

Iptables Basics

By Prince_Kenshi

I'm sure many of you have been wondering how to use iptables to set up a basic firewall. I was wondering the same thing for a long time until I recently figured it out. I'll try to explain the basics to at least get you started.

First you need to know how the firewall treats packets leaving, entering, or passing through your computer. Basically there is a chain for each of these. Any packet entering your computer goes through the INPUT chain. Any packet that your computer sends out to the network goes through the OUTPUT chain. Any packet that your computer picks up on one network and sends to another goes through the FORWARD chain. The chains are half of the logic behind iptables themselves.

Now the way that iptables works is that you set up certain rules in each of these chains that decide what happens to packets of data that pass through them. For instance, if your computer was to send out a packet to www.yahoo.com to request an HTML page, it would first pass through the OUTPUT chain. The kernel would look through the rules in the chain and see if any of them match. The first one that matches will decide the outcome of that packet. If none of the rules match, then the policy of the whole chain will be the final decision maker. Then whatever reply Yahoo! sent back would pass through the INPUT chain. It's no more complicated than that.

Now that we have the basics out of the way, we can start working on putting all this to practical use. There are a lot of different letters to memorize when using iptables and you'll probably have to peek at the man page often to remind yourself of a certain one. Now let's start with manipulation of certain IP addresses. Suppose you wanted to block all packets coming from 200.200.200.1. First of all, -s is used to specify a source IP or DNS name. So from that, to refer to traffic coming from this address, we'd use this:

```
iptables -s 200.200.200.1
```

But that doesn't tell what to do with the packets. The -j option is used to specify what happens to the packet. The most common three are ACCEPT, DENY, and DROP. Now you can probably figure out what ACCEPT does and it's not what we want. DENY sends a message back that this computer isn't accepting connections. DROP just totally ignores the packet. If we're really suspicious about this certain IP address, we'd probably prefer DROP over DENY. So here is the command with the result:

```
iptables -s 200.200.200.1 -j DROP
```

But the computer still won't understand this. There's one more thing we need to add and that's which chain it goes on. You use -A for this. It just appends the rule to the end of whichever chain you specify. Since we want to keep the computer from talking to us, we'd put it on INPUT. So here's the entire command:

```
iptables -A INPUT -s 200.200.200.1 -j DROP
```

This single command would ignore everything coming from 200.200.200.1 (with exceptions, but we'll get into that later). The order of the options doesn't matter; the -j DROP could go before -s 200.200.200.1. I just like to put the outcome part at the end of the command. Ok, we're now capable of ignoring a certain computer on a network. If you wanted to keep your computer from talking to it, you'd simply change INPUT to OUTPUT and change the -s to -d for destination. Now that's not too hard, is it?

Iptables Basics

By Prince_Kenshi

So what if we only wanted to ignore telnet requests from this computer? Well that's not very hard either. You might know that port 23 is for telnet, but you can just use the word telnet if you like. There are at least 3 protocols that can be specified: TCP, UDP, and ICMP. Telnet, like most services, runs on TCP so we're going with it. The -p option specifies the protocol. But TCP doesn't tell it everything; telnet is only a specific protocol used on the larger protocol of TCP. After we specify that the protocol is TCP, we can use --destination-port to denote the port that they're trying to contact us on. Make sure you don't get source and destination ports mixed up. Remember, the client can run on any port, it's the server that will be running the service on port 23. Any time you want to block out a certain service, you'll use --destination-port. The opposite is --source-port in case you need it. So let's put this all together. This should be the command that accomplishes what we want:

```
iptables -A INPUT -s 200.200.200.1 -p tcp --destination-port telnet -j DROP
```

