Connection tracking

Introduction

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.

Because of connection tracking you can use stateful firewall functionality even with stateless protocols such as UDP.

Firewall features affected by connection tracking:

- NAT
- firewall:
  - connection-bytes
  - connection-mark
  - connection-type
  - connection-state
  - connection-limit
  - connection-rate
  - layer7-protocol
  - new-connection-mark
  - tarpit

List of tracked connections can be seen in /ip firewall connection for IPv4 and /ipv6 firewall connection for IPv6.

```
[admin@3C22-atombumba] /ip firewall connection> print
Flags: S - seen-reply, A - assured
#    PR.. SRC-ADDRESS           DST-ADDRESS           TCP-STATE   TIMEOUT
0    udp  10.5.8.176:5678       255.255.255.255:5678              0s
1    udp  10.5.101.3:646        224.0.0.2:646                     5s
2    ospf 10.5.101.161          224.0.0.5                         9m58s
3    udp  10.5.8.140:5678       255.255.255.255:5678              8s
4 SA tcp  10.5.101.147:48984    10.5.101.1:8291       established 4m59s
```

```
[admin@3C22-atombumba] /ipv6 firewall connection> print
Flags: S - seen reply, A - assured
#    PRO.. SRC-ADDRESS                DST-ADDRESS                TCP-STATE
0    udp   fe80::d6ca:6dff:fe77:3698   ff02::1
1    udp   fe80::d6ca:6dff:fe98:7c28   ff02::1
2    ospf  fe80::d6ca:6dff:fe73:9822   ff02::5
```

Connection states

Based on connection table entries arrived packet can get assigned one of the connection states: **new, invalid, established, related,** or **untracked**.

There are two different methods when the packet is considered **new**. The first one is in the case of stateless connections (like UDP) when there is no connection entry in the connection table. The other one is in the case of a stateful protocol (TCP). In this case, a new packet that starts a new connection is always a TCP packet with an *SYN* flag.

If a packet is not new it can belong to either an ***established*** or ***related*** connection or not belong to any connection making it ***invalid***. A packet with an ***established*** state, as most of you already guessed, belongs to an existing connection from the connection tracking table. A ***related*** state is very similar, except that the packet belongs to a connection that is related to one of the existing connections, for example, ICMP error packets or FTP data connection packets.

Connection state **notrack** is a special case when **RAW** firewall rules are used to exclude connection from connection tracking. This rule would make all forwarded traffic bypass the connection tracking, improving packet processing speed through the device.

Any other packet is considered ***invalid*** and in most cases should be dropped.

Based on this information we can set a basic set of filter rules to speed up packet filtering and reduce the load on the CPU by accepting *established/related* packets, dropping *invalid* packets, and working on more detailed filtering only for *new* packets.

```
ip firewall filter
add chain=input connection-state=invalid action=drop comment="Drop Invalid connections"
add chain=input connection-state=established,related,untracked action=accept comment="Allow Established/Related
/Untracked connections
```

Such a rule set must not be applied on routers with asymmetric routing, because asymmetrically routed packets may be considered invalid and dropped.

FastTrack

IPv4 FastTrack is a special handler that bypasses Linux facilities allowing for faster packet forwarding. The handler is used for **TCP** and **UDP** connections marked with "`fasttrack-connection`" action. IPv4 FastTrack handler supports NAT (SNAT, DNAT, or both).

Note that not all packets of the connection can be FastTracked, so it is likely to see some packets going through a slow path even though the connection is marked for FastTrack. This is the reason why **fasttrack-connection** is usually followed by an identical "*`action=accept`*" rule.

FastTrack-ed packets are bypassing:

- firewall,
- connection tracking,
- simple queues,
- queue tree with *parent=global*,
- IP accounting,
- IPSec,
- hotspot universal client,
- VRF assignment

It is up to the administrator to make sure FastTrack does not interfere with other configuration.

Requirements

IPv4 FastTrack is active if the following conditions are met:

- no mesh, metarouter interface configuration;
- sniffer, torch, or traffic generator is not running;
- */tool mac-scan* is not actively used;
- */tool ip-scan* is not actively used;
- FastPath and Route cache is enabled under *IP/Settings*

Example

For example, for SOHO routers with factory default configuration, you could FastTrack all LAN traffic with this one rule placed at the top of the Firewall Filter. The same configuration accept rule is required:

```
/ip firewall filter add chain=forward action=fasttrack-connection connection-state=established,related
/ip firewall filter add chain=forward action=accept connection-state=established,related
```

- Connection is FastTracked until the connection is closed, timed-out, or router is rebooted.
- Dummy rules will disappear only after FastTrack firewall rules will be deleted/disabled and the router rebooted.
- While FastPath and FastTrack both are enabled on the device only one can be active at a time.

Queues (except Queue Trees parented to interfaces), firewall filter, and mangle rules will not be applied for FastTracked traffic.

Connection tracking settings

Connection tracking settings are managed from `/ip firewall connection tracking` menu.

