thegeekstuff.com

25 Most Frequently Used Linux IPTables Rules Examples

by Ramesh Natarajan on June 14, 2011

At a first glance, IPTables rules might look cryptic.

In this article, I've given 25 practical IPTables rules that you can copy/paste and use it for your needs.

These examples will act as a basic templates for you to tweak these rules to suite your specific requirement.

For easy reference, all these 25 iptables rules are in shell script format: iptables-rules

1. Delete Existing Rules

Before you start building new set of rules, you might want to clean-up all the default rules, and existing rules. Use the iptables flush command as shown below to do this.

```
iptables -F
(or)
iptables --flush
```

2. Set Default Chain Policies

The default chain policy is ACCEPT. Change this to DROP for all INPUT, FORWARD, and OUTPUT chains as shown below.

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

When you make both INPUT, and OUTPUT chain's default policy as DROP, for every firewall rule requirement you have, you should define two rules. i.e one for incoming and one for outgoing.

In all our examples below, we have two rules for each scenario, as we've set DROP as default policy for both INPUT and OUTPUT chain.

If you trust your internal users, you can omit the last line above. i.e Do not DROP all outgoing packets by default. In that case, for every firewall rule requirement you have, you just have to define only one rule. i.e define rule only for incoming, as the outgoing is ACCEPT for all packets.

Note: If you don't know what a chain means, you should first familiarize yourself with the IPTables fundamentals.

3. Block a Specific ip-address

Before we proceed further will other examples, if you want to block a specific ip-address, you should do that first as shown below. Change the "x.x.x.x" in the following example to the specific ip-address that you like to block.

```
BLOCK_THIS_IP="x.x.x.x"
iptables -A INPUT -s "$BLOCK_THIS_IP" -j DROP
```

This is helpful when you find some strange activities from a specific ip-address in your log files, and you want to temporarily block that ip-address while you do further research.

You can also use one of the following variations, which blocks only TCP traffic on etho connection for this ip-address.

```
iptables -A INPUT -i eth0 -s "$BLOCK_THIS_IP" -j DROP
iptables -A INPUT -i eth0 -p tcp -s "$BLOCK_THIS_IP" -j DROP
```

4. Allow ALL Incoming SSH

The following rules allow ALL incoming ssh connections on etho interface.

```
iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT
```

Note: If you like to understand exactly what each and every one of the arguments means, you should read How to Add IPTables Firewall Rules

5. Allow Incoming SSH only from a Specific Network

The following rules allow incoming ssh connections only from 192.168.100.X network.

```
iptables -A INPUT -i eth0 -p tcp -s 192.168.100.0/24 --dport 22 -m
state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT
```

In the above example, instead of /24, you can also use the full subnet mask. i.e "192.168.100.0/255.255.255.0".

6. Allow Incoming HTTP and HTTPS

The following rules allow all incoming web traffic. i.e HTTP traffic to port 80.

```
iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state
ESTABLISHED -j ACCEPT
```

The following rules allow all incoming secure web traffic. i.e HTTPS traffic to port 443.

```
iptables -A INPUT -i eth0 -p tcp --dport 443 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 443 -m state --state
ESTABLISHED -j ACCEPT
```

7. Combine Multiple Rules Together using MultiPorts

When you are allowing incoming connections from outside world to multiple ports, instead of writing individual rules for each and every port, you can combine them together using the multiport extension as shown below.

The following example allows all incoming SSH, HTTP and HTTPS traffic.

```
iptables -A INPUT -i eth0 -p tcp -m multiport --dports 22,80,443 -m
state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp -m multiport --sports 22,80,443 -m
state --state ESTABLISHED -j ACCEPT
```

8. Allow Outgoing SSH

The following rules allow outgoing ssh connection. i.e When you ssh from inside to an outside server.

```
iptables -A OUTPUT -o eth0 -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT
```

Please note that this is slightly different than the incoming rule. i.e We allow both the NEW and ESTABLISHED state on the OUTPUT chain, and only ESTABLISHED state on the INPUT chain. For the incoming rule, it is vice versa.

