Search guide

In This Guide:

Use Linux iptables to Manage IPv4 Traffic

Configure iptables

Use ip6tables to Manage IPv6 Traffic

Basic iptables Rulesets for IPv4 and IPv6

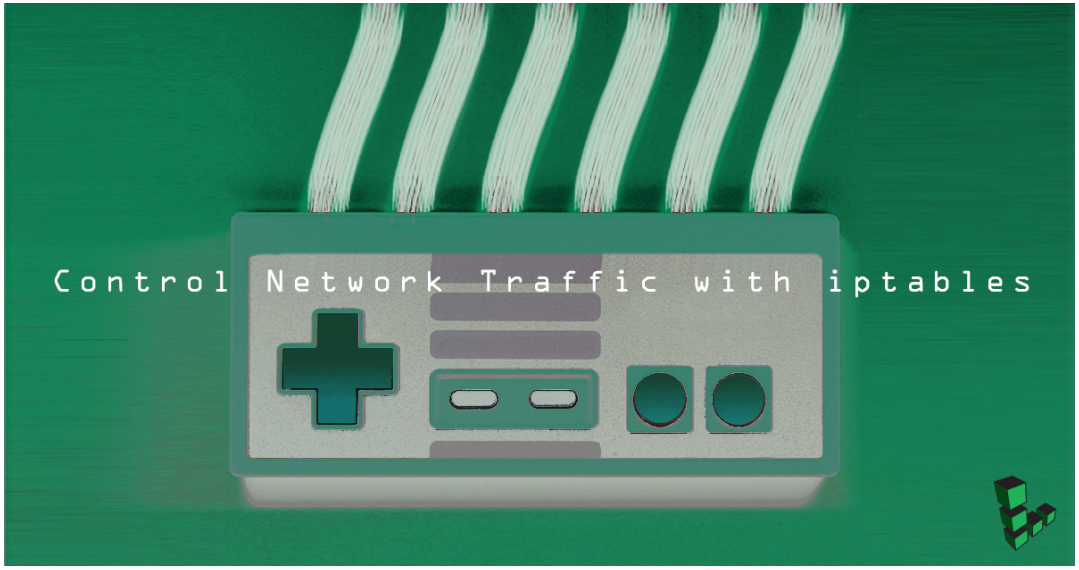
Control Network Traffic with iptables

Updated Monday, July 23, 2018 by Linode

Written by Linode

Contribute on GitHub Report an Issue | View File | Edit File

iptables is an application that allows users to configure specific rules that will be enforced by the kernel's `netfilter` framework. It acts as a packet filter and firewall that examines and directs traffic based on port, protocol and other criteria. This guide will focus on the configuration and application of iptables rulesets and will provide examples of ways they are commonly used.



By default, the iptables tool is included with your Linode-supplied distribution. In order to use iptables, you will need root (`sudo`) privileges.

Use Linux iptables to Manage IPv4 Traffic

The iptables Command

Many options can be used with the iptables command. As stated above, iptables sets the rules that control network traffic. You can define different tables to handle these rules through chains, lists of rules that match a subset of packets. The table contains a variety of built-in chains, but you can add your own.

Basic iptables Parameters and Syntax

Before we begin creating rules, let's review the syntax of an iptables rule.

For example, the following command adds a rule to the beginning of the chain that will drop all packets from the address `198.51.100.0`:

```
iptables -I INPUT -s 198.51.100.0 -j DROP
```

The sample command above:

1. Calls the `iptables` program
2. Uses the `-I` option for *insertion*. Using a rule with the insertion option will add it to the beginning of a chain and will be applied first. To indicate a specific placement in the chain, you may also use a number with the `-I` option.
3. The `-s` parameter, along with the IP address (198.51.100.0), indicates the *source*.
4. Finally, the `-j` parameter stands for *jump*. It specifies the target of the rule and what action will be performed if the packet is a match.

