IPtables commands cheat sheet

Overview

IPtables is one of the most well know firewalls(and quite effective as well) which also comes on many Linux distro's pre-installed by default. The purpose of this post is to act as an Iptables commands cheat sheet, considering how Iptables is powerfull, it's also has a numerous commands for many networking scenarios. Iptables uses various different table rules with multiple chains to block and allow traffic:

- **FILTER:** this is the default table that is meant to filter the traffic rules
    - **INPUT**: Under input chain we define the rules and behaviors to control incoming connections.
    - **OUTPUT**: Under the output chain we control the outgoing connections
    - **FORWARD**: as the name suggests, WE use forward chain to specify incoming connections that we are going to redirect right away to another route, address or port(forward it). Forwarding is commonly used together with NAT.
- **NAT** – network address translation table that is used for mapping multiple local traffic resources to the outgoing connections to establish a new route. NAT table has the following chain rules included:
    - **PREROUTING** – is used to control/modify a packet as soon it has arrived(incoming connections)
    - **OUTPUT** – is used for modifying locally generated packets
    - **POSTROUTING** – is used to control/modify a packet as soon as it's about to leave(outgoing connections)
- **MANGLE** – mangle table is used for packet modification or packet altering. Mangle table has 5 chain rules:
    - **PREROUTING** – for altering incoming connections
    - **OUTPUT** – for altering locally generated packets
    - **INPUT** – for incoming packets
    - **POSTROUTING** – for altering packets as they are about to go out
    - **FORWARD** – for packets routed through the box(or needs to be forwarded to a new connection/route)

IPtables commands

How to display firewall rules

– Display all rules

```
# iptables -L
```

```
root@markontech:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Example of a default iptables chain rules table

– Display all rules with line numbers

```
# iptables -L --line-numbers
```

– Display all rules with verbose output of the active packets

```
# iptables -n -L -v
```

```
root@markontech:~# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER-USER  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DOCKER-ISOLATION-STAGE-1  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
    0     0 DOCKER     all  --  *      docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  docker0 docker0  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Example of a verbose output

-Display rules from a NAT chain

```
# iptables -t nat -L
```

These output/display options can be combined with other display commands as well, like:

```
# iptables -t nat -L --line-numbers
```

– Display rules from a specific chain rule

```
# iptables -L INPUT
```

– Display rules but with chain specifications

```
# iptables -S INPUT
```

-Display rules from a chain and with active packets

```
# iptables -S INPUT -v
```

How to delete and add rules

– Delete a rule by a line number

```
# iptables -D INPUT 10
```

– Delete a rule by a specification

```
# iptables -D INPUT -m conntrack --ctstate
```

– Flush all chain(delete all chain rules)

```
# iptables -F
```

– Flush a single chain

```
# iptables -F INPUT
# Iptables -t nat -F
# Iptables -t mangle -F
```

– Add a new chain

```
# iptables -N custom-filter
```

– Add a new rule

```
# iptables -I INPUT -s 123.123.123.133 -j DROP
# iptables -A INPUT -p tcp --dport 22 -j REJECT
```

Note regarding adding the rules – as you can see here we have two different rules with different options. Note the first defined options right after the iptables: -I and -A.
These options tell you where are rules going to be placed in the table, at the beginning (-I option) of the chain rule or at the bottom (-A).
Iptables reads the rules by going from the first rule at the top of the chain then goes down to the bottom and applies them in that order.

Examples and most common used IPtables commands

Block traffic

– Block an IP address to have any access to incoming traffic

```
# iptables -A INPUT -s 192.168.100.1 -j DROP
```

– Block a specific IP subnet from incoming

```
# iptables -A INPUT -s 192.168.1.100/24 -j DROP
```

– Block an IP address and reject all packets

```
# iptables -A INPUT -s 192.168.1.100 -j REJECT
```

– Block an incomming traffic from an IP address to a specific network interface

```
# iptables -A INPUT -i eth0 -s 192.168.1.102 -j DROP
```

– Block only TCP traffic from a speciffic IP address or IP range

```
# iptables -A INPUT -p tcp -s 192.168.1.100 -j DROP
# iptables -A INPUT -p tcp -s 192.168.1.100/24 -j DROP
```

– Block a speciffic port to receive any traffic

```
# iptables -A INPUT -p tcp --dport xxx -j DROP
# iptables -A INPUT -p tcp --dport 22 -j DROP
```

– Drop all invalid network packets on incoming

```
# iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

Allow traffic or open ports with IPtables

– Allow traffic(incoming and outgoing) on SSH

```
# iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -
j ACCEPT
```

Note: to block traffic on these ports, change ACCEPT option to DROP

– Allow a specific IP or network range on SSH incoming(remove CIDR to set just an IP)

```
# iptables -A INPUT -p tcp -s 192.168.1.100/24 --dport 22 -m conntrack --
ctstate NEW,ESTABLISHED -j ACCEPT
```

Note: Same configuration for network range can be applied on other incoming rules listed bellow

Note: to block traffic on these ports, change ACCEPT option to DROP

– Allow traffic on HTTP and HTTPS(incoming and outgoing)

```
# iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 80 -m conntrack --ctstate ESTABLISHED -
j ACCEPT
# iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED
-j ACCEPT
```

– Multiport config

```
# iptables -A INPUT -p tcp -m multiport --dports 80,443 -m conntrack --
ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m conntrack --
ctstate ESTABLISHED -j ACCEPT
```

Note: to block traffic on these ports, change ACCEPT option to DROP

– Allow traffic on MySQL

```
# iptables -A INPUT -p tcp -s 192.168.1.100/24 --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

