IP addressnetmaskIP addressIP address*) - specifies the address range an IP packet is destined to. Note that console converts entered address/netmask value to a valid network address, i.e.:1.1.1.1/24 is converted to 1.1.1.0/24

**dst-address-list** (*name*) - matches destination address of a packet against user-defined address list

**dst-address-type** (*unicast* | *local* | *broadcast* | *multicast*) - matches destination address type of the IP packet, one of the:

- **unicast** - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case
- **local** - matches addresses assigned to router's interfaces
- **broadcast** - the IP packet is sent from one point to all other points in the IP subnetwork
- **multicast** - this type of IP addressing is responsible for transmission from one or more points to a set of other points

**dst-limit** (*integertimeintegerdst-address* | *dst-port* | *src-addresstime*) - limits the packet per second (pps) rate on a per destination IP or per destination port base. As opposed to the limit match, every destination IP address / destination port has it's own limit. The options are as follows (in order of appearance):

- **count** - maximum average packet rate, measured in packets per second (pps), unless followed by time option
- **time** - specifies the time interval over which the packet rate is measured
- **burst** - number of packets to match in a burst
- **mode** - the classifier(-s) for packet rate limiting
- **expire** - specifies interval after which recorded IP addresses / ports will be deleted

**dst-port** (*integer*: 0..65535*integer*: 0..65535) - destination port number or range

**fragment** (yes | no) - whether the packet is a fragment of an IP packet. Starting packet (i.e., first fragment) does not count. Note that is the connection tracking is enabled, there will be no fragments as the system automatically assembles every packet

**hotspot** (*multiple choice: auth* | *from-client* | *http* | *local-dst* | *to-client*) - matches packets received from clients against various HotSpot conditions. All values can be negated

- **auth** - true, if a packet comes from an authenticted HotSpotclient
- **from-client** - true, if a packet comes from any HotSpot client
- **http** - true, if a HotSpot client sends a packet to the address and port previously detected as his proxy server (Universal Proxy technique) or if the destination port is 80 and transparent proxying is enabled for that particular client
- **local-dst** - true, if a packet has local destination IP address
- **to-client** - true, if a packet is sent to a client

**icmp-options** (*integerinteger*) - matches ICMP Type:Code fields

**in-bridge-port** (*name*) - actual interface the packet has entered the router through (if bridged, this property matches the actual bridge port, while in-interface - the bridge itself)

**in-interface** (*name*) - interface the packet has entered the router through (if the interface is bridged, then the packet will appear to come from the bridge interface itself)

**ingress-priority** (*integer*: 0..63) - INGRESS (received) priority of the packet, if set (0 otherwise). The priority may be derived from either VLAN or WMM priority

**ipv4-options** (*any* | *loose-source-routing* | *no-record-route* | *no-router-alert* | *no-source-routing* | *no-timestamp* | *none* | *record-route* | *router-alert* | *strict-source-routing* | *timestamp*) - match ipv4 header options

- **any** - match packet with at least one of the ipv4 options

- **loose-source-routing** - match packets with loose source routing option. This option is used to route the internet datagram based on information supplied by the source
- **no-record-route** - match packets with no record route option. This option is used to route the internet datagram based on information supplied by the source
- **no-router-alert** - match packets with no router alter option
- **no-source-routing** - match packets with no source routing option
- **no-timestamp** - match packets with no timestamp option
- **record-route** - match packets with record route option
- **router-alert** - match packets with router alter option
- **strict-source-routing** - match packets with strict source routing option
- **timestamp** - match packets with timestamp

**jump-target** (*forward* | *input* | *outputname*) - name of the target chain to jump to, if the action=jump is used

**layer7-protocol** (*name*) - Layer 7 filter name as set in the /ip firewall layer7-protocol menu. Caution: this matcher needs high computational power

**limit** (*integertimeinteger*) - restricts packet match rate to a given limit. Usefull to reduce the amount of log messages

- **count** - maximum average packet rate, measured in packets per second (pps), unless followed by time option
- **time** - specifies the time interval over which the packet rate is measured
- **burst** - number of packets to match in a burst

**log-prefix** (*text*) - all messages written to logs will contain the prefix specified herein. Used in conjunction with action=log

**nth** (*integerinteger*: 0..15*integer*) - match a particular Nth packet received by the rule. One of 16 available counters can be used to count packets

- **every** - match every every+1th packet. For example, if every=1 then the rule matches every 2nd packet
- **counter** - specifies which counter to use. A counter increments each time the rule containing nth match matches
- **packet** - match on the given packet number. The value by obvious reasons must be between 0 and every. If this option is used for a given counter, then there must be at least every+1 rules with this option, covering all values between 0 and every inclusively.

