

It's Never the Firewall: Diagnosing Linux Firewall Issues

...

Sam Stelfox

Who Am I?

Software Engineer

... turned Systems Administrator

... then Network Administrator

... and Penetration Tester

Now Security Software Engineer

... with some DevOps thrown in

Official title “Engineer”



Linux Firewall Underpinnings

- ipfwadm (Linux kernel ≤ 2.0)
- ipchains (Linux kernel 2.2 - 2.4)
- iptables/netfilter (Linux kernel 2.4) - Introduced 1998
- nftables (Linux kernel 3.13) - Introduced 2014
- ... XDP / eBPF (Linux kernel 4.8) - ~2016... kind of

Common Management Tools

- firewalld (RH Default - Fedora/RHEL/CentOS)
- ufw - Uncomplicated firewall (Ubuntu)
- fail2ban
- Shorewall
- fwsnort
- FireHOL

isn't working! Must be the firewall!

It's Never the Firewall

...probably...

If You're in the Cloud...

... the Linux firewall is likely not in use*

- AWS: Disabled
- GCP: Disabled
- Digital Ocean: Disabled
- Vultr: Disabled
- Pre-made Cloud-Init Linux Image: Likely Disabled

* Dependent on the image you choose,
and can be enabled


```
Terminal
File Edit View Search Terminal Help
[user@sample-host] ~ $ sudo iptables -nL
Chain INPUT (policy ACCEPT)
target      prot opt source                destination

Chain FORWARD (policy ACCEPT)
target      prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target      prot opt source                destination
[user@sample-host] ~ $ █
```

Done. Not the Firewall. Check DNS.

Initial Diagnostics

- Is the service actually unavailable?
- Do the DNS records match the IP addresses I expect?
- Has there been enough time for the old DNS records to expire?
- Does the system have the IP I expect?
- Can I ping the IP?
- Is the service running?
- Can I access the service from the host?
- Is the service bound to the correct interface?

... Alright, maybe it's *a* firewall ...

Universal Firewall Diagnostics

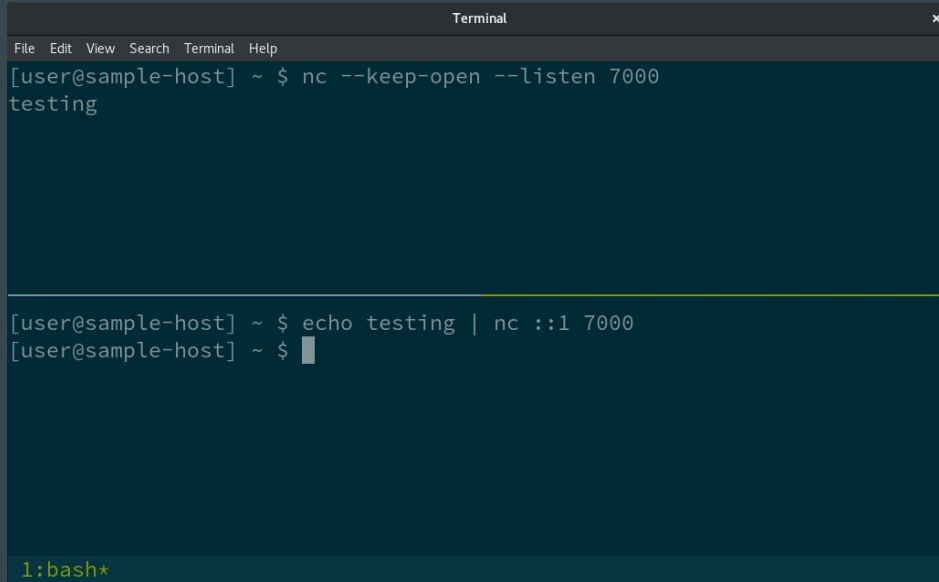
- Rule out the service as an issue. Replace it with something dead simple
- Start testing close to the service; Move outward from the service until it stops working. Test all zones a client is expected to be in.

Netcat

Simple TCP & UDP Client / Server Pair

- Shut down the target service
- Start up netcat on the port the service was running on:
`nc --keep-open --listen <port>`
- Attempt to connect from the machine:
`echo test | nc 127.0.0.1 <port>`
- Repeat to it's external address, then from the perspective of the real client
- Where does it stop working?

