- Login to Edit

Ubuntu Documentation > Community Documentation > IptablesHowTo

IptablesHowTo

Sommaire

Basic iptables howto

Iptables is a firewall, installed by default on all official Ubuntu distributions (Ubuntu, Kubuntu, Xubuntu). When you install Ubuntu, iptables is there, but it allows all traffic by default. Ubuntu 8.04 Comes with ufw - a program for managing the iptables firewall easily.

There is a wealth of information available about iptables, but much of it is fairly complex, and if you want to do a few basic things, this How To is for you.

Basic Commands

Typing

```
# iptables -L
```

lists your current rules in iptables. If you have just set up your server, you will have no rules, and you should see

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Basic Iptables Options

Here are explanations for some of the iptables options you will see in this tutorial. Don't worry about understanding everything here now, but remember to come back and look at this list as you encounter new options later on.

- `-A` - Append this rule to a rule chain. Valid chains for what we're doing are INPUT, FORWARD and OUTPUT, but we mostly deal with INPUT in this tutorial, which affects only incoming traffic.
- `-L` - List the current filter rules.
- `-m conntrack` - Allow filter rules to match based on connection state. Permits the use of the `--ctstate` option.
- `--ctstate` - Define the list of states for the rule to match on. Valid states are:
  - NEW - The connection has not yet been seen.
  - RELATED - The connection is new, but is related to another connection already permitted.
  - ESTABLISHED - The connection is already established.
  - INVALID - The traffic couldn't be identified for some reason.
- `-m limit` - Require the rule to match only a limited number of times. Allows the use of the `--limit` option. Useful for limiting logging rules.
  - `--limit` - The maximum matching rate, given as a number followed by "/second", "/minute", "/hour", or "/day" depending on how often you want the rule to match. If this option is not used and `-m limit` is used, the default is "3/hour".
- `-p` - The connection protocol used.

- `--dport` - The destination port(s) required for this rule. A single port may be given, or a range may be given as `start:end`, which will match all ports from `start` to `end`, inclusive.
- `-j` - Jump to the specified target. By default, iptables allows four targets:
  - `ACCEPT` - Accept the packet and stop processing rules in this chain.
  - `REJECT` - Reject the packet and notify the sender that we did so, and stop processing rules in this chain.
  - `DROP` - Silently ignore the packet, and stop processing rules in this chain.
  - `LOG` - Log the packet, and continue processing more rules in this chain. Allows the use of the `--log-prefix` and `--log-level` options.
- `--log-prefix` - When logging, put this text before the log message. Use double quotes around the text to use.
- `--log-level` - Log using the specified syslog level. 7 is a good choice unless you specifically need something else.
- `-i` - Only match if the packet is coming in on the specified interface.
- `-I` - Inserts a rule. Takes two options, the chain to insert the rule into, and the rule number it should be.
  - `-I INPUT 5` would insert the rule into the INPUT chain and make it the 5th rule in the list.
- `-v` - Display more information in the output. Useful for if you have rules that look similar without using `-v`.
- `-s --source` - address[/mask] source specification
- `-d --destination` - address[/mask] destination specification
- `-o --out-interface` - output name[+] network interface name ([+] for wildcard)

Allowing Established Sessions

We can allow established sessions to receive traffic:

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

- The above rule has no spaces either side of the comma in ESTABLISHED,RELATED

If the line above doesn't work, you may be on a VPS that uses OpenVZ or doesn't have some kernel extensions installed. In that case, try this line instead:

```
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Allowing Incoming Traffic on Specific Ports

You could start by blocking traffic, but you might be working over SSH, where you would need to allow SSH before blocking everything else.

To allow incoming traffic on the default SSH port (22), you could tell iptables to allow all TCP traffic on that port to come in.

```
# iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

Referring back to the list above, you can see that this tells iptables:

- append this rule to the input chain (-A INPUT) so we look at incoming traffic
- check to see if it is TCP (-p tcp).
- if so, check to see if the input goes to the SSH port (--dport ssh).
- if so, accept the input (-j ACCEPT).

Lets check the rules: (only the first few lines shown, you will see more)

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

Now, let's allow all incoming web traffic

```
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Checking our rules, we have

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
```

We have specifically allowed tcp traffic to the ssh and web ports, but as we have not blocked anything, all traffic can still come in.

