Linux IPTables: How to Add Firewall Rules (With Allow SSH Example)

by Ramesh Natarajan on February 14, 2011

This article explains how to add iptables firewall rules using the "iptables -A" (append) command.

"-A" is for append. If it makes it easier for you to remember "-A" as add-rule (instead of append-rule), it is OK. But, keep in mind that "-A" adds the rule at the end of the chain.

Again, it is very important to remember that -A adds the rule at the end.

Typically the last rule will be to drop all packets. If you already have a rule to drop all packets, and if you try to use "-A" from the command-line to create new rule, you will end-up adding the new rule after the current "drop all packets" rule, which will make your new rule pretty much useless.

Once you've mastered the iptables, and when you are implementing it on production, you should use a shell script, where you use -A command to add all the rules. In that shell script, your last line should always be "drop all packets" rule. When you want to add any new rules, modify that shell script and add your new rules above the "drop all packets" rule.

Syntax:

```
iptables -A chain firewall-rule
```

- -A chain – Specify the chain where the rule should be appended. For example, use INPUT chain for incoming packets, and OUTPUT for outgoing packets.
- firewall-rule – Various parameters makes up the firewall rule.

If you don't know what chain means, you better read about iptables fundamentals first.

Firewall Rule Parameters

The following parameters are available for all kinds of firewall rules.

-p is for protocol

- Indicates the protocol for the rule.
- Possible values are tcp, udp, icmp
- Use "all" to allow all protocols. When you don't specify -p, by default "all" protocols will be used. It is not a good practice to use "all", and always specify a protocol.
- Use either the name (for example: tcp), or the number (for example: 6 for tcp) for protocol.

- /etc/protocols file contains all allowed protocol name and number.
- You an also use –protocol

-s is for source

- Indicates the source of the packet.
- This can be ip address, or network address, or hostname
- For example: -s 192.168.1.101 indicates a specific ip address
- For network mask use /mask. For example: "-s 192.168.1.0/24" represents a network mask of 255.255.255.0 for that network. This matches 192.168.1.x network.
- When you don't specify a source, it matches all source.
- You can also use –src or –source

-d is for destination

- Indicates the destination of the packet.
- This is same as "-s" (except this represents destination host, or ip-address, or network)
- You can also use –dst or –destination

-j is target

- j stands for "jump to target"
- This specifies what needs to happen to the packet that matches this firewall rule.
- Possible values are ACCEPT, DROP, QUEUE, RETURN
- You can also specify other user defined chain as target value.

-i is for in interface

- i stands for "input interface"
- You might over look this and assume that "-i" is for interface. Please note that both -i and -o are for interfaces. However, -i for input interface and -o for output interface.
- Indicates the interface through which the incoming packets are coming through the INPUT, FORWARD, and PREROUTING chain.
- For example: -i eth0 indicates that this rule should consider the incoming packets coming through the interface eth0.
- If you don't specify -i option, all available interfaces on the system will be considered for input packets.
- You can also use –in-interface

-o is for out interface

- o stands for "output interface"
- Indicates the interface through which the outgoing packets are sent through the INPUT, FORWARD, and PREROUTING chain.
- If you don't specify -o option, all available interfaces on the system will be considered for output packets.
- You can also use –out-interface

Additional Options for Firewall Parameters

Some of the above firewall parameters in turn has it's own options that can be passed along with them. Following are some of the most common options.

To use these parameter options, you should specify the corresponding parameter in the firewall rule. For example, to use "–sport" option, you should've specified "-p tcp" (or "-p udp") parameter in your firewall rule.

Note: All of these options have two dashes in front of them. For example, there are two hyphens in front of sport.

–sport is for source port (for -p tcp, or -p udp)

- By default all source ports are matched.
- You can specify either the port number or the name. For example, to use SSH port in your firewall rule, use either "–sport 22" or "–sport ssh".
- /etc/services file contains all allowed port name and number.
- Using port number in the rule is better (for performance) than using port name.
- To match range of ports, use colon. For example, 22:100 matches port number from 22 until 100.
- You can also use –source-port

–dport is for destination port (for -p tcp, or -p udp)

- Everything is same as –sport, except this is for destination ports.
- You can also use –destination-port

–tcp-flags is for TCP flags (for -p tcp)

- This can contain multiple values separated by comma.
- Possible values are: SYN, ACK, FIN, RST, URG, PSH. You can also use ALL or NONE

–icmp-type is for ICMP Type (for -p icmp)

- When you use icmp protocol "-p icmp", you can also specify the ICMP type using "–icmp-type" parameter.
- For example: use "–icmp-type 0" for "Echo Reply", and "–icmp-type 8" for "Echo".

Example Firewall Rule to Allow Incoming SSH Connections

Now that you understand various parameters (and it's options) of firewall rule, let us build a sample firewall rule.