Netcat



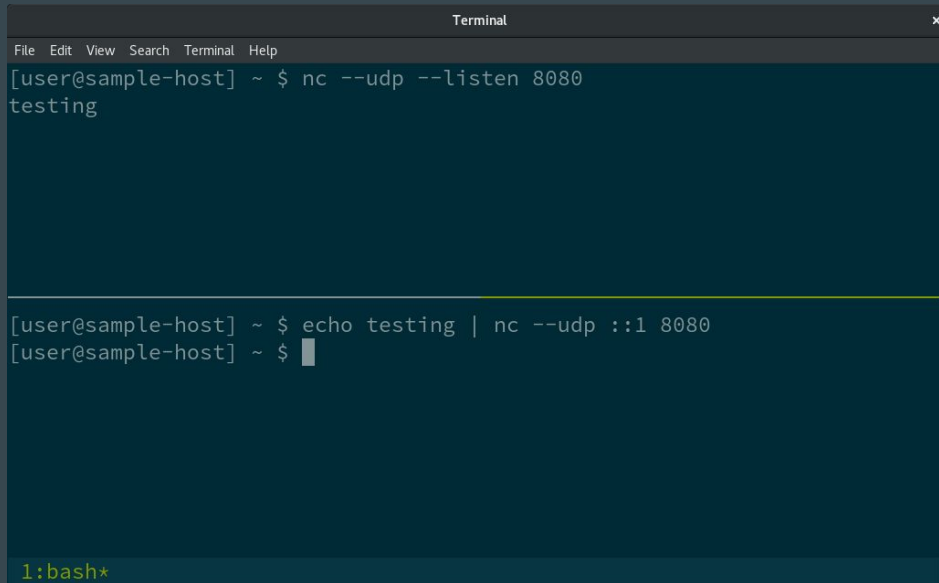
```
Terminal x
File Edit View Search Terminal Help
[user@sample-host] ~ $ nc --keep-open --listen 7000
testing

[user@sample-host] ~ $ echo testing | nc ::1 7000
[user@sample-host] ~ $ █

1: bash*
```

The image shows a terminal window titled "Terminal" with a standard menu bar (File, Edit, View, Search, Terminal, Help). The terminal content shows a Netcat listener on port 7000 receiving the word "testing" from a client. The listener then runs the command "echo testing | nc ::1 7000", which sends "testing" to a local Netcat client. The client responds with a cursor character "█". At the bottom of the terminal, there is a red prompt "1: bash*" indicating a shell connection.

Netcat



```
Terminal x
File Edit View Search Terminal Help
[user@sample-host] ~ $ nc --udp --listen 8080
testing

[user@sample-host] ~ $ echo testing | nc --udp :::1 8080
[user@sample-host] ~ $ █

1:bash*
```

The image shows a terminal window titled "Terminal" with a standard menu bar (File, Edit, View, Search, Terminal, Help). The terminal content shows a user at a "sample-host" running a Netcat listener on port 8080. It receives a connection from "testing". Then, the user runs a command to echo "testing" to a Netcat client on port 8080. The terminal shows the command being executed and a cursor. At the bottom, there is a prompt "1:bash*" in yellow.

...Ok... Maybe it is the Firewall

iptables Crash Course

Concepts

- Tables
- Chains
- Rules

Concepts - Tables

- Pre-defined and static
- All traffic passes through all tables*
- Different capabilities exist in different tables

```
Terminal
File Edit View Search Terminal Help
[user@sample-host] ~ $ sudo iptables -t filter -n -L
Chain INPUT (policy ACCEPT)
target    prot opt source                destination

Chain FORWARD (policy ACCEPT)
target    prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
[user@sample-host] ~ $

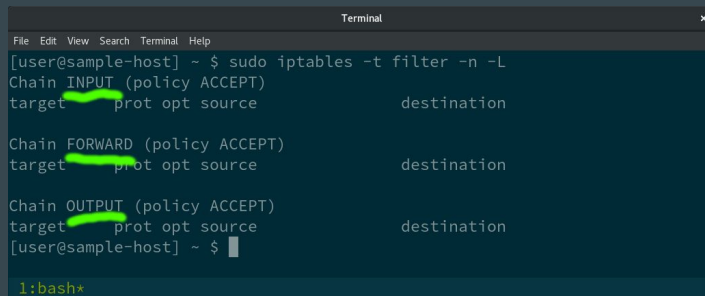
1: bash*
```