Blocking Traffic

Once a decision is made to accept a packet, no more rules affect it. As our rules allowing ssh and web traffic come first, as long as our rule to block all traffic comes after them, we can still accept the traffic we want. All we need to do is put the rule to block all traffic at the end.

```
# iptables -A INPUT -j DROP
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
DROP       all  --  anywhere             anywhere
```

Because we didn't specify an interface or a protocol, any traffic for any port on any interface is blocked, except for web and ssh.

Editing iptables

The only problem with our setup so far is that even the loopback port is blocked. We could have written the drop rule for just eth0 by specifying -i eth0, but we could also add a rule for the loopback. If we append this rule, it will come too late - after all the traffic has been dropped. We need to insert this rule before that. Since this is a lot of traffic, we'll insert it as the first rule so it's processed first.

```
# iptables -I INPUT 1 -i lo -j ACCEPT
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:www
DROP       all  --  anywhere             anywhere
```

The first and last lines look nearly the same, so we will list iptables in greater detail.

```
# iptables -L -v
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere
    0     0 ACCEPT     all  --  any    any     anywhere             anywhere            state
RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere            tcp dpt:ssh
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere            tcp dpt:www
    0     0 DROP       all  --  any    any     anywhere             anywhere
```

You can now see a lot more information. This rule is actually very important, since many programs use the loopback interface to communicate with each other. If you don't allow them to talk, you could break those programs!

Logging

In the above examples none of the traffic will be logged. If you would like to log dropped packets to syslog, this would be the quickest way:

```
# iptables -I INPUT 5 -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7
```

See Tips section for more ideas on logging.

Saving iptables

If you were to reboot your machine right now, your iptables configuration would disappear. Rather than type this each time you reboot, however, you can save the configuration, and have it start up automatically. To save the configuration, you can use `iptables-save` and `iptables-restore`.

Configuration on startup

WARNING: Iptables and Ubuntu:NetworkManager seem to have a conflict. However NetworkManager is still in Beta. If you are concerned enough about security to install a firewall you might not want to trust NetworkManager to manage it yet. Also note NetworkManager and iptables have opposite aims. Iptables aims to keep any questionable network traffic out. NetworkManager aims to keep you connected at all times. Therefore if you want security all the time, run iptables at boot time. If you want security some of the time then NetworkManager might be the right choice.

WARNING: If you use Ubuntu:NetworkManager (installed by default on Feisty and later) these steps will leave you unable to use Ubuntu:NetworkManager for the interfaces you modify. Please follow the steps in the next section instead.

NOTE: It appears on Hardy, Ubuntu:NetworkManager has an issue with properly on saving and restoring the iptable rules when using the method in the next section. Using this first method appears to work. If you find otherwise, please update this note.

Save your firewall rules to a file

```
# sudo bash -c "iptables-save > /etc/iptables.rules"
```

At this point you have several options. You can make changes to `/etc/network/interfaces` or add scripts to `/etc/network/if-pre-up.d/` and `/etc/network/if-post-down.d/` to achieve similar ends. The script solution allows for slightly more flexibility.

Solution #1 - /etc/network/interfaces

(NB: I followed the directions in this section and this disabled most (all?) of my scripts in /etc/rc2.d/: polipo, dnsmasq, bluetooth, etc. I recommend using ufw/gufw instead. ~gasull)

Modify the `/etc/network/interfaces` configuration file to apply the rules automatically. You will need to know the interface that you are using in order to apply the rules - if you do not know, you are probably using the interface eth0, although you should check with the following command first to see if there are any wireless cards:

```
$ iwconfig
```

If you get output similar to the following, then you do not have any wireless cards at all and your best bet is probably eth0.

```
$ iwconfig

lo        no wireless extensions.

eth0      no wireless extensions.

$
```

When you have found out the interface you are using, please open your `/etc/network/interfaces` file depending on what editor you want and/or what distribution you have:

Command line:

```
# nano /etc/network/interfaces
```

For Ubuntu and Xubuntu: type ALT+F2, then in the window that pops up, type:

```
gksudo gedit /etc/network/interfaces
```

and press Enter.

For Kubuntu: type ALT+F2, then in the window that pops up, type:

```
kdesu kate /etc/network/interfaces
```

and press enter.

