

Iptables

From Noah.org

Contents

- 1 Linux iptables Basic Examples
 - 1.1 Block everything
 - 1.2 Wide Open Firewall
 - 1.3 Minimal emergency firewall
 - 1.4 Basic firewall
- 2 Load firewall on boot
 - 2.1 RedHat
 - 2.2 Ubuntu/Debian
 - 2.3 init.d script
- 3 Traffic shaping
- 4 Handy commands
 - 4.1 Block an annoying machine from reaching your server
 - 4.2 Show packet and byte counts

Linux iptables Basic Examples

The following are simple iptables firewalls for Linux. I use these as starter firewalls when I setup a machine... I don't like using iptables-restore. I prefer to simply script the iptables commands that I would otherwise type at the command line.

Most of these scripts start by reinitializing iptables, so you will lose any rules, chains, or accounting information that iptables knows about. For example, this deletes any policies, chains, and rules in place.

```
iptables -F
iptables -X
iptables -P INPUT ACCEPT # open up default policy on built-in chain
iptables -P OUTPUT ACCEPT # open up default policy on built-in chain
iptables -P FORWARD ACCEPT # open up default policy on built-in chain
iptables -F # delete all rules from all chains
iptables -X # delete all user chains (non built-in chains)
```

Block everything

This blocks everything. You will only be able to access the machine from the console. Don't do this if you are working remotely because your connection will instantly be dropped. Another way to do this would be to disable the network interface. The advantage of blocking everything with iptables instead of shutting down a network interface is that this leaves the kernel network layer still running. Running apps will not complain about the network being unavailable. This also blocks all network interfaces at once, so if you have a machine with multiple interfaces this will take care of them all.

```
#!/bin/sh
iptables -F
iptables -X
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

Wide Open Firewall

This opens up everything. It's the exact opposite of Block everything. The firewall is still technically running, but every packet is allowed through. This is the safe way to open the firewall without accidentally locking yourself out.

```
#!/bin/sh
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -F
iptables -X
```

Minimal emergency firewall

I use this to shut down everything except SSH port 22. This is my panic script. If something seems suspicious then I use this script to put a machine into as safe a state as possible while still allowing remote SSH connections.

Note that a machine with these rules won't even be visible on the network. If you want to scan it with nmap you will have to use "nmap -PO" which scans without first checking with ICMP (ping).

```
#!/bin/sh
# Minimal emergency firewall (block everything except SSH).
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -F
iptables -X
iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
iptables -P INPUT DROP
iptables -A OUTPUT -p tcp -m tcp --sport 22 -j ACCEPT
iptables -P OUTPUT DROP
```

Basic firewall

This example is a little more practical. This will block bad packets; it will allow friendly ICMP (ping requests); it will allow SSH, HTTP, HTTPS; and it will allow established and related connections, so you can use applications such as outbound ftp clients.

```

#!/bin/sh

# Flush any old policies and rules.
iptables -F
iptables -X

#
# Accept some remote connections.
#
# SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# HTTP
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
# HTTPS
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
# SMTP
iptables -A INPUT -p tcp --dport 25 -j ACCEPT
# DNS
iptables -A INPUT -p udp --dport 53 -j ACCEPT
iptables -A INPUT -p tcp --dport 53 -j ACCEPT
# VMware
iptables -A INPUT -p tcp --dport 902 -j ACCEPT

#
# Accept some localhost connections.
#
# IMAP
iptables -A INPUT -p tcp -s 127.0.0.1 --dport 143 -j ACCEPT
# MySQL
iptables -A INPUT -p tcp -s 127.0.0.1 --dport 3306 -j ACCEPT
# PostgreSQL
iptables -A INPUT -p tcp -s 127.0.0.1 --dport 5432 -j ACCEPT
# BIND RND
iptables -A INPUT -p tcp -s 127.0.0.1 --dport 953 -j ACCEPT
# VNC -- You should use an SSH tunnel to expose this to remote connections.
iptables -A INPUT -p tcp -s 127.0.0.1 --dport 5900 -j ACCEPT
# Subversion svnserve (you should probably just use the svn+ssh: URL scheme)
iptables -A INPUT -p tcp -s 127.0.0.1 --dport 3690 -j ACCEPT

# Drop illegal packets
iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP # NULL packets
iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP #XMAS
iptables -A INPUT -p tcp --tcp-flags FIN,ACK FIN -j DROP # FIN packet scans
iptables -A INPUT -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP

# Allow some ICMP (ping)
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -m limit --limit 1/second -j ACCEPT
iptables -A INPUT -p icmp -j DROP

# Match related and established state connections.
# This allows client-side connections such as ftp work properly.
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# Default policies to handle everything not covered by a rule.
iptables -F
iptables -X
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

```

Load firewall on boot

When you shutdown your server all the iptables rules will be lost. You need to run a firewall script every time you boot. Some people use firestarter (<http://www.fs-security.com/>), but I prefer edit my own simple init script.

RedHat

For RedHat you need to edit:

```
/etc/sysconfig/iptables
```

Don't confuse this with `/etc/sysconfig/iptables-config`. Also note that RedHat has a tool called `system-security-level` that overwrites `/etc/sysconfig/iptables`, so if you run `system-security-level` you will lose your changes. You can edit the file manually or you can use `system-security-level`. Choose one or the other, not both.

Ubuntu/Debian

For Ubuntu/Debian you can put an init script into `/etc/init.d` then link to an 'S' file in `/etc/rc2.d`

```
-----  
cp firewall /etc/init.d/firewall  
chmod 755 /etc/init.d/firewall  
cd /etc/rc2.d/  
ln -s ../init.d/firewall S99firewall  
-----
```

init.d script

The following init script is based on the scripts given previously. Save this in `/etc/init.d/firewall`. This includes options to start, stop, or show the status of iptables. This doesn't really stop iptables. It just deletes all firewall rules.

