Iptables is tool used to configure firewall rules. There are various firewall programs available in most Linux OSs which sit on top of and use iptable commands. However I prefer to use a simple bash .sh script to set my custom firewall rules.

Contents

Iptables Setup on Debian Squeeze

```
apt-get install iptables
```

Iptable command line options/switches:

```
-A => Append this rule to the WhatEver Chain
-s => Source Address
-d => Destination Address
-p => Protocol
--dport => Destination Port
-j => Jump. If everything in this rule matches then 'jump' to ACCEPT
```

```
-I => Insert at position 1 of the ACCEPT Chain
-P => Set Policy e.g. iptables -P INPUT DROP
```

Rules of Iptables:

As it is a table of rules, the first rule has precedence. If the first rule dis-allows everything then nothing else afterwards will matter.

- INIVIDUAL REJECTS FIRST
- THEN OPEN IT UP
- BLOCK ALL

```
List iptable rules:
iptables -n -L (-n prevents slow reverse DNS lookup)

Reject all from an IP Address:
iptables -A INPUT -s 136.xxx.xxx.xxx -d 136.xxx.xxx.xxx -j REJECT

Allow in SSH:
iptables -A INPUT -d 136.xxx.xxx.xxx -p tcp --dport 22 -j ACCEPT

If Logging - Insert Seperate Line *BEFORE* the ACCEPT / REJECT / DROP
iptables -A INPUT -d 136.xxx.xxx.xxx -p tcp --dport 3306 -j LOG
iptables -A INPUT -d 136.xxx.xxx.xxx -p tcp --dport 3306 -j ACCEPT
Block All:
iptables -A INPUT -j REJECT
```

My Firewall Config Script

```
vi /root/firewall.sh
#!/bin/sh
### Flush any Existing iptable Rules and start afresh ###
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -F POSTROUTING -t nat
iptables -F PREROUTING -t nat
### Set our own simple iptable rules ###
iptables -A INPUT -p tcp --dport 80 -j ACCEPT    //apache
iptables -A INPUT -p tcp --dport 443 -j ACCEPT    //apache ssl
iptables -A INPUT -p tcp --dport 53 -j ACCEPT    //dns - udp for large queries
iptables -A INPUT -p udp --dport 53 -j ACCEPT    //dns - udp for small queries
iptables -A INPUT -p tcp --dport 953 -j ACCEPT    //dns internal
iptables -A INPUT -p tcp --dport 1080 -j ACCEPT    //dante socks server
iptables -A INPUT -d 136.201.1.250 -p tcp --dport 22 -j ACCEPT      //sshd
iptables -A INPUT -d 136.201.1.250 -p tcp --dport 3306 -j ACCEPT    //mysql
iptables -A INPUT -d 136.201.1.250 -p tcp --dport 8000 -j ACCEPT    //apache on phi
iptables -A INPUT -s 136.201.1.250 -p tcp --dport 8080 -j ACCEPT   //jboss for ejc
iptables -A INPUT -d 136.201.1.250 -p tcp --dport 993 -j ACCEPT    //imaps
iptables -A INPUT -s 127.0.0.1 -p tcp --dport 111 -j ACCEPT       //to speed up mail via courier. Id
iptables -A INPUT -d 136.201.1.250 -p tcp --dport 139 -j ACCEPT   //samba
iptables -A INPUT -s 127.0.0.1 -p tcp --dport 143 -j ACCEPT    //squirrelmail
iptables -A INPUT -p tcp --dport 4949 -j ACCEPT   //munin stats
iptables -A INPUT -p tcp --dport 25 -j ACCEPT    //incoming mail
iptables -A INPUT -p tcp --dport 3128 -j ACCEPT  //squid
iptables -A INPUT -p udp --dport 161 -j ACCEPT  //snmpd
```

```
iptables -A INPUT -p icmp -j ACCEPT  //Allow ICMP Ping packets.
iptables -A INPUT -p tcp -m tcp --tcp-flags ACK ACK -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -m state --state RELATED -j ACCEPT
iptables -A INPUT -j REJECT   //Close up firewall. All else blocked.
################################

##########PORT FORWARDING########
iptables -t nat -A PREROUTING -p tcp -d 136.201.1.250 --dport 8000 -j DNAT --to 136.201.146.211:80
iptables -t nat -A POSTROUTING -d 136.201.146.211 -j MASQUERADE
################################
```

Control / Use Firewall Script

As its a simple bash file, it just needs to be executed. It is OK to run this at any time. Any existing connections will not be dropped.

```
chmod 755 /root/firewall.sh
/root/firewall.sh
```