Parameter	Description

Parameter	Description
`-p, --protocol`	The protocol, such as TCP, UDP, etc.
`-s, --source`	Can be an address, network name, hostname, etc.
`-d, --destination`	An address, hostname, network name, etc.
`-j, --jump`	Specifies the target of the rule; i.e. what to do if the packet matches.
`-g, --goto chain`	Specifies that the processing will continue in a user-specified chain.
`-i, --in-interface`	Names the interface from where packets are received.
`-o, --out-interface`	Name of the interface by which a packet is being sent.
`-f, --fragment`	The rule will only be applied to the second and subsequent fragments of fragmented packets.

Parameter	Description
`-c, --set-counters`	Enables the admin to initialize the packet and byte counters of a rule.

Default Tables

Tables are made up of built-in chains and may also contain user-defined chains. The built-in tables will depend on the kernel configuration and the installed modules.

The default tables are as follows:

- **Filter** - This is the default table. Its built-in chains are:
    - Input: packets going to local sockets
    - Forward: packets routed through the server
    - Output: locally generated packets
- **Nat** - When a packet creates a new connection, this table is used. Its built-in chains are:
    - Prerouting: designating packets when they come in
    - Output: locally generated packets before routing takes place
    - Postrouting: altering packets on the way out

- **Mangle** - Used for special altering of packets. Its chains are:
    - Prerouting: incoming packets
    - Postrouting: outgoing packets
    - Output: locally generated packets that are being altered
    - Input: packets coming directly into the server
    - Forward: packets being routed through the server
- **Raw** - Primarily used for configuring exemptions from connection tracking. The built-in chains are:
    - Prerouting: packets that arrive by the network interface
    - Output: processes that are locally generated
- **Security** - Used for Mandatory Access Control (MAC) rules. After the filter table, the security table is accessed next. The built-in chains are:
    - Input: packets entering the server
    - Output: locally generated packets
    - Forward: packets passing through the server

Basic iptables Options

There are many options that may be used with the `iptables` command:

Option	Description
`-A --append`	Add one or more rules to the end of the selected chain.
`-C --check`	Check for a rule matching the specifications in the selected chain.
`-D --delete`	Delete one or more rules from the selected chain.
`-F --flush`	Delete *all* the rules one-by-one.
`-I --insert`	Insert one or more rules into the selected chain as the given rule number.
`-L --list`	Display the rules in the selected chain.
`-n --numeric`	Display the IP address or hostname and post number in numeric format.
`-N --new-chain <name>`	Create a new user-defined chain.
`-v --verbose`	Provide more information when used with the list option.

Option	Description
`-X --delete-chain <name>`	Delete the user-defined chain.

Insert, Replace or Delete iptables Rules

iptables rules are enforced top down, so the first rule in the ruleset is applied to traffic in the chain, then the second, third and so on. This means that rules cannot necessarily be added to a ruleset with `iptables -A` or `ip6tables -A`. Instead, rules must be *inserted* with `iptables -I` or `ip6tables -I`.

Insert

Inserted rules need to be placed in the correct order with respect to other rules in the chain. To get a numerical list of your iptables rules:

```
sudo iptables -L -nv --line-numbers
```

For example, let's say you want to insert a rule into the basic ruleset provided in this guide, that will accept incoming connections to port 8080 over the TCP protocol. We'll add it as rule 7 to the INPUT chain, following the web traffic rules:

```
sudo iptables -I INPUT 7 -p tcp --dport 8080 -m state --state NEW -j ACCE
```

If you now run `sudo iptables -L -nv` again, you'll see the new rule in the output.

Replace

Replacing a rule is similar to inserting, but instead uses `iptables -R`. For example, let's say you want to reduce the logging of denied entries to only 3 per minute, down from 5 in the original ruleset. The LOG rule is ninth in the INPUT chain:

```
sudo iptables -R INPUT 9 -m limit --limit 3/min -j LOG --log-prefix "ipta
```

Delete

Deleting a rule is also done using the rule number. For example, to delete the rule we just inserted for port 8080:

```
sudo iptables -D INPUT 7
```

> **Caution**
> Editing rules does not automatically save them. See our section on deploying rulesets for the specific instructions for your distribution.