Concepts - Table List

- **filter** - traditional 'stateful firewall'
- **nat** - specify & track NAT'd connections
- **raw** - rules for untracked packets
- **mangle** - manipulate the bytes of the packet
- **security** - SELinux context tracking
- **broute** - Layer 2 routing, filtering, manipulation

Concepts - Chains

- Location where in the kernel the list of rules is getting applied
- Always displayed in ALL CAPS
- Contains a list of rules
- User can create custom chains that can be jumped to
- Defines the default action if nothing matches: ACCEPT, DROP, REJECT
- Custom chains can return to where they were called

A terminal window titled "Terminal" with a dark background and light text. The window shows the output of the command "sudo iptables -t filter -n -L". The output lists three chains: INPUT, FORWARD, and OUTPUT, each with a policy of ACCEPT. Each chain has a target field, followed by "prot opt source" and "destination" fields. The "target" field for each chain is highlighted with a yellow marker. The terminal prompt is "[user@sample-host] ~ \$" and the command prompt is "1: bash*".

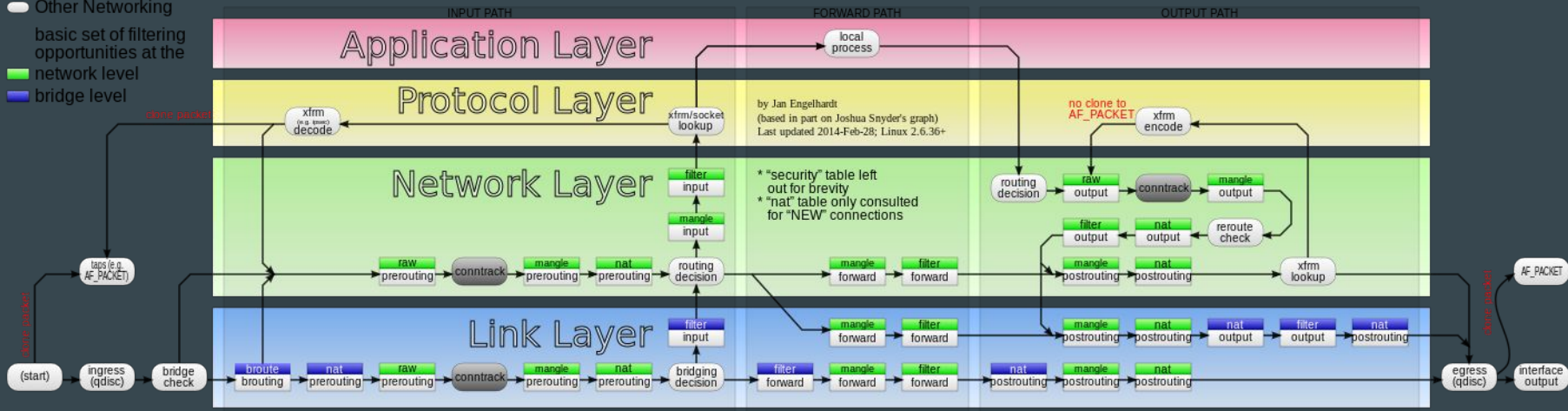
```
Terminal
File Edit View Search Terminal Help
[user@sample-host] ~ $ sudo iptables -t filter -n -L
Chain INPUT (policy ACCEPT)
target      prot opt source                destination

Chain FORWARD (policy ACCEPT)
target      prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target      prot opt source                destination
[user@sample-host] ~ $
1: bash*
```