Update Firewall

It's a simple case of editing the firewall.sh script and re-running it.

```
vi /root/firewall.sh
/add or adjust as necessary
/root/firewall.sh
```

Stop Firewall

iptables does not run as a daemon, so instead of stopping it, we "flush" any iptable rules:

```
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -F POSTROUTING -t nat
iptables -F PREROUTING -t nat
```

Calling the Firewall script on boot

When the computer is rebooted, the firewall rules are flushed. As we have all the iptable rules in firewall.sh, all we need to do is to execute this bash script when the computer boots up. There are a number of different ways of doing this.

1. Simplest Method of Executing Firewall Script on boot

```
vi /etc/rc.local
#add in:
/root/firewall.sh
```

2. Init.d firewall script

I used just to place the firewall.sh script in /etc/init.d/ and then symlink it into /etc/rcX.d (or use sysv-rc-conf to set runlevels 2,3,4,5). In debian squeeze with dependancy based boot this can cause a lot of errors when installing subsequent packages (see sample below).

Errors recieved with firewall script in /etc/init.d/

```
Error:
Setting up postfix (2.7.1-1) ...
insserv: warning: script 'firewall' missing LSB tags and overrides
insserv: There is a loop between service munin-node and firewall if stopped
insserv:  loop involving service firewall at depth 2
insserv:  loop involving service munin-node at depth 1
insserv: Stopping firewall depends on munin-node and therefore on system facility `$all' which can n
insserv: exiting now without changing boot order!
update-rc.d: error: insserv rejected the script header
dpkg: error processing postfix (--configure):
 subprocess installed post-installation script returned error exit status 1
configured to not write apport reports
                                      Errors were encountered while processing:
postfix
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

Things have changed in Debian Squeeze. Upgrading from etch or lenny to squeeze could cause any firewall.sh (or other) script to cause errors. This is due to the new dependency based boot in squeeze. Instead of using update-rc.d, you should use insserv firewall.sh Also, with dependency based boot with insserv LSB headers are required in all /etc/init.d/ scripts.

LSB Headers for Firewall script

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          custom firewall
# Required-Start:    $remote_fs $syslog $network
# Required-Stop:     $remote_fs $syslog $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: firewall initscript
# Description:       Custom Firewall
### END INIT INFO
```

This was obtained from /etc/init.d/skeleton and shorewall and apf-firewall deb files.

Init.d Firewall script complete

```
mv /root/firewall.sh /etc/init.d/firewall.sh
cd /etc/init.d/
vi firewall.sh
//add in the following lines to the start:
#! /bin/sh
### BEGIN INIT INFO
# Provides:          custom firewall
# Required-Start:    $remote_fs $syslog $network
# Required-Stop:     $remote_fs $syslog $network
```

```
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: firewall initscript
# Description:       Custom Firewall
### END INIT INFO

### Flush any Existing iptable Rules and start afresh ###
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -F POSTROUTING -t nat
iptables -F PREROUTING -t nat
### Set our own simple iptable rules ###
iptables -A INPUT -p tcp --dport 80 -j ACCEPT    //apache
iptables -A INPUT -p tcp --dport 443 -j ACCEPT    //apache ssl
iptables -A INPUT -p tcp --dport 53 -j ACCEPT    //dns - udp for large queries
iptables -A INPUT -p udp --dport 53 -j ACCEPT    //dns - udp for small queries
....
```

insserv v's update-rc.d

Update-rc.d required you to choose a position between 0 and 99 and it created symlinks. insserv does the same, however it does so cleaner. It also requires the correct LSB headers such as "Required-Start" and "Required-Stop" which set when it should be run. In the above firewall.sh with LSB headers we can see that it will be started after the network and any filesystems are up. $all can also be used, where the firewall will be called when all other init.d scripts have been run. See: http://wiki.debian.org/LSBInitScripts

```
insserv -n firewall.sh  (-n = dry run)
insserv firewall.sh (do it. -v = verbose if you want to see it doing stuff)
check if you like by looking in rc3.d
```

Ref's for init.d, update-rc.d, and insserv:
http://wiki.debian.org/LSBInitScripts
http://www.debianuserforums.org/viewtopic.php?f=46&t=555
http://adminhowtos.com/index.php?topic=9.0
http://wiki.pcprobleemloos.nl/using_lxc_linux_containers_on_debian_squeeze/vm-net
http://pastebin.com/gYBHx5cP