View Your Current iptables Rules

IPv4:

```
sudo iptables -L -nv
```

IPv6:

```
sudo ip6tables -L -nv
```

On most distributions, iptables has no default rules for either IPv4 and IPv6. As a result, on a newly created Linode you will likely see what is shown below - three empty chains without any firewall rules. This means that all incoming, forwarded and outgoing traffic is *allowed*. It's important to limit inbound and forwarded traffic to only what's necessary.

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

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               desti

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

Configure iptables

iptables can be configured and used in a variety of ways. The following sections will outline how to configure rules by port and IP, as well as

how to blacklist (block) or whitelist (allow) addresses.

Block Traffic by Port

You may use a port to block all traffic coming in on a specific interface. For example:

```
iptables -A INPUT -j DROP -p tcp --destination-port 110 -i eth0
```

Let's examine what each part of this command does:

- `-A` will add or append the rule to the end of the chain.
- `INPUT` will add the rule to the table.
- `DROP` means the packets are discarded.
- `-p tcp` means the rule will only drop TCP packets.
- `--destination-port 110` filters packets targeted to port 110.
- `-i eth0` means this rule will impact only packets arriving on the *eth0* interface.

It is important to understand that iptables do *not* recognize aliases on the network interface. Therefore, if you have several virtual IP interfaces, you will have to specify the destination address to filter the traffic. A sample command is provided below:

```
iptables -A INPUT -j DROP -p tcp --destination-port 110 -i eth0 -d 198.51
```

You may also use `-D` or `--delete` to remove rules. For example, these commands are equivalent:

```
iptables --delete INPUT -j DROP -p tcp --destination-port 110 -i eth0 -d
iptables -D INPUT -j DROP -p tcp --destination-port 110 -i eth0 -d 198.51
```

Drop Traffic from an IP

In order to drop all incoming traffic from a specific IP address, use the iptables command with the following options:

```
iptables -I INPUT -s 198.51.100.0 -j DROP
```

To remove these rules, use the `--delete` or `-D` option:

```
iptables --delete INPUT -s 198.51.100.0 -j DROP
iptables -D INPUT -s 198.51.100.0 -j DROP
```

Block or Allow Traffic by Port Number to Create an iptables Firewall

One way to create a firewall is to block all traffic to the system and then allow traffic on certain ports. Below is a sample sequence of commands to illustrate the process:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -m comment --comment "Allow loopback connections"
iptables -A INPUT -p icmp -m comment --comment "Allow Ping to work as exp
iptables -A INPUT -p tcp -m multiport --destination-ports 22,25,53,80,443
iptables -A INPUT -p udp -m multiport --destination-ports 53 -j ACCEPT
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

Let's break down the example above. The first two commands add or append rules to the `INPUT` chain in order to allow access on specific ports. The `-p tcp` and `-p udp` options specify either UDP or TCP packet types. The `-m multiport` function matches packets on the basis of their source or destination ports, and can accept the specification of up to 15 ports. Multiport also accepts ranges such as `8999:9003` which counts as 2 of the 15 possible ports, but matches ports `8999`, `9000`, `9001`, `9002`, and `9003`. The next command allows all incoming and outgoing packets that are associated with existing connections so that they will not be inadvertently blocked by the firewall. The final two commands use the `-P` option to describe the *default policy* for these chains. As a result, all packets processed by `INPUT` and `FORWARD` will be dropped by default.

Note that the rules described above only control incoming packets, and do not limit outgoing connections.