9. Allow Outgoing SSH only to a Specific Network

The following rules allow outgoing ssh connection only to a specific network. i.e You an ssh only to 192.168.100.0/24 network from the inside.

```
iptables -A OUTPUT -o eth0 -p tcp -d 192.168.100.0/24 --dport 22 -m
state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT
```

10. Allow Outgoing HTTPS

The following rules allow outgoing secure web traffic. This is helpful when you want to allow internet traffic for your users. On servers, these rules are also helpful when you want to use wget to download some files from outside.

```
iptables -A OUTPUT -o eth0 -p tcp --dport 443 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --sport 443 -m state --state
ESTABLISHED -j ACCEPT
```

Note: For outgoing HTTP web traffic, add two additional rules like the above, and change 443 to 80.

11. Load Balance Incoming Web Traffic

You can also load balance your incoming web traffic using iptables firewall rules.

This uses the iptables nth extension. The following example load balances the HTTPS traffic to three different ip-address. For every 3th packet, it is load balanced to the appropriate server (using the counter 0).

```
iptables -A PREROUTING -i eth0 -p tcp --dport 443 -m state --state
NEW -m nth --counter 0 --every 3 --packet 0 -j DNAT --to-destination
192.168.1.101:443
iptables -A PREROUTING -i eth0 -p tcp --dport 443 -m state --state
NEW -m nth --counter 0 --every 3 --packet 1 -j DNAT --to-destination
192.168.1.102:443
iptables -A PREROUTING -i eth0 -p tcp --dport 443 -m state --state
NEW -m nth --counter 0 --every 3 --packet 2 -j DNAT --to-destination
192.168.1.103:443
```

12. Allow Ping from Outside to Inside

The following rules allow outside users to be able to ping your servers.

```
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

13. Allow Ping from Inside to Outside

The following rules allow you to ping from inside to any of the outside servers.

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

14. Allow Loopback Access

You should allow full loopback access on your servers. i.e access using 127.0.0.1

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

15. Allow Internal Network to External network.

On the firewall server where one ethernet card is connected to the external, and another ethernet card connected to the internal servers, use the following rules to allow internal network talk to external network.

In this example, eth1 is connected to external network (internet), and eth0 is connected to internal network (For example: 192.168.1.x).

```
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

16. Allow outbound DNS

The following rules allow outgoing DNS connections.

```
iptables -A OUTPUT -p udp -o eth0 --dport 53 -j ACCEPT
iptables -A INPUT -p udp -i eth0 --sport 53 -j ACCEPT
```

17. Allow NIS Connections

If you are running NIS to manage your user accounts, you should allow the NIS connections. Even when the SSH connection is allowed, if you don't allow the NIS related ypbind connections, users will not be able to login.

The NIS ports are dynamic. i.e When the ypbind starts it allocates the ports.

First do a rpcinfo -p as shown below and get the port numbers. In this example, it was using port 853 and 850.

```
rpcinfo -p | grep ypbind
```

Now allow incoming connection to the port 111, and the ports that were used by ypbind.

```
iptables -A INPUT -p tcp --dport 111 -j ACCEPT
iptables -A INPUT -p udp --dport 111 -j ACCEPT
iptables -A INPUT -p tcp --dport 853 -j ACCEPT
iptables -A INPUT -p udp --dport 853 -j ACCEPT
iptables -A INPUT -p tcp --dport 850 -j ACCEPT
iptables -A INPUT -p udp --dport 850 -j ACCEPT
```

The above will not work when you restart the ypbind, as it will have different port numbers that time.

There are two solutions to this: 1) Use static ip-address for your NIS, or 2) Use some clever shell scripting techniques to automatically grab the dynamic port number from the "rpcinfo -p" command output, and use those in the above iptables rules.