When in the file, search for the interface you found, and at the end of the network related lines for that interface, add the line:

```
pre-up iptables-restore < /etc/iptables.rules
```

You can also prepare a set of down rules, save them into second file `/etc/iptables.downrules` and apply it automatically using the above steps:

```
post-down iptables-restore < /etc/iptables.downrules
```

A fully working example using both from above:

```
auto eth0
iface eth0 inet dhcp
  pre-up iptables-restore < /etc/iptables.rules
  post-down iptables-restore < /etc/iptables.downrules
```

You may also want to keep information from byte and packet counters.

```
iptables-save -c > /etc/iptables.rules
```

The above command will save the whole rule-set to a file called `/etc/iptables.rules` with byte and packet counters still intact.

Solution #2 /etc/network/if-pre-up.d and ../if-post-down.d

NOTE: This solution uses `iptables-save -c` to save the counters. Just remove the `-c` to only save the rules.

Alternatively you could add the `iptables-restore` and `iptables-save` to the `if-pre-up.d` and `if-post-down.d` directories in the `/etc/network` directory instead of modifying `/etc/network/interface` directly.

The script `/etc/network/if-pre-up.d/iptablesload` will contain:

```
#!/bin/sh
iptables-restore < /etc/iptables.rules
exit 0
```

and `/etc/network/if-post-down.d/iptablessave` will contain:

```
#!/bin/sh
iptables-save -c > /etc/iptables.rules
if [ -f /etc/iptables.downrules ]; then
   iptables-restore < /etc/iptables.downrules
fi
exit 0
```

Then be sure to give both scripts execute permissions:

```
# chmod +x /etc/network/if-post-down.d/iptablessave
```

```
# chmod +x /etc/network/if-pre-up.d/iptablesload
```

Configuration on Startup for NetworkManager

Ubuntu:NetworkManager includes the ability to run scripts when it activates or deactivates an interface. To save iptables rules on shutdown, and to restore them on startup, we are going to create such a script. To begin, press Alt+F2 and enter this command:

For Ubuntu:

```
$ gksudo gedit /etc/NetworkManager/dispatcher.d/01firewall
```

For Kubuntu:

```
kdesu kate /etc/NetworkManager/dispatcher.d/01firewall
```

Then, paste this script into your editor, save, and exit the editor.

```
if [ -x /usr/bin/logger ]; then
        LOGGER="/usr/bin/logger -s -p daemon.info -t FirewallHandler"
else
        LOGGER=echo
fi

case "$2" in
        up)
                if [ ! -r /etc/iptables.rules ]; then
                        ${LOGGER} "No iptables rules exist to restore."
                        return
                fi
                if [ ! -x /sbin/iptables-restore ]; then
                        ${LOGGER} "No program exists to restore iptables rules."
                        return
                fi
                ${LOGGER} "Restoring iptables rules"
                /sbin/iptables-restore -c < /etc/iptables.rules
                ;;
        down)
                if [ ! -x /sbin/iptables-save ]; then
                        ${LOGGER} "No program exists to save iptables rules."
                        return
                fi
                ${LOGGER} "Saving iptables rules."
                /sbin/iptables-save -c > /etc/iptables.rules
                ;;
        *)
                ;;
esac
```

Finally, we need to make sure Ubuntu:NetworkManager can execute this script. In a terminal window, enter this command:

```
# chmod +x /etc/NetworkManager/dispatcher.d/01firewall
```

Tips

If you manually edit iptables on a regular basis

The above steps go over how to setup your firewall rules and presume they will be relatively static (and for most people they should be). But if you do a lot of development work, you may want to have your iptables saved everytime you reboot. You could add a line like this one in `/etc/network/interfaces`:

```
  pre-up iptables-restore < /etc/iptables.rules
  post-down iptables-save > /etc/iptables.rules
```

The line "post-down iptables-save > /etc/iptables.rules" will save the rules to be used on the next boot.

