Rocky Enterprise Linux 9.2 Manual Pages on command 'tc-ct.8'

$ man tc-ct.8

ct action in tc(8)   Linux   ct action in tc(8)

NAME

  ct - tc connection tracking action

SYNOPSIS

  tc ... action ct commit [force] [zone ZONE] [mark MASKED_MARK] [label MASKED_LABEL

    ] [nat NAT_SPEC]

  tc ... action ct [nat] [zone ZONE]

  tc ... action ct clear

DESCRIPTION

  The ct action is a tc action for sending packets and interacting with the netfilter con?

  ntrack module.

  It can (as shown in the synopsis, in order):

  Send the packet to conntrack, and commit the connection, while configuring a 32bit mark,

  128bit label, and src/dst nat.

  Send the packet to conntrack, which will mark the packet with the connection's state and

  configured metadata (mark/label), and execute previous configured nat.

  Clear the packet's of previous connection tracking state.

OPTIONS

  zone ZONE

    Specify a conntrack zone number on which to send the packet to conntrack.

  mark MASKED_MARK

    Specify a masked 32bit mark to set for the connection (only valid with commit).

  label MASKED_LABEL

Specify a masked 128bit label to set for the connection (only valid with commit).

nat NAT_SPEC

    Where NAT_SPEC := {src|dst} addr addr1[-addr2] [port port1[-port2]]

    Specify src/dst and range of nat to configure for the connection (only valid with commit).

    src/dst - configure src or dst nat

    addr1/addr2 - IPv4/IPv6 addresses

    port1/port2 - Port numbers

nat   Restore any previous configured nat.

clear Remove any conntrack state and metadata (mark/label) from the packet (must only op‐ tion specified).

force Forces conntrack direction for a previously committed connections, so that current direction will become the original direction (only valid with commit).

EXAMPLES

Example showing natted firewall in conntrack zone 2, and conntrack mark usage:

#Add ingress qdisc on eth0 and eth1 interfaces

$ tc qdisc add dev eth0 handle ingress

$ tc qdisc add dev eth1 handle ingress

#Setup filters on eth0, allowing opening new connections in zone 2, and doing src nat + mark for each new connection

$ tc filter add dev eth0 ingress prio 1 chain 0 proto ip flower ip_proto tcp ct_state -trk \

action ct zone 2 pipe action goto chain 2

$ tc filter add dev eth0 ingress prio 1 chain 2 proto ip flower ct_state +trk+new \

action ct zone 2 commit mark 0xbb nat src addr 5.5.5.7 pipe action mirred egress redirect dev eth1

$ tc filter add dev eth0 ingress prio 1 chain 2 proto ip flower ct_zone 2 ct_mark 0xbb ct_state +trk+est \

action ct nat pipe action mirred egress redirect dev eth1

#Setup filters on eth1, allowing only established connections of zone 2 through, and reverse nat (dst nat in this case)

$ tc filter add dev eth1 ingress prio 1 chain 0 proto ip flower ip_proto tcp ct_state -trk \

action ct zone 2 pipe action goto chain 1

$ tc filter add dev eth1 ingress prio 1 chain 1 proto ip flower ct_zone 2 ct_mark 0xbb ct_state +trk+est \

action ct nat pipe action mirred egress redirect dev eth0

SEE ALSO

tc(8), tc-flower(8) tc-mirred(8)

AUTHORS

Paul Blakey <paulb@mellanox.com>

Marcelo Ricardo Leitner <marcelo.leitner@gmail.com>

Yossi Kuperman <yossiku@mellanox.com>

iproute2   14 May 2020   ct action in tc(8)