Properties

Property	Description
enabled (*yes* / *no* / *auto*; Default: **auto**)	Allows to disable or enable connection tracking. With disabled connection tracking firewall features listed above will stop working. If set to "auto" connection tracking is disabled until at least one firewall rule is added.
liberal-tcp-tracking (*yes* / *no;* Default: **no**)	Enables or disables liberal TCP connection tracking by toggling the kernel parameter `nf_conntrack_tcp_be_liberal`. When set to **yes**, the system mark only out of window RST segments as INVALID. Enabling this setting may allow malformed packets that would otherwise be considered `invalid` by the firewall's `connection-state` matcher. This can increase exposure to certain evasion techniques. This property should be enabled only when troubleshooting or working around known issues.
loose-tcp-tracking (*yes*; Default: **yes**)	• In case loose-tcp-tracking=yes, the 2nd part (SYN,ACK) and 3rd part (ACK) of the handshake without having seen the first initial SYN will be considered ESTABLISHED • In case loose-tcp-tracking=no, the 2nd part (SYN,ACK) and 3rd part (ACK) without having seen the first initial SYN will be considered INVALID
tcp-syn-sent-timeout (*time*; Default: **5s**)	TCP SYN timeout.
tcp-syn-received-timeout (*time*; Default: **5s**)	TCP SYN timeout.
tcp-established-timeout (*time*; Default: **1d**)	Time after which established TCP connection times out.
tcp-fin-wait-timeout (*time*; Default: **10s**)	

tcp-close-wait-timeout (*time*; Default: **10s**)	
tcp-last-ack-timeout (*time*; Default: **10s**)	
tcp-time-wait-timeout (*time*; Default: **10s**)	
tcp-close-timeout (*time*; Default: **10s**)	
udp-timeout (*time*; Default: **30s**)	Specifies the timeout for UDP connections that have seen packets in one direction
udp-stream-timeout (*time*; Default: **3m**)	Specifies the timeout of UDP connections that have seen packets in both directions
icmp-timeout (*time*; Default: **10s**)	ICMP connection timeout
generic-timeout (*time*; Default: **10m**)	Timeout for all other connection entries

Read-only properties

Property	Description
max-entries (*integer*)	Max amount of entries that the connection tracking table can hold. This value depends on the installed amount of RAM. Note that the system does not create a maximum-size connection tracking table when it starts, it may increase if the situation demands it and the system still has free RAM, but the size will not exceed 1048576
total-entries (*integer*)	Amount of connections that the connection table currently holds

Connection List

List of tracked connections ban be seen in /ip firewall connection for ipv4 and /ipv6 firewall connection for IPv6.

Properties

All properties in the connection list are read-only

Property	Description
assured (*yes* | *no*)	Indicates that this connection is assured and that it will not be erased if the maximum possible tracked connection count is reached.
confirmed (*yes* | *no*)	Connection is confirmed and a packet is sent out from the device
connection-mark (*string*)	Connection mark that was set by the mangle rule.
connection-type (*pptp* | *ftp*)	Type of connection, the property is empty if connection tracking is unable to determine a predefined connection type.

dst-address (*ip*)	Destination address.
dst-port (*integer*)	Destination port.
dstnat (*yes* | *no*)	A connection has gone through DST-NAT (for example, port forwarding).
dying (*yes* | *no*)	The connection is dying due to a connection timeout.
expected (*yes* | *no*)	Connection is set up using connection helpers (pre-defined service rules).
fasttrack (*yes* | *no*)	Whether the connection is FastTracked.
gre-key (*integer*)	Contents of the GRE Key field.
gre-protocol (*string*)	Protocol of the encapsulated payload.
gre-version (*string*)	A version of the GRE protocol was used in the connection.
connection-mark (*string*)	Connection mark assigned for the connection from firewall.
hw-offload (*yes* | *no*)	Hardware offloaded connection.
icmp-code (*string*)	ICMP Code Field
icmp-id (*integer*)	Contains the ICMP ID
icmp-type (*integer*)	ICMP Type Number
orig-bytes (*integer*)	Amount of bytes sent out from the source address using the specific connection.
orig-fasttrack-bytes (*integer*)	Amount of FastTracked bytes sent out from the source address using the specific connection.
orig-fasttrack-packets (*integer*)	Amount of FastTracked packets sent out from the source address using the specific connection.
orig-packets (*integer*)	Amount of packets sent out from the source address using the specific connection.
orig-rate (*integer*)	The data rate at which packets are sent out from the source address using the specific connection.
protocol (*string*)	IP protocol type
repl-bytes (*integer*)	Amount of bytes received from the destination address using the specific connection.
repl-fasttrack-bytes (*string*)	Amount of FastTracked bytes received from the destination address using the specific connection.
repl-fasttrack-packets (*integer*)	Amount of FastTracked packets received from the destination address using the specific connection.
repl-packets (*integer*)	Amount of packets received from the destination address using the specific connection.
repl-rate (*string*)	The data rate at which packets are received from the destination address using the specific connection.
reply-dst-address (*ip*)	Destination address expected of return packets.
reply-dst-port (*integer*)	Destination port expected of return packets.
reply-src-address (*ip*)	Source address expected of return packets.
reply-src-port (*integer*)	Source port expected of return packets.
seen-reply (*yes* | *no*)	The destination address has replied to the source address.
src-address (*ip*)	The source address.
src-port (*integer*)	The source port.
srcnat (*yes* | *no*)	Connection is going through SRC-NAT, including packets that were masqueraded through NAT.

tcp-state (*string*)	The current state of TCP connection : • "established" • "time-wait" • "close" • "syn-sent" • "syn-recv" • "fin-wait" • "close-wait" • "last-ack" • "listen"
timeout (*time*)	Time after connection will be removed from the connection list.
uses-helper (*yes* | *no*)	"IP/Firewall/Service Port" helper has been applied to the particular connection.