Using iptables-save/restore to test rules

If you edit your iptables beyond this tutorial, you may want to use the `iptables-save` and `iptables-restore` feature to edit and test your rules. To do this open the rules file in your favorite text editor (in this example gedit).

```
$ sudo iptables-save > /etc/iptables.rules
$ gksudo gedit /etc/iptables.rules
```

You will have a file that appears similiar to (following the example above):

```
# Generated by iptables-save v1.3.1 on Sun Apr 23 06:19:53 2006
*filter
:INPUT ACCEPT [368:102354]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [92952:20764374]
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7
-A INPUT -j DROP
COMMIT
# Completed on Sun Apr 23 06:19:53 2006
```

Notice that these are iptables commands minus the `iptable` command. Feel free to edit this to file and save when complete. Then to test simply:

```
# iptables-restore < /etc/iptables.rules
```

NOTE: With iptables 1.4.1.1-1 and above, a script allow you to test your new rules without risking to brick your remote server. If you are applying the rules on a remote server, you should consider testing it with:

```
# iptables-apply /etc/iptables.rules
```

After testing, if you have not added the `iptables-save` command above to your `/etc/network/interfaces` remember not to lose your changes:

```
# iptables-save > /etc/iptables.rules
```

More detailed Logging

For further detail in your syslog you may want create an additional Chain. This will be a very brief example of my /etc/iptables.rules showing how I setup my iptables to log to syslog:

```
# Generated by iptables-save v1.3.1 on Sun Apr 23 05:32:09 2006
*filter
:INPUT ACCEPT [273:55355]
:FORWARD ACCEPT [0:0]
:LOGNDROP - [0:0]
:OUTPUT ACCEPT [92376:20668252]
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -j LOGNDROP
-A LOGNDROP -p tcp -m limit --limit 5/min -j LOG --log-prefix "Denied TCP: " --log-level 7
-A LOGNDROP -p udp -m limit --limit 5/min -j LOG --log-prefix "Denied UDP: " --log-level 7
-A LOGNDROP -p icmp -m limit --limit 5/min -j LOG --log-prefix "Denied ICMP: " --log-level 7
-A LOGNDROP -j DROP
COMMIT
# Completed on Sun Apr 23 05:32:09 2006
```

Note a new CHAIN called `LOGNDROP` at the top of the file. Also, the standard `DROP` at the bottom of the INPUT chain is replaced with `LOGNDROP` and add protocol descriptions so it makes sense looking at the log. Lastly we drop the traffic at the end of the `LOGNDROP` chain. The following gives some idea of what is happening:

- `--limit` sets the number of times to log the same rule to syslog

- `--log-prefix "Denied..."` adds a prefix to make finding in the syslog easier
- `--log-level 7` sets the syslog level to informational (see man syslog for more detail, but you can probably leave this)

Disabling the firewall

If you need to disable the firewall temporarily, you can flush all the rules using

```
# iptables -F
```

or create a script using text editor such as nano

```
# nano -w /root/fw.stop
```

```
echo "Stopping firewall and allowing everyone..."
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
```

Make sure you can execute the script

```
$ chmod +x /root/fw.stop
```

You can run the script

```
$ /root/fw.stop
```

Easy configuration via GUI

GUFW - Gufw is a graphical frontend to UFW (Uncomplicated Firewall).

A new user can use Firestarter (a gui), available in repositories (Synaptic or apt-get) to configure her/his iptable rules, without needing the command line knowledge. Please see the tutorial though... Configuration is easy, but may not be enough for the advanced user. However, it should be enough for the most home users... The (read:my) suggested outbound configuration is "restrictive", with whitelisting each connection type whenever you need it (port 80 for http, 443 for secure http -https-, 1863 for msn chat etc) from the "policy" tab within firestarter. You can also use it to see active connections from and to your computer... The firewall stays up once it is configured using the wizard. Dial-up users will have to specify it to start automatically on dial up in the wizard.

Homepage for firestarter: http://www.fs-security.com/ (again, available in repositories, no compiling required) Tutorial: http://www.fs-security.com/docs/tutorial.php

Please note that it conflicts with ufw.

Further Information

Iptables Tutorial

Iptables How To

Netfilter and Iptables Multilingual Documentation

Easy Firewall Generator for IPTables

For some reason the Ubuntu wiki page on Firestarter does not come up on searches, so here is a link :

- Ubuntu Wiki Firestarter
- Firestarter is a gui tool to help configure IP Tables.

Please note that it conflicts with ufw.

Credits

Thanks to Rusty Russell and his How-To, as much of this is based off that.

CategorySecurity

IptablesHowTo (dernière édition le 2010-09-29 12:49:18 par https://login.launchpad.net/+id/cAfxADx @ d54C0600F.access.telenet.be[84.192.96.15]:Jan Celis)

- Page History