Whitelist/Blacklist Traffic by Address

You can use iptables to block all traffic and then only allow traffic from certain IP addresses. These firewall rules limit access to specific resources at the network layer. Below is an example sequence of commands:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -m comment --comment "Allow loopback connections"
iptables -A INPUT -p icmp -m comment --comment "Allow Ping to work as exp
iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
iptables -A INPUT -s 198.51.100.0 -j ACCEPT
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

In the first command, the `-s 192.168.1.0/24` statement specifies that all source IPs (`-s`) in the address space of `192.168.1` are allowed. You may specify an IP address range using CIDR (Classless Inter-Domain Routing) notation, or individual IP addresses, as in the second command. The third command allows all incoming and outgoing packets that are associated with existing connections. The final two commands set the default policy for all `INPUT` and `FORWARD` chains to drop all packets.

Use ip6tables to Manage IPv6 Traffic

When you're working with IPv6, remember that the `iptables` command is not compatible. Instead, there is an `ip6tables` command. The

options such as append, check, etc. are the same. The tables used by ip6tables are *raw*, *security*, *mangle* and *filter*. The parameters such as protocol, source, etc. are the same. The syntax is essentially the same as IPv4. Sample syntax is below:

```
ip6tables [-t table] -N chain
```

To view what rules are configured for IPv6, use the command:

```
ip6tables -L
```

Configure Rules for IPv6

ip6tables works by using ports, specific addresses for blacklisting, protocols and so forth. The primary difference is that ip6tables can use extended packet matching modules with the `-m` or `match` options, followed by the module name. Below are some of the extended modules:

- **addrtype** - Matches packets based on their address type. Some of the address types are:
    - Local
    - Unicast
    - Broadcast
    - Multicast

- **ah** - Matches the parameters in the authentication header of IPsec packets.
- **cluster** - You can deploy gateway and backend load-sharing clusters without a load balancer.
- **comment** - Allows you to add a comment to any rule.
- **connbytes** - Matches by how many bytes or packets a connection has transferred, or average bytes per packet.

This is not intended to be a complete or comprehensive list. You may review the full list of extended modules by using the `man` page:

```
man ip6tables
```

Below is a sample rule used in ip6tables:

```
# limit the number of parallel HTTP requests to 16 for the link local net
ip6tables -A INPUT -p tcp --syn --dport 80 -s fe80::/64 -m connlimit --cc
ip6tables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
```

This rule breaks down as follows:

- The first line is a comment.
- `-A` is for append.
- `INPUT` is to add the rule to the table.
- `-p` is for protocol, which is TCP.

- `--syn` only matches TCP packets with the SYN bit set and the ACK, RST, and FIN bits cleared.
- `--dport` is the destination port, which is 80.
- `-s` is the source, which is the local address range fe80::/64.
- `-m` is for match.
- `connlimit` is the extended packet module name, which is connection limit.
- `--connlimit-above 16` means if the number of connections exceeds 16, only the first 16 will be used.
- `--connlimit-mask 64` means the group hosts are using a prefix length of 64.
- `-j` is for jump, it tells the target of the rule what to do if the packet is a match.
- `REJECT` means the packet is dropped.

Required Rules for Non-Static IPv6 Allocations

```
# Below are the rules which are required for your IPv6 address to be prop
ip6tables -A INPUT -p icmpv6 --icmpv6-type router-advertisement -m hl --h
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-solicitation -m hl --
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-advertisement -m hl -
ip6tables -A INPUT -p icmpv6 --icmpv6-type redirect -m hl --hl-eq 255 -j
```

Basic iptables Rulesets for IPv4 and IPv6

Appropriate firewall rules depend on the services being run. Below are iptables rulesets to secure your Linode if you're running a web server.

> **Caution**
> **These rules are given only as an example.** A real production web server may require more or less configuration, and these rules would not be appropriate for a database, Minecraft or VPN server. Iptables rules can always be modified or reset later, but these basic rulesets serve as a demonstration.