In this example, let us allow only the incoming SSH connection to the server. All other connections will be blocked (including ping).

WARNING: Playing with firewall rules might render your system inaccessible. If you don't know what you are doing, you might lock yourself (and everybody else) out of the system. So, do all your learning only on a test system that is not used by anybody, and you have access to the console to restart the iptables, if you get locked out.

1. Delete Existing Rules

If you already have some iptables rules, take a backup before delete the existing rules.

Delete all the existing rules and allow the firewall to accept everything. Use iptables flush as we discussed earlier to clean-up all your existing rules and start from scratch.

Test to make sure you are able to ssh and ping this server from outside.

When we are done with this example, you'll only be able to SSH to this server. You'll not be able to ping this server from outside.

2. Allow only SSH

Allow only the incoming SSH connection to this server. You can ssh to this server from anywhere.

```
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
```

The above iptables command has the following 4 components.

- "-A INPUT" – This indicates that we are appending a new rule (or adding) to the INPUT chain. So, this rule is for incoming traffic.
- "-i eth0" – Incoming packets through the interface eth0 will be checked against this rule.
- "-p tcp –dport 22" – This rule is for TCP packets. This has one tcp option called "–dport 22", which indicates that the destination port for this rule on the server is 22 (which is ssh).
- "-j ACCEPT" – Jump to accept, which just ACCEPTS the packet.

In simple terms the above rule can be stated as: All incoming packets through eth0 for ssh will be accepted.

3. Drop all Other Packets

Once you've specified your custom rules to accept packets, you should also have a default rule to drop any other packets.

This should be your last rule in the INPUT chain.

To drop all incoming packets, do the following.

```
iptables -A INPUT -j DROP
```

4. View the SSH rule and Test

To view the current iptables firewall rules, use "iptables -L" command.

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
DROP       all  --  anywhere             anywhere
```

As you see from the above output, it has the following two rules in sequence.

- Accept all incoming ssh connections
- Drop all other packets.

Instead of adding the firewall rules from the command line, it might be better to create a shell script that contains your rules as shown below.

```
# vi iptables.sh
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -j DROP

# sh -x iptables.sh
+ iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
+ iptables -A INPUT -j DROP

# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
DROP       all  --  anywhere             anywhere
```

Similar to iptables append/add command, there are few other commands available for iptables. I'll cover them in the upcoming articles in the iptables series. I'll also provide several practical firewall rule examples that will be helpful in real life scenarios.

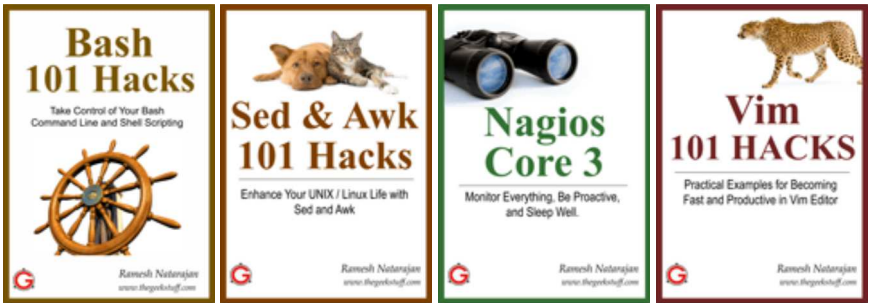
Previous articles in the iptables series:

- Linux Firewall Tutorial: IPTables Tables, Chains, Rules Fundamentals
- IPTables Flush: Delete / Remove All Rules On RedHat and CentOS Linux

If you enjoyed this article, you might also like..

1. 50 Linux Sysadmin Tutorials
2. 50 Most Frequently Used Linux Commands (With Examples)
3. Top 25 Best Linux Performance Monitoring and Debugging Tools
4. Mommy, I found it! – 15 Practical Linux Find Command Examples
5. Linux 101 Hacks 2nd Edition eBook Free

- Awk Introduction – 7 Awk Print Examples
- Advanced Sed Substitution Examples
- 8 Essential Vim Editor Navigation Fundamentals
- 25 Most Frequently Used Linux IPTables Rules Examples
- Turbocharge PuTTY with 12 Powerful Add-Ons



Tagged as: CentOS Add Firewall Rule, IPTables, IPTables Add Rule

{ 25 comments… add one }

- Karn Chandra February 14, 2011, 4:41 am

  Hi Ramesh,

  Thanks for your lucid explanation! Could you also please link your past IPTables posts (those are part of this ongoing tutorial) within this post? It would be really helpful to view them in context and move around them swiftly. Presently one has to use the search box.

  Once again thanks and keep up the good work!

  Link
- Karn Chandra February 14, 2011, 4:43 am

  Sorry just noticed they are already linked at the end of the post. Please ignore the comment above.

  Link
- Kuldeep February 14, 2011, 6:11 am

  very gud !!!!!

  Thanx alot for very informative article…..