Iptables Essentials: Common Firewall Rules and Commands

Posted August 10, 2015 480.9k FIREWALL CENTOS UBUNTU

Introduction

Iptables is the software firewall that is included with most Linux distributions by default. This cheat sheet-style guide provides a quick reference to ipta scenarios. This includes iptables examples of allowing and blocking various services by port, network interface, and source IP address.

How To Use This Guide

- If you are just getting started with configuring your iptables firewall, check out our introduction to iptables
- Most of the rules that are described here assume that your iptables is set to **DROP** incoming traffic, through the default input policy, and you want to s
- Use whichever subsequent sections are applicable to what you are trying to achieve. Most sections are not predicated on any other, so you can use t
- Use the Contents menu on the right side of this page (at wide page widths) or your browser's find function to locate the sections you need
- Copy and paste the command-line examples given, substituting the values in red with your own values

Keep in mind that the order of your rules matter. All of these `iptables` commands use the `-A` option to append the new rule to the end of a chain. If y allows you to specify the position of the new rule (or simply place it at the beginning of the chain by not specifying a rule number).

Note: When working with firewalls, take care not to lock yourself out of your own server by blocking SSH traffic (port 22, by default). If you lose access due to access. Once you are connected via the console, you can change your firewall rules to allow SSH access (or allow all traffic). If your saved firewall rules allow

Remember that you can check your current iptables ruleset with `sudo iptables -S` and `sudo iptables -L`.

Let's take a look at the iptables commands!

Saving Rules

Iptables rules are ephemeral, which means they need to be manually saved for them to persist after a reboot.

Ubuntu

On Ubuntu, the easiest way to save iptables rules, so they will survive a reboot, is to use the `iptables-persistent` package. Install it with apt-get like

```
$ sudo apt-get install iptables-persistent
```

During the installation, you will asked if you want to save your current firewall rules.

If you update your firewall rules and want to save the changes, run this command:

```
$ sudo invoke-rc.d iptables-persistent save
```

CentOS 6 and Older

On CentOS 6 and older—CentOS 7 uses FirewallD by default—you can use the `iptables` init script to save your iptables rules:

```
$ sudo service iptables save
```

This will save your current iptables rules to the `/etc/sysconfig/iptables` file.

Listing and Deleting Rules

If you want to learn how to list and delete iptables rules, check out this tutorial: How To List and Delete Iptables Firewall Rules.

Generally Useful Rules

This section includes a variety of iptables commands that will create rules that are generally useful on most servers.

Allow Loopback Connections

The **loopback** interface, also referred to as `lo`, is what a computer uses to forward network connections to itself. For example, if you run `ping local` loopback interface is also used if you configure your application server to connect to a database server with a "localhost" address. As such, you will w

To accept all traffic on your loopback interface, run these commands:

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

Allow Established and Related Incoming Connections

As network traffic generally needs to be two-way—incoming and outgoing—to work properly, it is typical to create a firewall rule that allows **establishe** outgoing connections initiated by the server itself. This command will allow that:

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

Allow Established Outgoing Connections

You may want to allow outgoing traffic of all **established** connections, which are typically the response to legitimate incoming connections. This comm

```
$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Internal to External

Assuming `eth0` is your external network, and `eth1` is your internal network, this will allow your internal to access the external:

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

Drop Invalid Packets

Some network traffic packets get marked as **invalid**. Sometimes it can be useful to log this type of packet but often it is fine to drop them. Do so with t

```
$ sudo iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

Block an IP Address

To block network connections that originate from a specific IP address, `15.15.15.51` for example, run this command:

```
$ sudo iptables -A INPUT -s 15.15.15.51 -j DROP
```

In this example, `-s 15.15.15.51` specifies a **source** IP address of "15.15.15.51". The source IP address can be specified in any firewall rule, includir

If you want to **reject** the connection instead, which will respond to the connection request with a "connection refused" error, replace "DROP" with "RE

```
$ sudo iptables -A INPUT -s 15.15.15.51 -j REJECT
```

Block Connections to a Network Interface

To block connections from a specific IP address, e.g. `15.15.15.51`, to a specific network interface, e.g. `eth0`, use this command:

```
$ iptables -A INPUT -i eth0 -s 15.15.15.51 -j DROP
```

This is the same as the previous example, with the addition of `-i eth0`. The network interface can be specified in any firewall rule, and is a great way

Service: SSH

If you're using a cloud server, you will probably want to allow incoming SSH connections (port 22) so you can connect to and manage your server. Thi

Allow All Incoming SSH

To allow all incoming SSH connections run these commands:

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

The second command, which allows the outgoing traffic of **established** SSH connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow Incoming SSH from Specific IP address or subnet

To allow incoming SSH connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire `15.15.15.0`

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** SSH connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow Outgoing SSH

If your firewall `OUTPUT` policy is not set to `ACCEPT`, and you want to allow outgoing SSH connections—your server initiating an SSH connection to ano

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

Allow Incoming Rsync from Specific IP Address or Subnet

Rsync, which runs on port 873, can be used to transfer files from one computer to another.