IPv4

/tmp/v4

```
1  *filter
2
3  # Allow all loopback (lo0) traffic and reject traffic
4  # to localhost that does not originate from lo0.
5  -A INPUT -i lo -j ACCEPT
6  -A INPUT ! -i lo -s 127.0.0.0/8 -j REJECT
7
8  # Allow ping.
9  -A INPUT -p icmp -m state --state NEW --icmp-type 8 -j ACCEPT
10
11 # Allow SSH connections.
12 -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
13
14 # Allow HTTP and HTTPS connections from anywhere
15 # (the normal ports for web servers).
16 -A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
```

```
-A INPUT -p tcp --dport 443 -m state --state NEW -j ACCEPT

# Allow inbound traffic from established connections.
# This includes ICMP error returns.
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Log what was incoming but denied (optional but useful).
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables_INPU

# Reject all other inbound.
-A INPUT -j REJECT

# Log any traffic that was sent to you
# for forwarding (optional but useful).
-A FORWARD -m limit --limit 5/min -j LOG --log-prefix "iptables_FO

# Reject all traffic forwarding.
-A FORWARD -j REJECT

COMMIT
```

Optional: If you plan to use Linode Longview or Linode's NodeBalancers, add the respective rule after the section for allowing HTTP and HTTPS connections:

```
# Allow incoming Longview connections from longview.linode.com
-A INPUT -s 96.126.119.66 -m state --state NEW -j ACCEPT

# Allow incoming NodeBalancer connections
-A INPUT -s 192.168.255.0/24 -m state --state NEW -j ACCEPT
```

IPv6

If you would like to supplement your web server's IPv4 rules with IPv6 as well, this ruleset will allow HTTP/S access and all ICMP functions.

/tmp/v6

```
*filter

# Allow all loopback (lo0) traffic and reject traffic
# to localhost that does not originate from lo0.
-A INPUT -i lo -j ACCEPT
-A INPUT ! -i lo -s ::1/128 -j REJECT

# Allow ICMP
-A INPUT -p icmpv6 -j ACCEPT

# Allow HTTP and HTTPS connections from anywhere
# (the normal ports for web servers).
-A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
-A INPUT -p tcp --dport 443 -m state --state NEW -j ACCEPT

# Allow inbound traffic from established connections.
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Log what was incoming but denied (optional but useful).
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "ip6tables_INE

# Reject all other inbound.
-A INPUT -j REJECT

# Log any traffic that was sent to you
# for forwarding (optional but useful).
-A FORWARD -m limit --limit 5/min -j LOG --log-prefix "ip6tables_F

# Reject all traffic forwarding.
-A FORWARD -j REJECT

COMMIT
```

Note
APT attempts to resolve mirror domains to IPv6 as a result of `apt-get update`. If you choose to entirely disable and deny IPv6, this will slow down the update process for Debian and Ubuntu because APT waits for each resolution to time out before moving on.

To remedy this, uncomment the line `precedence ::ffff:0:0/96 100` in `/etc/gai.conf`.

Deploy Your iptables Rulesets

The process for deploying iptables rulesets varies depending on which Linux distribution you're using:

Debian / Ubuntu

UFW is the iptables controller included with Ubuntu, but it is also available in Debian's repositories. If you prefer to use UFW instead of iptables, see our guide: How to Configure a Firewall with UFW.

1. Create the files `/tmp/v4` and `/tmp/v6`. Paste the above rulesets into their respective files.
2. Import the rulesets into immediate use:

```
sudo iptables-restore < /tmp/v4
sudo ip6tables-restore < /tmp/v6
```

3. To apply your iptables rules automatically on boot, see our section on configuring iptables-persistent.

CentOS / Fedora

CentOS 7 or Fedora 20 and above

In these distros, FirewallD is used to implement firewall rules instead of using the iptables command. If you prefer to use it over iptables, see our guide: Introduction to FirewallD on CentOS.

