**INSTALLING, ENABLING AND CONFIGURING IPTABLES**

Managing iptables requires root-level access so it’s best to either log in directly as root, or elevate to a root shell.

If you want to use iptables on CentOS 7, the first thing we need to do is disable firewalld . We can do that with the following commands:

|  |  |
| --- | --- |
| 1  2  3 | [root@vpscheap-blog ~]# systemctl stop firewalld    [root@vpscheap-blog ~]# systemctl mask firewalld |

To make sure it’s actually disabled, we can use:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | [root@vpscheap-blog ~]# systemctl status firewalld    ● firewalld.service    Loaded: masked (/dev/null)    Active: inactive (dead)    Jul 29 13:57:47 vpscheap-blog systemd[1]: Stopped firewalld - dynamic firewall daemon. |

If iptables isn’t installed on your system, we can use yum to install it:

|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# yum install iptables-services -y |

And after it’s installed, we’ll want to set it to start on boot, and start up the service:



|  |  |
| --- | --- |
| 1  2  3 | [root@vpscheap-blog ~]# systemctl enable iptables    [root@vpscheap-blog ~]# systemctl start iptables |

Now, to check the rules and default policies, we can use iptables -L :



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | [root@vpscheap-blog ~]# iptables -L    Chain INPUT (policy ACCEPT)    target prot opt source destination    ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED    ACCEPT icmp -- anywhere anywhere    ACCEPT all -- anywhere anywhere    ACCEPT tcp -- anywhere anywhere state NEW tcp dpt:ssh    REJECT all -- anywhere anywhere reject-with icmp-host-prohibited    Chain FORWARD (policy ACCEPT)    target prot opt source destination    REJECT all -- anywhere anywhere reject-with icmp-host-prohibited    Chain OUTPUT (policy ACCEPT)    target prot opt source destination |

We can see here our default policies for all three chains are set to ACCEPT, and that we do have some rules in place, mainly for the INPUT chain.

We’ll cover a basic iptables setup from here on, but it’s strongly advised to get acquainted will all the various options iptables has (man pages are your best friend!). We won’t get into too much detail on what the various options do, but the commands below will give you a good starting point which you can then further tweak to your specific needs.

First we’ll set the default policy of the INPUT chain to ACCEPT, so we don’t get locked out of the server, as we’ll flush (clear) the default ruleset:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -P INPUT ACCEPT |

Next, we’ll flush the ruleset with:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -F |

We’ll append (-A) a rule to the INPUT chain, for our localhost interface (-i lo), to allow all connections (-j ACCEPT), as that’s usually needed for many applications to work properly:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -A INPUT -i lo -j ACCEPT |

We can then add a rule that will load up a *conntrack*module (-m conntrack), which will inspect the state of the packet and determine if it’s NEW, ESTABLISHED or RELATED. NEW refers to incoming packets that are new incoming connections that weren’t initiated by the host system. ESTABLISHED and RELATED refers to incoming packets that are part of an already established connection or related to and already established connection.  
This rule will allow only the latter 2:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT |

Alternatively, we can use the *state*module:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT |

We’ll then allow all traffic on our ssh, http and https ports:



|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@vpscheap-blog ~]# iptables -A INPUT -p tcp --dport ssh -j ACCEPT    [root@vpscheap-blog ~]# iptables -A INPUT -p tcp --dport 80 -j ACCEPT    [root@vpscheap-blog ~]# iptables -A INPUT -p tcp --dport 443 -j ACCEPT |

Feel free to add any other rules such as these for any other ports you need.

As we now have our basic rules in place, that will allow all traffic on localhost interface, along with ssh and http/s traffic, we can set the default policy of the INPUT chain to DROP:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -P INPUT DROP |

We’ll DROP all traffic on the FORWARD chain as we generally don’t want our server to act as a proxy, and will ALLOW all outbound traffic since we trust it’s source:



|  |  |
| --- | --- |
| 1  2  3 | [root@vpscheap-blog ~]# iptables -P FORWARD DROP    [root@vpscheap-blog ~]# iptables -P OUTPUT ACCEPT |

As it’s better to be safe than sorry, we can check our new rules at this point, with some additional verbosity:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -L -v |

The last thing to do is to actually save these rules, so they get loaded on every reboot. We can do so in a couple of ways:



|  |  |
| --- | --- |
| 1  2  3 | [root@vpscheap-blog ~]# /sbin/service iptables save    iptables: Saving firewall rules to /etc/sysconfig/iptables:[ OK ] |

or



|  |  |
| --- | --- |
| 1  2  3 | [root@vpscheap-blog ~]# service iptables save    iptables: Saving firewall rules to /etc/sysconfig/iptables:[ OK ] |

And of course, we should restart the firewall at this point:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# systemctl restart iptables.service |

You should do these 2 steps each and every time you modify your ruleset!

At this point we have a pretty basic, but working, firewall setup. Since our default policy on INPUT chain is now set to DROP, if we want to allow (or whitelist) a specific IP, we can use something like:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -A INPUT -s 1.2.3.4 -j ACCEPT |

Of, course, if your default policy is set to ACCEPT, and you want to block an IP, just replace ACCEPT with DROP in that rule.

Remember that rules are processed in order. If you have a DROP rule somewhere in your ruleset, and you want to ACCEPT some connection that would otherwise match the DROP rule, you’ll have to add it before that rule, so -A (append) won’t work. In that case, we can use -I (insert) option, like so:



|  |  |
| --- | --- |
| 1 | [root@vpscheap-blog ~]# iptables -I INPUT 1 -i lo -j ACCEPT |

We’d have to insert this particular rule if we didn’t already add it previously. *Notice how the -I option needs two arguments, the first is for the chain (INPUT), the other is the position where the rule will be inserted (in this case it would be the first rule).*

**CONCLUSION**

Hopefully, this article illustrated the power and flexibility iptables offers. You can mix and match all the various flags to set it up just the way you need it. There are many frontends to iptables on the market, both graphical and command-line (in my work I often use ConfigServer Firewall and have only positive experience with it so I highly recommend it), but few of them offer a granular approach to setting up your ruleset like iptables does.

**BEFORE YOU LEAVE…**

Would you be interested in hosting your CentOS 7 server with the best price-quality ratio on the market? Have a look at one of our [cheap VPS hosting](https://www.vpscheap.net/pricing.aspx) offers.