```

#!/bin/sh

N=/etc/init.d/firewall

set -e

case "$1" in
start)
    # Flush any old policies and rules.
    iptables -F INPUT ACCEPT
    iptables -F OUTPUT ACCEPT
    iptables -F FORWARD ACCEPT
    iptables -F
    iptables -X

    #
    # Accept some remote connections.
    #
    # SSH
    iptables -A INPUT -p tcp --dport 22 -j ACCEPT
    # HTTP
    iptables -A INPUT -p tcp --dport 80 -j ACCEPT
    # HTTPS
    iptables -A INPUT -p tcp --dport 443 -j ACCEPT
    # SMTP
    iptables -A INPUT -p tcp --dport 25 -j ACCEPT
    # DNS
    iptables -A INPUT -p udp --dport 53 -j ACCEPT
    iptables -A INPUT -p tcp --dport 53 -j ACCEPT
    # VMware
    iptables -A INPUT -p tcp --dport 902 -j ACCEPT

    #
    # Accept some localhost connections.
    #
    # IMAP
    iptables -A INPUT -p tcp -s 127.0.0.1 --dport 143 -j ACCEPT
    # MySQL
    iptables -A INPUT -p tcp -s 127.0.0.1 --dport 3306 -j ACCEPT
    # PostgreSQL
    iptables -A INPUT -p tcp -s 127.0.0.1 --dport 5432 -j ACCEPT
    # BIND RND
    iptables -A INPUT -p tcp -s 127.0.0.1 --dport 953 -j ACCEPT
    # VNC -- You should use an SSH tunnel to expose this to remote connections.
    iptables -A INPUT -p tcp -s 127.0.0.1 --dport 5900 -j ACCEPT
    # Subversion svnserve (you should probably just use the svn+ssh: URL scheme)
    iptables -A INPUT -p tcp -s 127.0.0.1 --dport 3690 -j ACCEPT

    # Drop illegal packets
    iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
    iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
    iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP # NULL packets
    iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
    iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP #XMAS
    iptables -A INPUT -p tcp --tcp-flags FIN,ACK FIN -j DROP # FIN packet scans
    iptables -A INPUT -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP

    # Allow some ICMP (ping)
    iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
    iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
    iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
    iptables -A INPUT -p icmp --icmp-type 8 -m limit --limit 1/second -j ACCEPT
    iptables -A INPUT -p icmp -j DROP

    # Match related and established state connections.
    # This allows client-side connections such as ftp work properly.
    iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

    # Default policies to handle everything not covered by a rule.
    iptables -F INPUT DROP
    iptables -F OUTPUT ACCEPT
    iptables -F FORWARD ACCEPT
    ;;
stop)
    iptables -F INPUT ACCEPT
    iptables -F OUTPUT ACCEPT
    iptables -F FORWARD ACCEPT
    iptables -F
    iptables -X
    ;;
status)
    iptables -L -v
    ;;
*)
    echo "Usage: $N {start|stop|status}" >&2
    exit 1
    ;;
esac

exit 0

```

Traffic shaping

Most iptables installs come with the "TOS" module (Type Of Service):

```
iptables -m tos -h
```

This lets you set priority options for packets.

This is a complex topic. I need to expand this with a simple setup that shows how to boost priority of interactive applications like SSH and possibly HTTP, while lowering priority for everything else. | Gentoo Wiki (http://gentoo-wiki.com/HOWTO_Packet_Shaping) is one of the better documentation sources I have found.

TCNG seems interesting. I have yet to try it: <http://tcng.sourceforge.net/>

I tried this without much luck: <http://lartc.org/howto/lartc.cookbook.ultimate-tc.html#AEN2210>

This page also has some notes: <http://www.void.gr/kargig/blog/2005/07/27/traffic-shaping-a-dsl-line-with-linux/>

Handy commands

Block an annoying machine from reaching your server

This blocks a specific IP address from your server. This is useful if you are getting annoying traffic from another machine and you want to get rid of them. Replace 255.255.255.255 with the IP address you want to drop.

```
iptables -I INPUT -s 255.255.255.255 -j DROP
```

Show packet and byte counts

This shows the counters for each rule in a chain. This shows the number of packets and bytes that have gone through a specific chain. You can use this to measure traffic.

```
iptables -L INPUT -v
```

Retrieved from "<https://www.noah.org/wiki/index.php/Iptables>"

Category: Engineering

- Disclaimers