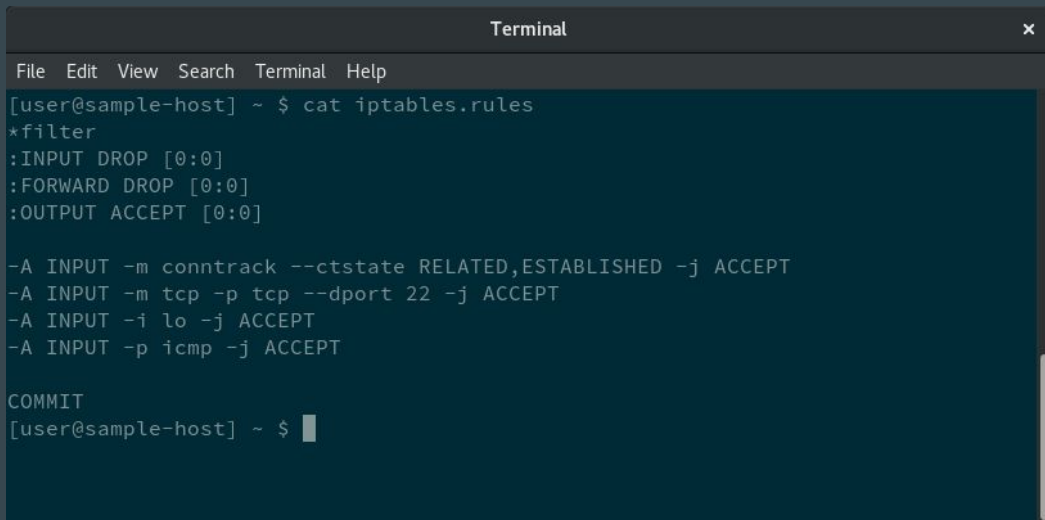
Packet flow in Netfilter and General Networking

- Other NF parts
- Other Networking
- basic set of filtering opportunities at the
- network level
- bridge level



Rules

- Individual statements of matching conditions and actions (target)
- Can choose to jump to a different chain as an action
- Each rule is evaluated in order until one matches... for every packet

A terminal window titled "Terminal" with a close button (x) in the top right corner. The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows the output of the command "cat iptables.rules". The output is as follows:

```
[user@sample-host] ~ $ cat iptables.rules
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]

-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT

COMMIT
[user@sample-host] ~ $
```


Common Rule Matchers

-m <module name> - Load a match module for this rule

-p <protocol> - Match against a specific protocol (tcp, udp, icmp, etc)

-i <interface> - Match the interface the packet entered the system on

-o <interface> - Match the interface the packet will be leaving on

-s <IP> - The source IP address of the packet

-d <IP> - The destination IP address of the packet

--dport <port> - The destination port of the packet

--ctstate <state> - The tracked connection state

Crash course complete!



Diagnosing the Rules

Reset the rule counters and monitor them:

```
iptables -Z
```

```
iptables -L -v -n --line-numbers
```

Chain INPUT (policy DROP 0 packets, 0 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	19	4319	ACCEPT	all	--	lo	*	0.0.0.0/0	0.0.0.0/0
2	4	336	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0
3	43526	26M	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
-----	------	-------	--------	------	-----	----	-----	--------	-------------

Chain OUTPUT (policy ACCEPT 116 packets, 8816 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	19	4319	ACCEPT	all	--	*	lo	0.0.0.0/0	0.0.0.0/0
2	16	1344	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0
3	42662	12M	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

ctstate RELATED,ESTABLISHED

Chain INPUT (policy DROP 0 packets, 0 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	19	4319	ACCEPT	all	--	lo	*	0.0.0.0/0	0.0.0.0/0
2	4	336	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0
3	43526	26M	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
-----	------	-------	--------	------	-----	----	-----	--------	-------------

Chain OUTPUT (policy ACCEPT 116 packets, 8816 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	19	4319	ACCEPT	all	--	*	lo	0.0.0.0/0	0.0.0.0/0
2	16	1344	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0
3	42662	12M	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

ctstate RELATED,ESTABLISHED

Chain INPUT (policy DROP 0 packets, 0 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	19	4319	ACCEPT	all	--	lo	*	0.0.0.0/0	0.0.0.0/0
2	4	336	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0
3	43526	26M	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
-----	------	-------	--------	------	-----	----	-----	--------	-------------

Chain OUTPUT (policy ACCEPT 116 packets, 8816 bytes)

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	19	4319	ACCEPT	all	--	*	lo	0.0.0.0/0	0.0.0.0/0
2	16	1344	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0
3	42662	12M	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

ctstate RELATED,ESTABLISHED

Diagnosing the Rules

Reset the rule counters and monitor them:

```
iptables -Z
```

```
iptables -L -v -n --line-numbers
```

Dump the full rule configuration:

```
iptables-save
```

```
*filter
```

```
:INPUT ACCEPT [2:120]
```

```
:FORWARD DROP [0:0]
```

```
:OUTPUT ACCEPT [81:4860]
```

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

```
-A INPUT -p icmp -j ACCEPT
```

```
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

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

