

Dissecting an Untracked Policy

Untracked policy is very simple, yet can be deceiving. Let us start with a clean slate and trace a simple untracked policy. The files referred in this example are in this [calico policies by example repo](#). We will use ncat (nc) command for client/server connections.

Objective is to create an untracked policy to permit traffic into port 7777 of the host. We will verify that conntrack is not created when untracked policy is in use.

First, start with a clean slate. As you can see below, nc is listening on port 7777 on left, and nc client can connect (on the right). Also the conntrack table creates an entry (left bottom).

```
...term* 1#2#3# — ssh -i ~/.ssh/singleNodeAwsK8s.pem ubuntu@ec2-54-167-164-217.compute-1.amazonaws.com ~ — -bash
[centos@ip-172-31-8-215 hep]$ nc -k -l 172.31.8.215 7777
Testing without policy
Testing return

cloudterm [/home/ubuntu] >> hostname -I
172.31.92.5 172.17.0.1
cloudterm [/home/ubuntu] >> nc 172.31.8.215 7777
Testing without policy
Testing return

[centos@ip-172-31-8-215 ~]$ sudo conntrack -L | grep 7777
tcp        6 86347 ESTABLISHED src=172.31.92.5 dst=172.31.8.215 sport=57160
dport=7777 packets=5 bytes=291 src=172.31.8.215 dst=172.31.92.5 sport=7777
dport=57160 packets=4 bytes=232 [ASSURED] mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 396 flow entries have been shown.
[centos@ip-172-31-8-215 ~]$ _
```

It takes about 120 seconds (default) for conntrack entry to expire after the connection has been terminated. As part of the next step, let's create a calico host endpoint. **Note: Creating a host endpoint (if you follow the example below) will make your host network blind, except for the fail-safe ports in Calico. So do not attempt to try this exercise on a production system.**

Apply the configuration (kubectl apply -f hep0.yaml) [hep0.yaml](#). Note that you will need to customize the configuration per your host specification. Now let us see the behavior.

```
...udterm* 1#3# — ssh -i ~/.ssh/singleNodeAwsK8s.pem ubuntu@ec2-54-167-164-217.compute-1.amazonaws.com ~ — -bash
[centos@ip-172-31-8-215 hep]$ nc -k -l 172.31.8.215 7777
cloudterm [/home/ubuntu] >> nc 172.31.8.215 7777
Testing connection with hostendpoint

[centos@ip-172-31-8-215 ~]$ sudo conntrack -L | grep 7777
conntrack v1.4.4 (conntrack-tools): 378 flow entries have been shown.
[centos@ip-172-31-8-215 ~]$ _
```

So there's no connection! This is expected, as the hostendpoint creates a default-deny policy for all traffic into the host. Let us whitelist all traffic (apply manifest [whitelist.yaml](#)) into the host. This will allow us to move to an untracked policy next.

```
[centos@ip-172-31-8-215 hep]$ nc -k -l 172.31.8.215 7777
Testing connection with hostendpoint and whitelist
cloudterm [/home/ubuntu] >> nc 172.31.8.215 7777
Testing connection with hostendpoint and whitelist

[centos@ip-172-31-8-215 ~]$ sudo conntrack -L | grep 7777
conntrack v1.4.4 (conntrack-tools): 378 flow entries have been shown.
[centos@ip-172-31-8-215 ~]$ sudo conntrack -L | grep 7777
tcp      6 86393 ESTABLISHED src=172.31.92.5 dst=172.31.8.215 sport=57168
dport=7777 packets=3 bytes=215 src=172.31.8.215 dst=172.31.92.5 sport=7777
dport=57168 packets=2 bytes=112 [ASSURED] mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 380 flow entries have been shown.
[centos@ip-172-31-8-215 ~]$ _
```

Now we are ready to create an untracked policy on host endpoint. Apply the policy [untrack-singleside.yaml](#). It basically applies the untracked policy in a single direction (permit into port 7777 of the host).

```

[centos@ip-172-31-8-215 hep]$ nc -k -l 172.31.8.215 7777
Testing connection with hostendpoint and whitelist
^C
[centos@ip-172-31-8-215 hep]$ ka notrack-single-side.yaml
globalnetworkpolicy.projectcalico.org/default.donottrackforport7777 created
[centos@ip-172-31-8-215 hep]$ nc -k -l 172.31.8.215 7777
^C
[centos@ip-172-31-8-215 hep]$ nc -k -l 172.31.8.215 7777

cloudterm [/home/ubuntu] >> nc 172.31.8.215 7777
Testing with untracked policy ONLY on ingress
cloudterm [/home/ubuntu] >> Testing with untracked policy ONLY on ingress
Testing: command not found
cloudterm [/home/ubuntu] >>

[centos@ip-172-31-8-215 ~]$ sudo conntrack -L | grep 7777
tcp        6 3491 CLOSE_WAIT src=172.31.92.5 dst=172.31.8.215 sport=57168 dport=7777 packets=4 bytes=267 src=172.31.8.215 dst=172.31.92.5 sport=7777 dport=57168 packets=3 bytes=164 [ASSURED] mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 384 flow entries have been shown.
[centos@ip-172-31-8-215 ~]$ _

```

So it did not work! And there's some unexpected behavior. Conntrack table shows an entry stuck in CLOSE_WAIT mode. What actually happened here? Looking at the packet processing chain diagram above, the incoming flow bypassed the conntrack (because of untracked policy), but the return flow (in OUTPUT chain) hit the conntrack. And conntrack did not find any matching connection for the response packet. We do not want this kind of behavior. Hence it is important to test and deploy untracked policies properly.