**out-bridge-port** (*name*) - actual interface the packet is leaving the router through (if bridged, this property matches the actual bridge port, while out-interface - the bridge itself)

**out-interface** (*name*) - interface the packet is leaving the router through (if the interface is bridged, then the packet will appear to leave through the bridge interface itself)

**p2p** (*all-p2p* | *bit-torrent* | *blubster* | *direct-connect* | *edonkey* | *fasttrack* | *gnutella* | *soulseek* | *warez* | *winmx*) - matches packets from various peer-to-peer (P2P) protocols

**packet-mark** (*text*) - matches packets marked via mangle facility with particular packet mark

**packet-size** (*integer*: 0..65535*integer*: 0..65535) - matches packet of the specified size or size range in bytes

- **min** - specifies lower boundary of the size range or a standalone value

- **max** - specifies upper boundary of the size range

**port** (*port*) - matches if any (source or destination) port matches the specified list of ports or port ranges (note that the protocol must still be selected, just like for the regular src-port and dst-port matchers)

**protocol** (*ddp* | *egp* | *encap* | *ggp* | *gre* | *hmp* | *icmp* | *idrp-cmtp* | *igmp* | *ipencap* | *ipip* | *ipsec-ah* | *ipsec-esp* | *iso-tp4* | *ospf* | *pup* | *rdp* | *rspf* | *st* | *tcp* | *udp* | *vmtp* | *xns-idp* | *xtpinteger*) - matches particular IP protocol specified by protocol name or number. You should specify this setting if you want to specify ports

**psd** (*integertimeintegerinteger*) - attempts to detect TCP and UDP scans. It is advised to assign lower weight to ports with high numbers to reduce the frequency of false positives, such as from passive mode FTP transfers

- **WeightThreshold** - total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence
- **DelayThreshold** - delay for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence
- **LowPortWeight** - weight of the packets with privileged (<=1024) destination port
- **HighPortWeight** - weight of the packet with non-priviliged destination port

**random** (*integer*: 1..99) - matches packets randomly with given propability

**reject-with** (*icmp-admin-prohibited* | *icmp-echo-reply* | *icmp-host-prohibited* | *icmp-host-unreachable* | *icmp-net-prohibited* | *icmp-network-unreachable* | *icmp-port-unreachable* | *icmp-protocol-unreachable* | *tcp-resetinteger*) - alters the reply packet of reject action

**routing-mark** (*name*) - matches packets marked by mangle facility with particular routing mark

**src-address** (*IP addressnetmaskIP addressIP address*) - specifies the address range an IP packet is originated from. Note that console converts entered address/netmask value to a valid network address, i.e.:1.1.1.1/24 is converted to 1.1.1.0/24

**src-address-list** (*name*) - matches source address of a packet against user-defined address list

**src-address-type** (*unicast* | *local* | *broadcast* | *multicast*) - matches source address type of the IP packet, one of the:

- **unicast** - IP addresses used for one point to another point transmission. There is only one sender and one receiver in this case
- **local** - matches addresses assigned to router's interfaces
- **broadcast** - the IP packet is sent from one point to all other points in the IP subnetwork
- **multicast** - this type of IP addressing is responsible for transmission from one or more points to a set of other points

**src-mac-address** (*MAC address*) - source MAC address

**src-port** (*integer*: 0..65535*integer*: 0..65535) - source port number or range

**tcp-flags** (*ack* | *cwr* | *ece* | *fin* | *psh* | *rst* | *syn* | *urg*) - tcp flags to match

- **ack** - acknowledging data
- **cwr** - congestion window reduced
- **ece** - ECN-echo flag (explicit congestion notification)
- **fin** - close connection
- **psh** - push function

- **rst** - drop connection
- **syn** - new connection
- **urg** - urgent data

**tcp-mss** (*integer*: 0..65535) - matches TCP MSS value of an IP packet

**time** (*timetimesat* | *fri* | *thu* | *wed* | *tue* | *mon* | *sun*) - allows to create filter based on the packets' arrival time and date or, for locally generated packets, departure time and date