```
-A OUTPUT -p icmp -j ACCEPT
```

```
-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

```
COMMIT
```


*filter

:INPUT ACCEPT [2:120]

:FORWARD DROP [0:0]

:OUTPUT ACCEPT [81:4860]

-A INPUT -i lo -j ACCEPT

-A INPUT -p icmp -j ACCEPT

-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

-A OUTPUT -o lo -j ACCEPT

-A OUTPUT -p icmp -j ACCEPT

-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

COMMIT

```
*filter
:INPUT ACCEPT [2:120]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [81:4860]

-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

COMMIT
```

```
*filter
```

```
:INPUT ACCEPT [2:120]
```

```
:FORWARD DROP [0:0]
```

```
:OUTPUT ACCEPT [81:4860]
```

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

```
-A INPUT -p icmp -j ACCEPT
```

```
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

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

```
-A OUTPUT -p icmp -j ACCEPT
```

```
-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

```
COMMIT
```

```
*filter
```

```
:INPUT ACCEPT [2:120]
```

```
:FORWARD DROP [0:0]
```

```
:OUTPUT ACCEPT [81:4860]
```

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

```
-A INPUT -p icmp -j ACCEPT
```

```
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

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

```
-A OUTPUT -p icmp -j ACCEPT
```

```
-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

```
COMMIT
```

```
*filter
:INPUT ACCEPT [2:120]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [81:4860]

-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

COMMIT
```

```
*filter
```

```
:INPUT ACCEPT [2:120]
```

```
:FORWARD DROP [0:0]
```

```
:OUTPUT ACCEPT [81:4860]
```

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

```
-A INPUT -p icmp -j ACCEPT
```

```
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

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

```
-A OUTPUT -p icmp -j ACCEPT
```

```
-A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

```
COMMIT
```

Diagnosing the Rules

Reset the rule counters and monitor them:

```
iptables -Z  
iptables -L -v -n --line-numbers
```

Dump the full rule configuration:

```
iptables-save
```

Insert tracing rule:

```
iptables -I <num> <rule conditions> (-j LOG)?  
iptables -R <num> <rule conditions> (-j LOG)?
```

Tracing Rules

```
[root@sample-host] ~ # iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 ctstate RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
3 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0
<snip>
```


Tracing Rules

```
[root@sample-host] ~ # iptables -L -n --line-numbers
```

```
Chain INPUT (policy DROP)
```

num	target	prot	opt	source	destination	
1	ACCEPT	all	--	0.0.0.0/0	0.0.0.0/0	ctstate RELATED,ESTABLISHED
2	ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0	tcp dpt:22
3	ACCEPT	icmp	--	0.0.0.0/0	0.0.0.0/0	

```
<snip>
```

```
[root@sample-host] ~ # iptables -I INPUT 2 -s 10.0.0.100
```

```
[root@sample-host] ~ # iptables -L -n -v --line-numbers
```

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
```

num	pkts	bytes	target	prot	opt	in	out	source	destination	
1	597	44180	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0	ctstate RELATED,ESTABLISHED
2	0	0		all	--	*	*	10.0.0.100	0.0.0.0/0	
3	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:22
4	0	0	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0	

```
<snip>
```

Tracing Rules

```
[root@sample-host] ~ # iptables -R INPUT 2 -s 10.0.0.100 -j LOG
[root@sample-host] ~ # iptables -L -n -v --line-numbers
Chain INPUT (policy DROP 0 packets, 0 bytes)
num  pkts  bytes  target    prot opt in  out  source      destination
1    597  44180  ACCEPT    all  --  *   *   0.0.0.0/0   0.0.0.0/0   ctstate RELATED,ESTABLISHED
2     1     1  LOG       all  --  *   *   10.0.0.100  0.0.0.0/0   LOG flags 0 level 4
3     1     1  ACCEPT    tcp  --  *   *   0.0.0.0/0   0.0.0.0/0   tcp dpt:22
4     0     0  ACCEPT    icmp --  *   *   0.0.0.0/0   0.0.0.0/0
<snip>
```

* The log target doesn't work inside of network namespaces

Tracing Rules - Log