Let us now deploy the untracked policy on both sides (apply [file](#)). This applies untracked allow policy to and from port 7777 on the host.

```

[centos@ip-172-31-8-215 hep]$ nc -k -l 172.31.8.215 7777
Testing with untracked policy on BOTH SIDES!!
That works!!

cloudterm [/home/ubuntu] >> nc 172.31.8.215 7777
Testing with untracked policy on BOTH SIDES!!
That works!!

[centos@ip-172-31-8-215 ~]$ sudo conntrack -L | grep 7777
tcp        6 2877 CLOSE_WAIT src=172.31.92.5 dst=172.31.8.215 sport=57168 dport=7777 packets=4 bytes=267 src=172.31.8.215 dst=172.31.92.5 sport=7777 dport=57168 packets=3 bytes=164 [ASSURED] mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 390 flow entries have been shown.
[centos@ip-172-31-8-215 ~]$ _

```

As you can see above, the traffic went through just fine. And no conntrack entry was created.

Finally, let us see how to navigate the untracked policy in the iptables itself. We know that it's hooked into raw table in PREROUTING chain. So start traversing until you hit the policy itself. The following diagram shows the snapshot of starting from the root upto the policy.

```
[centos@ip-172-31-8-215 ~]$ sudo iptables -L PREROUTING -t raw_ [4/1450]
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
cali-PREROUTING all -- anywhere anywhere /* cali:6gwbT8clXdHdC1b1 */
[centos@ip-172-31-8-215 ~]$ sudo iptables -L cali-PREROUTING -t raw
Chain cali-PREROUTING (1 references)
target prot opt source destination
MARK all -- anywhere anywhere /* cali:DQ6LK2guiBRMyrLK */ MARK and 0xffe4ffff
MARK all -- anywhere anywhere /* cali:o0awFTLnSamXVVNW */ MARK or 0x80000
cali-from-host-endpoint all -- anywhere anywhere /* cali:rqYLrN00NR-C00Ss */ mark match 0x0/0x80000
ACCEPT all -- anywhere anywhere /* cali:9iRqCz12lyucnEkb */ mark match 0x10000/0x10000
[centos@ip-172-31-8-215 ~]$ sudo iptables -L cali-from-host-endpoint -t raw
Chain cali-from-host-endpoint (1 references)
target prot opt source destination
cali-fh-eth0 all -- anywhere anywhere [goto] /* cali:HIN_eur26mTLJ97D */
[centos@ip-172-31-8-215 ~]$ sudo iptables -L cali-fh-eth0 -t raw
Chain cali-fh-eth0 (1 references)
target prot opt source destination
cali-failsafe-in all -- anywhere anywhere /* cali:RKoA06cNXCcLf1Xr */
MARK all -- anywhere anywhere /* cali:VqqtlnzJsuQQ16hW */ MARK and 0xffffaffff
MARK all -- anywhere anywhere /* cali:DefjwbMffQ6eziu7 */ /* Start of tier default */ MARK and 0xffffdffff
cali-pi-_lSucmfdQFnsmd0M97oC all -- anywhere anywhere /* cali:kQdSfSYRWQY8YRL */ mark match 0x0/0x20000
CT all -- anywhere anywhere /* cali:0puPMJrzfW2_uXhv */ mark match 0x10000/0x10000 NOTRACK
RETURN all -- anywhere anywhere /* cali:-IKn0fIrWa5JZQJF */ /* Return if policy accepted */ mark match 0x10000/0x10000
[centos@ip-172-31-8-215 ~]$ sudo iptables-save | grep "-A cali-pi-_lSucmfdQFnsmd0M97oC"
grep: cali-pi-_lSucmfdQFnsmd0M97oC: invalid context length argument
[centos@ip-172-31-8-215 ~]$ sudo iptables-save | grep "A cali-pi-_lSucmfdQFnsmd0M97oC"
-A cali-pi-_lSucmfdQFnsmd0M97oC -p tcp -m comment --comment "cali:SSVCN0dc47bS3Dft" -m multiport --dports 7777 -j MARK --set-xmark 0x10000/$x10000
-A cali-pi-_lSucmfdQFnsmd0M97oC -m comment --comment "cali:ayxVm0xaYYvXJeTv" -m mark --mark 0x10000/0x10000 -j RETURN
-A cali-pi-_lSucmfdQFnsmd0M97oC -p tcp -m comment --comment "cali:SSVCN0dc47bS3Dft" -m multiport --dports 7777 -j MARK --set-xmark 0x10000/$x10000
-A cali-pi-_lSucmfdQFnsmd0M97oC -m comment --comment "cali:ayxVm0xaYYvXJeTv" -m mark --mark 0x10000/0x10000 -j RETURN
```

Untracked policies can be very powerful, if used properly.

This completes the review of untracked policy in calico.