

# Linux Firewalls using IPTables

Contributed by Administrator  
Monday, 08 March 2010

## Linux Firewalls using IPTables

This beginner article provides details with regard to the basics of setting up a Linux firewall using the iptables tool. It's important to note that configuring firewalls is slightly different depending on which flavour you use. If you are just starting out and need to enable a firewall on your Linux system, I suggest you try a basic tool such as lokkit (which is available in major flavours including Fedora/RedHat and Ubuntu). It is a very simple tool that walks you through a configuration. To use this, run the command: `gnome-lokkit`. For more information or to download the tool, you may visit: <http://www.linux.org.uk/apps/lokkit.shtml>. You may also be interested in a graphical tool, rather than using the command line to make changes to the iptables. If you prefer a graphical tool, there are many available on the Internet. Start with a visit to [freshmeat.net](http://freshmeat.net). Introduction to iptables For those who don't know or are not aware, iptables is the Linux tool that controls network packets, allowing you to perform very fine grained control of network related transactions through a set of rules. The tool itself has been around for quite a while, and is based on Rusty Russell's excellent work. But before you start creating rules using the iptables command, you also need to be aware that any rules you create will be lost if a server restart occurs. For this reason, most server administrators apply the set of iptable commands they use into a bash script that runs each time the server is restarted. With some flavours, you can also run a command set like this: `service iptables save`. This ensures that your configuration is saved and is automatically loaded upon bootup. Getting started To use iptables, you apply rules to network packets that are either inbound (INPUT), outbound (OUTPUT), or being forwarded through your server (FORWARD). This is very important to understand. To view the rules that are currently applied to your server, type the command: `iptables -L`. Basics of iptables Creating properly functioning firewall rules is dependent upon your knowledge of what your server is doing. For a secure server, it is best to establish rules that will DENY all incoming traffic. Once you do so, then you can make explicit rules that only allow exceptions such as for port 80 requests. This is far more comprehensive than trying to filter out things you want to block, because you could miss something important such as one open port that will be used to attack your server. It is also beneficial to use iptables in conjunction with a hardware firewall, since this provides several levels of security and reduces the possibility that you missed something in your configuration. Remember, that you can get all syntax details using the command: `man iptables`. Using firewall rules Let's look at three sample rules and their core parts. Remember that some of the longer command lines wrap on the column, so make sure you type the entire command, not just the single line. `iptables -P INPUT DROP` This rule is very easy to understand and highly secure. It initiates the iptables tool, then sets a Policy (-P) for all inbound (INPUT) packets. The policy is to drop them all (DROP). Nice and secure. Note that you can only apply a policy to built in commands for iptables. `iptables -A INPUT -i lo -j ACCEPT` This rule is a bit more useful, in that it allows network traffic to occur on your local interface. Note that the append option (-A) is used, because this is not part of the built in policies. The new rule is appended to all inbound (INPUT) packets that are going to the interface (-i) local (lo). The rule is to allow all these packets (-j ACCEPT). Often if you run into problems with configuring installed applications, it begins with ensuring you allow local host connectivity, as shown. `iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT` This command also appends (-A) a rule to all inbound (INPUT) packets that are coming through the ethernet card interface (-i eth0). But it only applies to packets that use the TCP protocol (-p tcp). It is specific to any such packets going to the designated port 80 (--dport 80), and is set to allow them to pass (-j ACCEPT). If you put all three of these rules together into one script you have:

```
- a server that will block every inbound connection
- but it allows for internal host connectivity through local
- while it also allows port 80 tcp requests that are inbound to also go through

Notice that you can also get finer
grain control by designating a specific IP. For example:
iptables -A INPUT -d 196.1.1.2 -i eth0 -p tcp --dport 80 -j
ACCEPT

Sample Script
The options are truly limitless, but you need to be careful. Below is a basic script you
may find useful for beginning your firewall rule settings. When copying this script remember the longer commands are
broken into several lines:
#!/bin/bash
# # iptables firewall settings for linux server
# ### DEFAULT
POLICY iptables -P INPUT DROP iptables -P OUTPUT ACCEPT iptables -P FORWARD DROP ###
ESSENTIAL RULES # Allow internal host packets on local interface iptables -A INPUT -i lo -j ACCEPT ###
PROTOCOL LEVEL RULES # Allow PORT 80 TCP packets on ethernet interface iptables -A INPUT -i eth0 -p tcp --
dport 80 -j ACCEPT
# LOG ALL OTHER PACKETS # Logging for any failed packets for troubleshooting use iptables -A INPUT -j
LOG --log-prefix "INPUT: "
```

Hopefully this brief introduction to firewalls helps you identify key rules that can make a more secure server. In the next issue, I will share the list of the most essential system administrator commands.

This brief opinion piece should not be construed as factual information, and only contains the opinions and personal

experiences of the author at the time of publication. Microsoft, Microsoft Windows are trademarks or registered trademarks of Microsoft Corporation both in the United States and Internationally. All other trademarks or registered trademarks in this opinion piece belong to their respective owners.