18. Allow Rsync From a Specific Network

The following rules allows rsync only from a specific network.

```
iptables -A INPUT -i eth0 -p tcp -s 192.168.101.0/24 --dport 873 -m
state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 873 -m state --state
ESTABLISHED -j ACCEPT
```

19. Allow MySQL connection only from a specific network

If you are running MySQL, typically you don't want to allow direct connection from outside. In most cases, you might have web server running on the same server where the MySQL database runs.

However DBA and developers might need to login directly to the MySQL from their laptop and desktop using MySQL client. In those case, you might want to allow your internal network to talk to the MySQL directly as shown below.

```
iptables -A INPUT -i eth0 -p tcp -s 192.168.100.0/24 --dport 3306 -m
state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 3306 -m state --state
ESTABLISHED -j ACCEPT
```

20. Allow Sendmail or Postfix Traffic

The following rules allow mail traffic. It may be sendmail or postfix.

```
iptables -A INPUT -i eth0 -p tcp --dport 25 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 25 -m state --state
ESTABLISHED -j ACCEPT
```

21. Allow IMAP and IMAPS

The following rules allow IMAP/IMAP2 traffic.

```
iptables -A INPUT -i eth0 -p tcp --dport 143 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 143 -m state --state
ESTABLISHED -j ACCEPT
```

The following rules allow IMAPS traffic.

```
iptables -A INPUT -i eth0 -p tcp --dport 993 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 993 -m state --state
ESTABLISHED -j ACCEPT
```

22. Allow POP3 and POP3S

The following rules allow POP3 access.

```
iptables -A INPUT -i eth0 -p tcp --dport 110 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 110 -m state --state
ESTABLISHED -j ACCEPT
```

The following rules allow POP3S access.

```
iptables -A INPUT -i eth0 -p tcp --dport 995 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 995 -m state --state
ESTABLISHED -j ACCEPT
```

23. Prevent DoS Attack

The following iptables rule will help you prevent the Denial of Service (DoS) attack on your webserver.

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute
--limit-burst 100 -j ACCEPT
```

In the above example:

- -m limit: This uses the limit iptables extension
- –limit 25/minute: This limits only maximum of 25 connection per minute. Change this value based on your specific requirement
- –limit-burst 100: This value indicates that the limit/minute will be enforced only after the total number of connection have reached the limit-burst level.

24. Port Forwarding

The following example routes all traffic that comes to the port 442 to 22. This means that the incoming ssh connection can come from both port 22 and 422.

```
iptables -t nat -A PREROUTING -p tcp -d 192.168.102.37 --dport 422
-j DNAT --to 192.168.102.37:22
```

If you do the above, you also need to explicitly allow incoming connection on the port 422.

```
iptables -A INPUT -i eth0 -p tcp --dport 422 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 422 -m state --state
ESTABLISHED -j ACCEPT
```

25. Log Dropped Packets

You might also want to log all the dropped packets. These rules should be at the bottom.

First, create a new chain called LOGGING.

```
iptables -N LOGGING
```

Next, make sure all the remaining incoming connections jump to the LOGGING chain as shown below.

```
iptables -A INPUT -j LOGGING
```

Next, log these packets by specifying a custom “log-prefix”.

```
iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix
"IPTables Packet Dropped: " --log-level 7
```

Finally, drop these packets.

```
iptables -A LOGGING -j DROP
```

All of the above 25 iptables rules are in shell script format: iptables-rules

Previous articles in the iptables series:

- Linux Firewall Tutorial: IPTables Tables, Chains, Rules Fundamentals
- IPTables Flush: Delete / Remove All Rules On RedHat and CentOS Linux
- Linux IPTables: How to Add Firewall Rules (With Allow SSH Example)
- Linux IPTables: Incoming and Outgoing Rule Examples (SSH and HTTP)

If you enjoyed this article, you might also like..