1. If you prefer to use iptables, FirewallD must first be stopped and disabled.

```
sudo systemctl stop firewalld.service && sudo systemctl disable fir
```

2. Install `iptables-services` and enable iptables and ip6tables:

```
sudo yum install iptables-services
sudo systemctl enable iptables && sudo systemctl enable ip6tables
sudo systemctl start iptables && sudo systemctl start ip6tables
```

3. Create the files `/tmp/v4` and `/tmp/v6`. Paste the rulesets above into their respective files.

4. Import the rulesets into immediate use:

```
sudo iptables-restore < /tmp/v4
sudo ip6tables-restore < /tmp/v6
```

5. Save each ruleset:

```
sudo service iptables save
sudo service ip6tables save
```

6. Remove the temporary rule files:

```
sudo rm /tmp/{v4,v6}
```

CentOS 6

1. Create the files `/tmp/v4` and `/tmp/v6`. Paste the rulesets above into their respective files.
2. Import the rules from the temporary files:

```
sudo iptables-restore < /tmp/v4
sudo ip6tables-restore < /tmp/v6
```

3. Save the rules:

```
sudo service iptables save
sudo service ip6tables save
```

> **Note**
> Firewall rules are saved to `/etc/sysconfig/iptables` and `/etc/sysconfig/ip6tables`.

4. Remove the temporary rule files:

```
sudo rm /tmp/{v4,v6}
```

Arch Linux

1. Create the files `/etc/iptables/iptables.rules` and `/etc/iptables/ip6tables.rules`. Paste the rulesets above into their respective files.
2. Import the rulesets into immediate use:

```
sudo iptables-restore < /etc/iptables/iptables.rules
sudo ip6tables-restore < /etc/iptables/ip6tables.rules
```

3. iptables does not run by default in Arch. Enable and start the systemd units:

```
sudo systemctl start iptables && sudo systemctl start ip6tables
sudo systemctl enable iptables && sudo systemctl enable ip6tables
```

   For more info on using iptables in Arch, see its Wiki entries for iptables and a simple stateful firewall.

Verify iptables Rulesets

Check your Linode's firewall rules with the `v` option for a verbose output:

```
sudo iptables -vL
sudo ip6tables -vL
```

The output for IPv4 rules should show:

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               desti
    0     0 ACCEPT     all  --  lo     any     anywhere             anywh
    0     0 REJECT     all  --  !lo    any     loopback/8           anywh
    0     0 ACCEPT     icmp --  any    any     anywhere             anywh
    0     0 ACCEPT     icmp --  any    any     anywhere             anywh
    0     0 ACCEPT     icmp --  any    any     anywhere             anywh
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywh
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywh
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywh
    0     0 ACCEPT     all  --  any    any     anywhere             anywh
    0     0 LOG        all  --  any    any     anywhere             anywh
    0     0 REJECT     all  --  any    any     anywhere             anywh

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               desti
    0     0 LOG        all  --  any    any     anywhere             anywh
    0     0 REJECT     all  --  any    any     anywhere             anywh

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

Output for IPv6 rules will look like this:

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               desti
    0     0 ACCEPT     all  --  lo     any     anywhere             anywh
    0     0 REJECT     all  --  !lo    any     localhost            anywh
    0     0 ACCEPT     ipv6-icmp    any    any     anywhere             a
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywh
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywh
    0     0 ACCEPT     all  --  any    any     anywhere             anywh
    0     0 LOG        all  --  any    any     anywhere             anywh
    0     0 REJECT     all  --  any    any     anywhere             anywh

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               desti
    0     0 LOG        all  --  any    any     anywhere             anywh
    0     0 REJECT     all  --  any    any     anywhere             anywh

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

Your firewall rules are now in place and protecting your Linode. Remember, you may need to edit these rules later if you install other packages that require network access.

Introduction to iptables-persistent

Ubuntu and Debian have a package called **iptables-persistent** that makes it easy to reapply your firewall rules at boot time. After installation, you can save all your rules in two files (one for IPv4 and one for IPv6). If you've already configured and applied iptables rules, iptables-persistent will detect them automatically and allow you to add them to the appropriate configuration file.