## Notes

Because the NAT rules are applied first, it is important to hold this in mind when setting up firewall rules, since the original packets might be already modified by the NAT

# Filter Applications

## Protect your RouterOS router

To protect your router, you should not only change admin's password but also set up packet filtering. All packets with destination to the router are processed against the ip firewall input chain. Note, that the input chain does not affect packets which are being transferred through the router.

```
/ ip firewall filter
add chain=input connection-state=invalid action=drop \
        comment="Drop Invalid connections"
add chain=input connection-state=established action=accept \
        comment="Allow Established connections"
add chain=input protocol=udp action=accept \
        comment="Allow UDP"
add chain=input protocol=icmp action=accept \
        comment="Allow ICMP"
add chain=input src-address=192.168.0.0/24 action=accept \
        comment="Allow access to router from known network"
add chain=input action=drop comment="Drop anything else"
```

## Protecting the Customer's Network

To protect the customer's network, we should check all traffic which goes through router and block unwanted. For icmp, tcp, udp traffic we will create chains, where will be droped all unwanted packets:

```
/ip firewall filter
add chain=forward protocol=tcp connection-state=invalid \
         action=drop comment="drop invalid connections"
add chain=forward connection-state=established action=accept \
         comment="allow already established connections"
add chain=forward connection-state=related action=accept \
         comment="allow related connections"
```

Block IP addreses called "bogons":

```
add chain=forward src-address=0.0.0.0/8 action=drop
add chain=forward dst-address=0.0.0.0/8 action=drop
add chain=forward src-address=127.0.0.0/8 action=drop
add chain=forward dst-address=127.0.0.0/8 action=drop
add chain=forward src-address=224.0.0.0/3 action=drop
add chain=forward dst-address=224.0.0.0/3 action=drop
```

Make jumps to new chains:

```
add chain=forward protocol=tcp action=jump jump-target=tcp
add chain=forward protocol=udp action=jump jump-target=udp
add chain=forward protocol=icmp action=jump jump-target=icmp
```

Create tcp chain and deny some tcp ports in it:

```
add chain=tcp protocol=tcp dst-port=69 action=drop \
         comment="deny TFTP"
add chain=tcp protocol=tcp dst-port=111 action=drop \
         comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=135 action=drop \
         comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=137-139 action=drop \
         comment="deny NBT"
add chain=tcp protocol=tcp dst-port=445 action=drop \
         comment="deny cifs"
add chain=tcp protocol=tcp dst-port=2049 action=drop comment="deny NFS"
add chain=tcp protocol=tcp dst-port=12345-12346 action=drop comment="deny NetBus"
add chain=tcp protocol=tcp dst-port=20034 action=drop comment="deny NetBus"
add chain=tcp protocol=tcp dst-port=3133 action=drop comment="deny BackOriffice"
add chain=tcp protocol=tcp dst-port=67-68 action=drop comment="deny DHCP"
```

Deny udp ports in udp chain:

```
add chain=udp protocol=udp dst-port=69 action=drop comment="deny TFTP"
add chain=udp protocol=udp dst-port=111 action=drop comment="deny PRC portmapper"
add chain=udp protocol=udp dst-port=135 action=drop comment="deny PRC portmapper"
add chain=udp protocol=udp dst-port=137-139 action=drop comment="deny NBT"
add chain=udp protocol=udp dst-port=2049 action=drop comment="deny NFS"
add chain=udp protocol=udp dst-port=3133 action=drop comment="deny BackOriffice"
```

Allow only needed icmp codes in icmp chain:

```
add chain=icmp protocol=icmp icmp-options=0:0 action=accept \
         comment="drop invalid connections"
add chain=icmp protocol=icmp icmp-options=3:0 action=accept \
         comment="allow established connections"
add chain=icmp protocol=icmp icmp-options=3:1 action=accept \
         comment="allow already established connections"
add chain=icmp protocol=icmp icmp-options=4:0 action=accept \
         comment="allow source quench"
add chain=icmp protocol=icmp icmp-options=8:0 action=accept \
         comment="allow echo request"
add chain=icmp protocol=icmp icmp-options=11:0 action=accept \
         comment="allow time exceed"
add chain=icmp protocol=icmp icmp-options=12:0 action=accept \
         comment="allow parameter bad"
add chain=icmp action=drop comment="deny all other types"
```