```
IN=eth0 OUT= MAC=13:5b:00:d2:00:a0:13:5b:00:46:02:ef:01:00 SRC=10.0.0.100  
DST=10.0.0.200 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=14825 DF PROTO=TCP  
SPT=49138 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
```

Tracing Rules - Log

IN=eth0

OUT=

MAC=13:5b:00:d2:00:a0:13:5b:00:46:02:ef:01:00

SRC=10.0.0.100

DST=10.0.0.200

LEN=60

TOS=0x00

PREC=0x00

TTL=64

ID=14825

DF

PROTO=TCP

SPT=49138

DPT=22

WINDOW=29200 RES=0x00 SYN URGP=0

Don't forget about IPv6

Each iptables Utility Has an IPv6 Equivalent

```
iptables <-> ip6tables  
iptables-save <-> ip6tables-save  
iptables-restore <-> ip6tables-restore
```

Make sure you're checking both

Check nftables

Just see if it's in use:

```
nft list ruleset
```

Nothing returned? It's not in use.

```
bash: nft: command not found
```

Also probably not in use...

Performance

Not the firewall. Check the database.

Performance

- Rule evaluation is very fast
- Each packet checks each rule until one matches
- Jumps to custom table “costs” more than a normal rule evaluation
- Connection tracking is relatively expensive

**t3.small on AWS can handle ~105k pps
~1k 128kB HTTPS Requests**

200 stateful rules dropped this 5%

500 stateful rules dropped this 15%

Performance

- Minimize the number of rules
- Rules should be ordered from mostly commonly evaluated to least
- Replace individual whitelist/blacklist rules with ipset hashes
- When mitigating malicious traffic, use the raw table before connection tracking is applied

ipset

- Aggregate ports / IPs into single iptables rule
- Atomic changes (add / remove / replace)
- Inclusion check much faster than individual rule evaluation
- Performance decreases the larger the size of set

ipset - blacklist example

```
ipset create blacklist iphash
iptables -t raw -A PREROUTING -m set \
    --match-set blacklist src -j DROP
```

Add an IP address to the blacklist:

```
ipset add blacklist 1.1.1.1
```

Removing:

```
ipset del blacklist 1.1.1.1
```


Network Namespaces

- Full independent network stack
- Has separate iptables configuration
- Can be used to sneak traffic through 'FORWARD' chains rather than 'OUTPUT' chains

Check for all active network namespaces:

```
ip netns list
```

Dump iptables rules in all non-default namespaces:

```
ip -all netns exec iptables -nL
```

Back to the Tools

- firewalld (RH Default - Fedora/RHEL/CentOS)
- ufw - Uncomplicated firewall (Ubuntu)
- Shorewall
- FireHOL

More Tools

- docker
- kubernetes
- libvirt
- fail2ban
- fwsnort

Next generations...

nftables

- New syntax - same primitives
- Uses a minimal VM running in kernel space
- Incremental changes
- Faster rule evaluation and set inclusion check
- Merges IPv4 & IPv6
- Supports individual rule tracing

```
Terminal
File Edit View Search Terminal Help
[user@sample-host] ~ $ cat nftables-rules
flush ruleset

table inet filter {
  chain input {
    type filter hook input priority 0; policy drop;

    ct state invalid drop
    ct state established,related accept
    iif "lo" accept

    ip protocol icmp accept
    ip6 nexthdr icmpv6 accept

    tcp dport 22 accept

    ip protocol tcp tcp dport 67 tcp sport 68 accept
    counter
  }

  chain forward {
    type filter hook forward priority 0; policy drop;
  }

  chain output {
    type filter hook output priority 0; policy drop;

    ct state established,related accept
    oif "lo" accept

    ip protocol icmp accept
    ip6 nexthdr icmpv6 accept

    tcp dport 67 tcp sport 68 accept
    tcp dport { 53, 80, 443, 873 } accept
    udp dport 53 accept

    ct state new log level warn prefix "egress attempt: "
    counter reject with icmp type admin-prohibited
  }
}
[user@sample-host] ~ $
```

XDP / eBPF

- eBPF introduced to provide diagnostic hooks into the kernel (dtrace equivalent)
- VM in kernel space
- Must be compiled / Examples written in C
- Not remotely friendly
- Can be hardware offloaded
- Incredibly fast (3x in software, 7x on hardware)

Questions?

Site: <https://stelfox.net/>
Twitter: @SamStelfox
Email: sam@stelfox.net
GPG: 0x30856D4EA0FFBA8F
GitHub: sstelfox