And there you go. If you wanted to specify a range of IP's, you could use 200.200.200.0/24. This would specify any IP that matched 200.200.200.*. Now it's time to fry some bigger fish. Let's say that, like me, you have a local area network and then you have a connection to the internet. We're going to also say that the LAN is eth0 while the internet connection is called ppp0. Now suppose we wanted to allow telnet to run as a service to computers on the LAN but not on the insecure internet. Well there is an easy way to do this. We can use -i for the input interface and -o for the output interface. You could always block it on the OUTPUT chain, but we'd rather block it on the INPUT so that the telnet daemon never even sees the request. Therefore we'll use -i. This should set up just the rule:

```
iptables -A INPUT -p tcp --destination-port telnet -i ppp0 -j DROP
```

So this should close off the port to anyone on the internet yet kept it open to the LAN. Now before we go on to more intense stuff, I'd like to briefly explain other ways to manipulate rules. The -A option appends a rule to the end of the list, meaning any matching rule before it will have say before this one does. If we wanted to put a rule before the end of the chain, we use -I for insert. This will put the rule in a numerical location in the chain. For example, if we wanted to put it at the top of the INPUT chain, we'd use "-I INPUT 1" along with the rest of the command. Just change the 1 to whatever place you want it to be in. Now let's say we wanted to replace whatever rule was already in that location. Just use -R to replace a rule. It has the same syntax as -I and works the same way except that it deletes the rule at that position rather than bumping everything down. And finally, if you just want to delete a rule, use -D. This also has a similar syntax but you can either use a number for the rule or type out all the options that you would if you created the rule. The number method is usually the optimal choice. There are two more simple options to learn though. -L lists all the rules set so far. This is obviously helpful when you forget where you're at. AND -F flushes a certain chain. (It removes all of the rules on the chain.) If you don't specify a chain, it will basically flush everything.

Well let's get a bit more advanced. We know that these packets use a certain protocol, and if that protocol is TCP, then it also uses a certain port. Now you might be compelled to just close all ports to incoming traffic, but remember, after your computer talks to another computer, that computer must talk back. If you close all of your incoming ports, you'll essentially render your connection useless. And for most non-service programs, you can't predict which port they're going to be communicating on. But there's still a way. Whenever two computers are talking over a TCP connection, that connection must first be initialized. This is the job of a SYN packet. A SYN packet simply tells the other computer that it's ready to talk. Now only the computer requesting the service sends a SYN packet. So if you only block incoming SYN packets, it stops other computers from opening services on your computer but doesn't stop you from communicating with them. It roughly makes your computer ignore anything that it didn't speak to first. It's mean but it gets the job done. Well the option for this is --syn after

Iptables Basics

By Prince_Kenshi

you've specified the TCP protocol. So to make a rule that would block all incoming connections on only the internet:

```
iptables -A INPUT -i ppp0 -p tcp --syn -j DROP
```

That's a likely rule that you'll be using unless you have a web service running. If you want to leave one port open, for example 80 (HTTP), there's a simple way to do this too. As with many programming languages, an exclamation mark means not. For instance, if you wanted to block all SYN packets on all ports except 80, I believe it would look something like this:

```
iptables -A INPUT -i ppp0 -p tcp --syn --destination-port ! 80 -j DROP
```

It's somewhat complicated but it's not so hard to comprehend. There's one last thing I'd like to cover and that's changing the policy for a chain. The chains INPUT and OUTPUT are usually set to ACCEPT by default and FORWARD is set to DENY. Well if you want to use this computer as a router, you would probably want to set the FORWARD policy to ACCEPT. How do we do this you ask? Well it's really very simple. All you have to do is use the -P option. Just follow it by the chain name and the new policy and you have it made. To change the FORWARD chain to an ACCEPT policy, we'd do this:

```
iptables -P FORWARD ACCEPT
```

Nothing to it, huh? This is really just the basics of iptables. It should help you set up a limited firewall but there's still a lot more that I couldn't talk about. You can look at the man page "man iptables" to learn more of the options (or refresh your memory when you forget). You can find more advanced documents if you want to learn some of the more advanced features of iptables. At the time of this writing, iptables documents are somewhat rare because the technology is new but they should be springing up soon. Good luck.