Install iptables-persistent

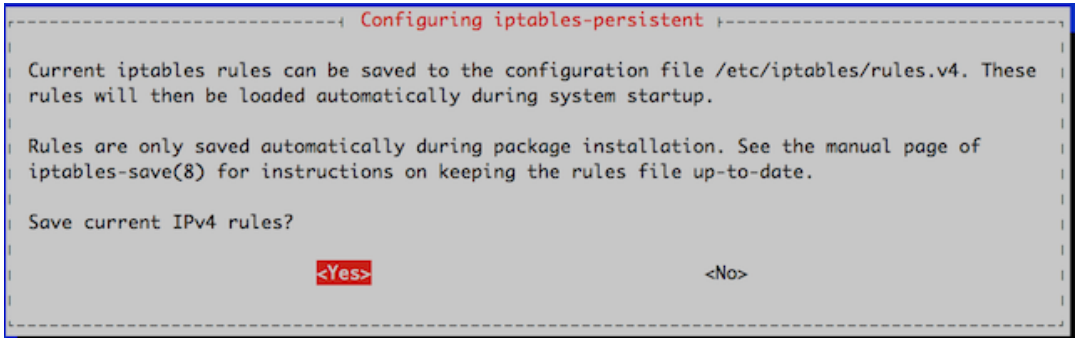
On Debian or Ubuntu use the following command to check whether `iptables-persistent` is already installed:

```
dpkg -l iptables-persistent
```

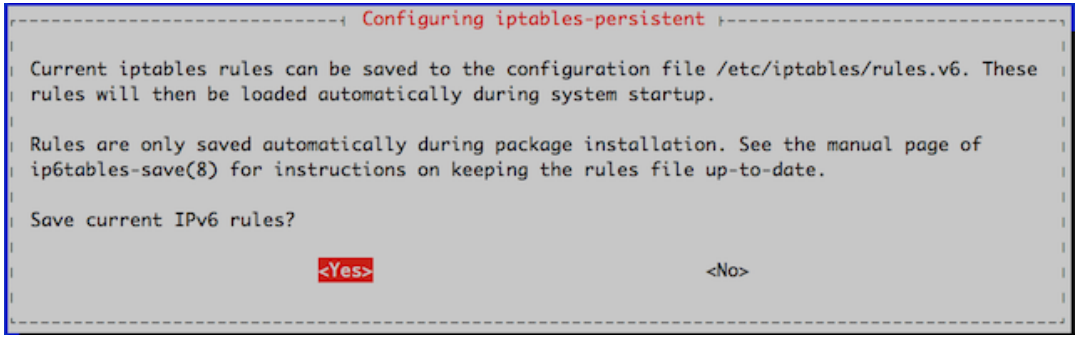
If `dpkg` returns that there are no matching packages, you will need to install the `iptables-persistent` package:

```
apt-get install iptables-persistent
```

During the installation, you will be prompted twice. The first prompt is asking if you would like to save your current IPv4 rules.



The second prompt is to save the rules configured for IPv6.



After the install is complete, you should see the iptables's subdirectory. Run the `ls /etc/iptables` command again to verify that your output resembles the following:

```
rules.v4  rules.v6
```

Use iptables-persistent

To view what rules are already configured on your server:

```
iptables -L
```

You should see output similar to:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  198.51.100.0         anywhere

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

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

The rules above allow anyone anywhere access to everything. If your output resembles this, you'll need to set rules that prevent unauthorized access.

iptables-persistent Rules

Use the `rules.v4` or `rules.v6` files to add, delete or edit the rules for your server. These files can be edited using a text editor to function as a proxy, NAT or firewall. The configuration depends on the requirements of your server and what functions are needed. Below is a file excerpt from both the `rules.v4` and `rules.v6` files:

