# 672.https://stackoverflow.com/questions/64801304/how-right-to-make-second-input-chain-in-other-table-nftables

**T:**How right to make second input chain in other table nftables?

**Q:**There is my test nft ruleset , and all works except table inet test but table f2b-table is absolytly similar (except drop vs accept) and it works fine :  
  
table inet f2b-table { set addr-set-sshd { type ipv4\_addr elements = { 0.0.0.0 } } chain input { type filter hook input priority filter - 1; policy accept; tcp dport { 222 } ip saddr @addr-set-sshd drop }}table inet default { set full\_op\_port { type inet\_service elements = { 222 } } set allowed\_ips { type ipv4\_addr elements = { 0.0.0.0 } } chain INPUT { type filter hook input priority filter; policy drop; ct state invalid drop ct state { established, related } accept iif "lo" accept tcp dport @full\_op\_port accept ip saddr @allowed\_ips accept ip protocol icmp accept counter packets 17 bytes 884 } chain FORWARD { type filter hook forward priority filter; policy drop; } chain OUTPUT { type filter hook output priority filter; policy accept; }}table ip test { chain PREROUTING { type nat hook prerouting priority filter; policy accept; } chain POSTROUTING { type nat hook postrouting priority srcnat; policy accept; } chain FORWARD { type filter hook forward priority filter; policy drop; }}table inet test { set op\_port { type inet\_service elements = { 8888 } } chain INPUT { type filter hook input priority filter - 2; policy accept; tcp dport @op\_port accept }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I see packages in tcpdump, i see packages when i makes count in table table inet test but packages don't be accepted. What do i make wrong?

2 **Answer**

**A1:**I am adding another answer with examples here to clarify the unintended consequences of mixing policies with multiple base-chains of the same family, type and hook. Although priority can be made the same on these, it never should. Lower priority numbers mean a higher priority and will be run first. Applying drop policies incorrectly can cause unintended consequences for traffic you intend to accept.  
  
As to the effect of mixing the hybrid family inet with ip and ip6, I won't even begin to pontificate except to say it's probably a bad idea.  
  
WARNING: The examples horribly break ipv4 traffic and were performed on a VM - buyer beware !  
  
An example of a bad drop policy:  
  
table inet filter { chain input1 { type filter hook input priority filter + 1; policy drop; tcp dport 80 log prefix "input1\_" # SEEN } # input2 chain not evaluated as there is no traffic left after input1 chain input2 { type filter hook input priority filter + 2; policy accept; tcp dport 80 accept tcp dport 80 log prefix "input2\_" }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
An example of a ok drop policy:  
  
table inet filter { chain input1 { type filter hook input priority filter + 1; policy accept; tcp dport 80 log prefix "input1\_" # SEEN } chain input2 { type filter hook input priority filter + 2; policy drop; tcp dport 80 accept tcp dport 80 log prefix "input2\_" # NOT SEEN due previous accept }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
An example of a bad accept policy:  
  
table inet filter { chain input1 { type filter hook input priority filter + 1; policy accept; tcp dport 80 accept tcp dport 80 log prefix "input1\_" # NOT SEEN due to previous accept } chain input2 { type filter hook input priority filter + 2; policy drop; tcp dport 80 log prefix "input2\_" # SEEN - chain evaluates # all traffic dropped here by policy including accepted input1 traffic }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
An example of an okay accept policy:  
  
table inet filter { chain input1 { type filter hook input priority filter + 1; policy accept; tcp dport 80 log prefix "input1\_" # SEEN } chain input2 { type filter hook input priority filter + 2; policy drop; tcp dport 80 accept tcp dport 80 log prefix "input2\_" # NOT SEEN due to previous accept }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
As stated in the man page for nft, a drop by rule or policy drops immediately without further processing of lower priority base chains. accept does not. it short circuits remaining rules at the current priority and hands off to the next lower priority but here it is still subject to being dropped if explicitly dropped by rule or implicitly dropped by policy if there is no rule to accept.  
  
Perhaps the easiest way to approach it is use a single base chain and jump/goto non-base chains, effectively the way iptables worked.

**A2:**The answer comes from A.B where he stated:  
  
just to clarify that a packet can be accepted (or not) multiple times in the same hook:  
  
and posted from the nft man page  
  
accept  
  
Terminate ruleset evaluation and accept the packet. The packet can still be droppedlater by another hook, for instance accept in the forward hook still allows to drop thepacket later in the postrouting hook, or another forward base chain that has a higherpriority number and is evaluated afterwards in the processing pipeline.  
  
Your default table base chain priority 0 will evaluate after your test table base chain priority -2 and because it has a drop policy and the packet is not matched there, it will be dropped.  
  
The man page is confusing on this. Where it says for the allow verdict "terminate ruleset evaluation and accept the packet", it really only terminates the view of the ruleset at the given base chain priority. Other base chains of the same type, hook and family with lower priority due to the higher priority number will still run after and can override by rule or policy. This is unlike the drop verdict where all is stopped and the packet is summarily dropped. You can see this in action using logging:  
  
nft flush rulesetnft create table ip table1nft add chain ip table1 input1 { type filter hook input priority filter\; policy drop\; }nft add rule ip table1 input1 tcp dport != 8888 acceptnft add rule ip table1 input1 tcp dport 8888 log prefix \"TABLE1\_INPUT1 DROPPING \" level infonft create table ip table2nft add chain ip table2 input2 { type filter hook input priority filter - 1\; policy accept\; }nft add rule ip table2 input2 tcp dport != 8888 acceptnft add rule ip table2 input2 tcp dport 8888 log prefix \"TABLE2\_INPUT2 BEFORE \" level infonft add rule ip table2 input2 tcp dport 8888 acceptnft add rule ip table2 input2 tcp dport 8888 log prefix \"TABLE2\_INPUT2 AFTER \" level info  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**I don't undestend, it makes tables in nftables almost usless, no? You can use them only for drop?

**C2:**No, it allows you to organize your rules more logically. You also have non-base chains, aka user chains from iptables. You can jump/goto from a base chain to a non-base chain but not vice-versa. There is no default policy on these but you can set a final drop or accept rule. Maybe rethink how you are doing it?