Port Forwarding & NAT - Network Address Translation - V.Basic:

```
iptables -t nat -A PREROUTING -p tcp -d 136.201.xxx.xxx --dport 443 -j DNAT --to 136.201.xxx.xxx:22
The Above will do on its Own. The above allows someone to ssh into the box on port 443 incase port 2
**NB** Set ipForwarding in /etc/networking/options !!!!!!!!!!
If Forwarding from another Network:
iptables -A FORWARD -p tcp -d 136.201.xxx.xxx --dport 22 -j ACCEPT
Web Port Forwarding: http://www.hackorama.com/network/portfwd.shtml
```

NB: Must allow IN Traffic and Connections the server started/ initiated (http://rimuhosting.com/howto/firewall.jsp):

```
iptables -A INPUT -p tcp -m tcp --tcp-flags ACK ACK -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -m state --state RELATED -j ACCEPT
```

Iptables Forward with NAT

This is already covered on this wiki here iptables_forward

Using iptables directly

Of course iptable commands can be entered live and will take effect immediately (instead of editing the firewall.sh script and re-running it).

Remove / Delete an individual /single Iptable Rule

```
iptables -D INPUT -s 127.0.0.1 -p tcp --dport 111 -j ACCEPT
// -D = delete appropriate rule. If you dont know the exact syntax of the rule to delete do the foll
iptables -L
//count down the number of lines until you reach the rule you wish to delete
iptables -D INPUT 4
//format = iptables -D CHAIN #Rule_No
```

Other pieces of information to remember:

iptables -P INPUT DROP (Setting the Default Policy)
iptables -A INPUT * * * -j ACCEPT | REJECT (send back 'connection refused') | DROP (keep quiet)

```
iptables -A INPUT -s 5.10.83.0/24 -p tcp --dport 80 -j REJECT
# Drop all from that IP Address range. I.e. block from 5.10.83.0 to 5.10.83.255
```

fail2ban - Debian Etch/Ubuntu

Fail2ban is a simple Debian etch package which uses iptables to dynamically add rules which blocks ips after various incorrect Authentication/Password attempts. To install:

```
apt-get install fail2ban
//configuration file is in /etc/fail2ban.conf
//fail and ban logs are saved in /var/log/fail2ban.log and /var/log/faillog
```

It monitors incorrect attempts in /var/log/auth.log for ssh attempts by default. The defaults are: 5 attempts before a rule is added blocking the client ip (on port 22) for 10minutes. Its a very very very nice package -)

Problems with fail2ban and ssh attempts on ubuntu

fail2ban was only banning ssh attempts where the user was "unknown". It was not stopping brute force attempts at root for example. The failregex for the sshd.conf had to be changed.

```
vi /etc/fail2ban/filter.d/sshd.conf
#change the failregex line to:
failregex = (?:Failed password [-/\w+]+) .*(?: from|FROM) <HOST>
```

It could be done a lot better, but the above works. Also see:
http://debaday.debian.net/2007/04/
http://debaday.debian.net/2007/04/29/fail2ban-an-enemy-of-script-kiddies/

Firewall on Centos / RH

http://www.linuxtopia.org/online_books/centos_linux_guides/centos_linux_reference_guide/s1-iptables-init.html

---

Main Page Information was Got From:

http://www.howtoforge.com/node/450

Good Firewall explaination & Netstat:

http://www.aboutdebian.com/firewall.htm

http://www.linuxguruz.com/iptables/howto/iptables-HOWTO-6.html#ss6.2

Archive

~~old debian sarge The init (start/stop) script for iptables is new within sarge - using if-up and if-down. The old init script is still available to load and save iptables rules. Do the following to set-up the iptables init script (details obtained from http://www.howtoforge.com/linux_iptables_sarge):~~

```
gunzip /usr/share/doc/iptables/examples/oldinitdscript.gz -c > /etc/init.d/iptables
chmod +x /etc/init.d/iptables
mkdir /var/lib/iptables
chmod 700 /var/lib/iptables
```

Control of Iptables (inactive is a blank file with no rules):

```
/etc/init.d/iptables save active
/etc/init.d/iptables load active | inactive
```

Saving ALL IPTABLE Rules

It seems that the method for saving & loading iptable rules from /etc/init.d/iptables load|save active|inactive does not save NAT rules.

The command for saving iptable rules manually is:

```
root:~# iptables-save > rules-saved
```

There is also command called iptables-restore. It is:

```
root:~# iptables-restore rules-saved
```