/etc/iptables/rules.v4

```
1  # Generated by iptables-save v1.4.14 on Wed Apr  2 13:24:27 2014
2  *security
3  :INPUT ACCEPT [18483:1240117]
4  :FORWARD ACCEPT [0:0]
5  :OUTPUT ACCEPT [17288:2887358]
6  COMMIT
```

/etc/iptables/rules.v6

```
1  # Generated by ip6tables-save v1.4.14 on Wed Apr  2 13:24:27 2014
2  *nat
3  :PREROUTING ACCEPT [0:0]
4  :INPUT ACCEPT [0:0]
5  :OUTPUT ACCEPT [27:2576]
```

```
:POSTROUTING ACCEPT [27:2576]
COMMIT
```

While some rules are configured in these files already, either file can be edited at any time. The syntax for altering table rules is the same as in the sections Configure iptables and Configuring Rules for IPv6.

Save iptables-persistent Rules Through Reboot

By default, iptables-persistent rules save on reboot for IPv4 only. Therefore, if you are running both IPv4 and IPv6 together you will need to manually edit both the `rules.v4` and `rules.v6` files. On older systems, `iptables-save` was used to write the changes to the `rules` file. Now that `iptables-persistent` is an option, do not use the `iptables-save > /etc/iptables/rules.v4` or `iptables-save > /etc/iptables/rules.v6` commands as any IPv6 changes will be overwritten by the IPv4 rules.

To enforce the iptables rules and ensure that they persist after reboot run `dpkg-reconfigure` and respond **Yes** when prompted. (If you ever edit your saved rules in the future, use this same command to save them again.)

```
dpkg-reconfigure iptables-persistent
```

To verify the rules are applied and available after the system reboot use the commands:

```
iptables -L
ip6tables -L
```

Network Lock-out

When you're applying network rules, especially with both IPv4 and IPv6 and multiple interfaces, it is easy to lock yourself out. In the event you apply the rule and are unable to access your server, you may gain access through Lish in the Linode Manager. The following steps will guide you through using the graphical interface of your Linode to gain access to your server:

1. Connect to your Linode Manager.
2. Click on the Remote Access tab.
3. Under the section entitled "Console Access," click on the **Launch Lish Console** link.
4. Login with your root or sudo user name and password.
5. Remove any rules causing the connectivity issues.
6. Log out of the Lish window.
7. Attempt login via a regular SSH session.

This Lish console will function similarly to a regular SSH terminal session.

Troubleshooting: netfilter-persistent doesn't come back up on reboot.

If you have upgraded to Debian 8 from an earlier version, you may see a situation where netfilter-persistent fails to start during boot when using the Linode kernel. The console output will show similar to:

```
[FAILED] Failed to start Load Kernel Modules.
See 'systemctl status systemd-modules-load.service' for details.
[DEPEND] Dependency failed for netfilter persistent configuration
```

You can also use `journalctl -xn` to see that systemd can not load the `loop` module:

```
systemd-modules-load[3452]: Failed to lookup alias 'loop': Function not i
```

To fix this, comment out the line `loop` in `/etc/modules`:

```
sed -i 's/loop/#loop/g' /etc/modules
```

Then restart netfilter-persistent:

```
systemctl restart netfilter-persistent
```

It should then be running fine. Confirm with:

```
systemctl status netfilter-persistent
```

This issue does not occur in new deployments of Debian 8 because the `loop` line isn't present in `/etc/modules`.

More Information

You may wish to consult the following resources for additional information on this topic. While these are provided in the hope that they will be useful, please note that we cannot vouch for the accuracy or timeliness of externally hosted materials.

- Security Basics
- Using the Linode Shell (Lish)
- iptables: Linux firewall rules for a basic Web Server
- Linux Firewalls with iptables

Join our Community

Find answers, ask questions, and help others.

Write for Linode.

We're always expanding our docs. If you like to help people, can write, and have expertise in a Linux or cloud infrastructure topic, learn how you can contribute to our library.

Get started in the Linode Cloud today.

Create an Account