– Allow MySQL traffic to a specific network interface

```
# iptables -A INPUT -i eth1 -p tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -o eth1 -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

– Allow PostgreSQL traffic

```
# iptables -A INPUT -p tcp -s 192.168.1.0/24 --dport 5432 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

– Allow PostgreSQL traffic to a specific network interface

```
# iptables -A INPUT -i eth1 -p tcp --dport 5432 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -o eth1 -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

– Allow incoming on SMTP/IMAP/IMAPS/POP3/POP3S/

```
SMPT
# iptables -A INPUT -p tcp --dport 25 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 25 -m conntrack --ctstate ESTABLISHED -j ACCEPT

IMAP
# iptables -A INPUT -p tcp --dport 143 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 143 -m conntrack --ctstate ESTABLISHED -j ACCEPT

IMAPS
```

```
# iptables -A INPUT -p tcp --dport 993 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 993 -m conntrack --ctstate ESTABLISHED
-j ACCEPT

POP3
# iptables -A INPUT -p tcp --dport 110 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 110 -m conntrack --ctstate ESTABLISHED
-j ACCEPT

POP3S
# iptables -A INPUT -p tcp --dport 995 -m conntrack --ctstate
NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 995 -m conntrack --ctstate ESTABLISHED
-j ACCEPT
```

– How to configure port forwarding in IPtables

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j REDIRECT --to-
port 2525
```

This command for example will redirect all incoming traffic on **eth0** port from port 25 to the port 2525

This parameters here are also used as an example. What will be achieved here is – on port 80, all incoming will be limited/reduced to 100 established connections per minute and set a limitr-burst of 200 matching packets.

– Block incoming ping requests

```
# iptables -A INPUT -p icmp -i eth0 -j DROP
```

– Block or allow access from a specific mac address

```
# iptables -A INPUT -m mac --mac-source 00:00:00:00:00:00 -j DROP
# iptables -A INPUT -p tcp --destination-port 22 -m mac --mac-source
00:00:00:00:00:00 -j ACCEPT
```

– Alliow loopback access

```
# iptables -A INPUT -i lo -j ACCEPT
```

```
# iptables -A OUTPUT -o lo -j ACCEPT
```

– Limit the number of concurrent connections per IP address

```
# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above
3 -j REJECT
```

The above command will limit the number of connections to port 22(ssh) and not allow more than connections per client. The port number of course can be changed.

– **Log dropped packets**

```
# iptables -A INPUT -i eth0 -j LOG --log-prefix "Dropped packets:"
```

You can change the the –log-prefix to your choosing. The logs can be search with the following command:

```
# grep "IPtables dropped packets:" /var/log/messages
```

– Log dropped packet coming from a specific network range

```
# iptables -A INPUT -i eth1 -s 10.0.0.0/8 -j LOG --log-prefix "IP_SPOOF A:
"
```

– How to search entries/rules within the IPtables with grep

```
# iptables -L INPUT -v -n | grep 192.168.0.100
```

```
root@markontech:~# iptables -L INPUT -v -n | grep 123.123.123.123
    0     0 DROP       all  --  *      *       123.123.123.123      0.0.0.0/0
root@markontech:~#
```

Example output of a rule search with grep

Iptables commands to prevent more advanced cyber attacks

– How to block network flooding on http port

```
# iptables -A INPUT -p tcp --dport 80 -m limit --limit 100/minute --limit-
burst 200 -j ACCEPT
```

– Configure port scanning protection

```
# iptables -N port-scanning
# iptables -A port-scanning -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit
--limit 1/s --limit-burst 2 -j RETURN
# iptables -A port-scanning -j DROP
```

– Brute force protection for SSH

```
# iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent
--set
# iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent
--update --seconds 60 --hitcount 10 -j DROP
```

– Protection against SYN flood attacks

```
# iptables -N syn_flood
# iptables -A INPUT -p tcp --syn -j syn_flood
# iptables -A syn_flood -m limit --limit 1/s --limit-burst 3 -j RETURN
# iptables -A syn_flood -j DROPiptables -A INPUT -p icmp -m limit --limit
1/s --limit-burst 1 -j ACCEPT
# iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j LOG --
log-prefix PING-DROP:
# iptables -A INPUT -p icmp -j DROPiptables -A OUTPUT -p icmp -j ACCEPT
```

– Mitigating SYN flood attacks with SYNPROXY

```
# iptables -t raw -A PREROUTING -p tcp -m tcp --syn -j CT --notrack
# iptables -A INPUT -p tcp -m tcp -m conntrack --ctstate INVALID,UNTRACKED
-j SYNPROXY --sack-perm --timestamp --wscale 7 --mss 1460
# iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

– Block packets on incoming that are not SYN

```
# iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

– Force fragments check on all packets on incoming

```
# iptables -A INPUT -f -j DROP
```

– Block all packets on incoming to prevent XMAS packet attack

```
# iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
```

– Block all NULL packets on incoming

```
# iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
```

– Intercept and drop all packets with bogus TCP flags

```
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags
FIN,SYN,RST,PSH,ACK,URG NONE -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN -j
DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST -j
DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j
DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,FIN FIN -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL ALL -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL FIN,PSH,URG -j
DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL SYN,FIN,PSH,URG -
j DROP
# iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL
SYN,RST,ACK,FIN,URG -j DROP
```

– Save IPtables rules to a file and restore them

```
# iptables-save > ~/iptables.rules
# iptables-restore < ~/iptables.rules
```

– Save IPtables rules permanently with netfilter

Note: requires iptables plugin **“iptables-persistent”** to be installed. Can install it with **“apt install iptables-persistent”** on Debian/Ubuntu distros.

```
# netfilter-persistent save
```

Example output of some rules applied on INPUT from this post