To allow incoming rsync connections from a specific IP address or subnet, specify the source IP address and the destination port. For example, if you run these commands:

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 873 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 873 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** rsync connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`

Service: Web Server

Web servers, such as Apache and Nginx, typically listen for requests on port 80 and 443 for HTTP and HTTPS connections, respectively. If your defau that will allow your server to respond to those requests.

Allow All Incoming HTTP

To allow all incoming HTTP (port 80) connections run these commands:

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

The second command, which allows the outgoing traffic of **established** HTTP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`

Allow All Incoming HTTPS

To allow all incoming HTTPS (port 443) connections run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
```

```
$ sudo iptables -A OUTPUT -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** HTTP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`

Allow All Incoming HTTP and HTTPS

If you want to allow both HTTP and HTTPS traffic, you can use the **multiport** module to create a rule that allows both ports. To allow all incoming HTT

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

The second command, which allows the outgoing traffic of **established** HTTP and HTTPS connections, is only necessary if the `OUTPUT` policy is not s

Service: MySQL

MySQL listens for client connections on port 3306. If your MySQL database server is being used by a client on a remote server, you need to be sure t

Allow MySQL from Specific IP Address or Subnet

To allow incoming MySQL connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire `15.15.15`

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** MySQL connections, is only necessary if the `OUTPUT` policy is not set to `ACCE`

Allow MySQL to Specific Network Interface

To allow MySQL connections to a specific network interface—say you have a private network interface `eth1`, for example—use these commands:

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

The second command, which allows the outgoing traffic of **established** MySQL connections, is only necessary if the `OUTPUT` policy is not set to `ACCE`

Service: PostgreSQL

PostgreSQL listens for client connections on port 5432. If your PostgreSQL database server is being used by a client on a remote server, you need to

PostgreSQL from Specific IP Address or Subnet

To allow incoming PostgreSQL connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire `15.1`

```
$ sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 5432 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 5432 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** PostgreSQL connections, is only necessary if the `OUTPUT` policy is not set to `A`

Allow PostgreSQL to Specific Network Interface

To allow PostgreSQL connections to a specific network interface—say you have a private network interface `eth1`, for example—use these commands

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

The second command, which allows the outgoing traffic of **established** PostgreSQL connections, is only necessary if the `OUTPUT` policy is not set to `A`

Service: Mail

Mail servers, such as Sendmail and Postfix, listen on a variety of ports depending on the protocols being used for mail delivery. If you are running a m
types of traffic. We will also show you how to create a rule to block outgoing SMTP mail.

Block Outgoing SMTP Mail

If your server shouldn't be sending outgoing mail, you may want to block that kind of traffic. To block outgoing SMTP mail, which uses port 25, run this

```
$ sudo iptables -A OUTPUT -p tcp --dport 25 -j REJECT
```

This configures iptables to **reject** all outgoing traffic on port 25. If you need to reject a different service by its port number, instead of port 25, simply re

Allow All Incoming SMTP

To allow your server to respond to SMTP connections, port 25, run these commands:

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

The second command, which allows the outgoing traffic of **established** SMTP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEP`

> **Note:** It is common for SMTP servers to use port 587 for outbound mail.

Allow All Incoming IMAP

To allow your server to respond to IMAP connections, port 143, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 143 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 143 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** IMAP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`

Allow All Incoming IMAPS

To allow your server to respond to IMAPS connections, port 993, run these commands:

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

The second command, which allows the outgoing traffic of **established** IMAPS connections, is only necessary if the `OUTPUT` policy is not set to `ACCEP`

Allow All Incoming POP3

To allow your server to respond to POP3 connections, port 110, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 110 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 110 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

The second command, which allows the outgoing traffic of **established** POP3 connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`

Allow All Incoming POP3S

To allow your server to respond to POP3S connections, port 995, run these commands:

```
$ sudo iptables -A INPUT -p tcp --dport 995 -m conntrack --ctstate NEW,ESTABL
$ sudo iptables -A OUTPUT -p tcp --sport 995 -m conntrack --ctstate ESTABLISH
```

The second command, which allows the outgoing traffic of **established** POP3S connections, is only necessary if the `OUTPUT` policy is not set to `ACCE`

Conclusion

That should cover many of the commands that are commonly used when configuring an iptables firewall. Of course, iptables is a very flexible tool so f specific needs if they aren't covered here.

If you're looking for help determining how your firewall should be set up, check out this tutorial: How To Choose an Effective Firewall Policy to Secure

Good luck!