IPTABLES - BLOCK BRUTEFORCE

I run linux servers, I am also a great fan of *ssh* and for business reasons very often need to offer other services publicly. Having said that, I hate people trying to bruteforce their way onto my server, it just wastes my time and theirs, so I now use *iptables* to help take care of this, take a look...

Needed
Well, a linux box of course, and an *iptables* implementation with the *recent* patch applied. Once you have that, take a look at the following rules for dealing with ssh brute force attempts..

```
1  echo "CHECK FOR SSH BRUTE FORCE ATTEMPTS"
2  $IPT -N SSH_BF
3  $IPT -A SSH_BF -s $SAFEIP -j RETURN
4  $IPT -A SSH_BF -m recent --set --name SSH --rsource
5  $IPT -A SSH_BF -m recent ! --update --seconds 60 --hitcount 10 --name SSH --rsource -j RETURN
6  $IPT -A SSH_BF -m recent --update --name SSH --rsource
7  $IPT -A SSH_BF -j LOG --log-prefix "SSH Brute Force Attempt:  "
8  $IPT -A SSH_BF -p tcp -j DROP
9  $IPT -A INPUT -i $EXT -p tcp --dport 22 -m state --state NEW -j SSH_BF
```

Now lets explain in a bit more details...

- Line 2, this creates the rule table we will be using
- Line 3, this can be used to whitelist any IP we definitely trust
- Line 4, here we record the source ip of any trying the ssh port
- Line 5, if the source has not had more then 10 attempts in 60 secs its good
- Line 6, if a person has had more then 10 attempts we log the source ip
- Line 7, keep a log of who we find trying bruteforce us
- Line 8, drop all packets from people who attempted more then 10 connections in 60 seconds
- Line 9, this is where we pass packets from the main rule chains to our custom brute force checking one

*More*Now you can tweak this is change the time allowed, the number of connections allowed, even what ports you test for brute forcing (can use this to check for pop3 attempts). You can even use the logs to twist the knife in a little more. For example you could run a script to periodically check for people trying to brute force your services and then stop all traffic from them. To start with you will need to put the following little rule snippet quite high in your firewall rules...

```
echo "-BAD PEOPLE BANNING"
$IPT -N BAD_PEOPLE
$IPT -A BAD_PEOPLE -j RETURN
$IPT -A INPUT -j BAD_PEOPLE
$IPT -A FORWARD -j BAD_PEOPLE
```

This is just to set the stage for the following script which you can *cronjob* to run as often as you want..

```
CHKSTR="SSH Brute Force Attempt"
TMPLST=/tmp/list.tmp
BADIP=/admin/fwall/badip.list
IPT=/sbin/iptables

/bin/dmesg | grep -e "$CHKSTR" | gawk '{ print $8 }' | sort | uniq | cut -f 2 -d "=" > $TMPLST
echo "Bad List = `cat $BADIP | wc -l`"
cat $TMPLST >> $BADIP
cat $BADIP | sort | uniq > $TMPLST
cat $TMPLST > $BADIP
echo "Bad List = `cat $BADIP | wc -l`"

rm -rf $TMPLST
echo "updating firewall.."
$IPT -vF BAD_PEOPLE
for x in `cat $BADIP | grep -v -e "<safe network range1>"`
 do
  $IPT -A BAD_PEOPLE -s $x -j DROP
 done
$IPT -A BAD_PEOPLE -j RETURN
$IPT -vnL BAD_PEOPLE
```

As always with my scripts, they work for me, I make no claim to them being perfect or works of art. Please use them as you see fit. I run that script every hour and just monitor who ends up on my list. So far it has worked nicely for me.

Final Words
While these options have proved useful to me, you must still keep an eye on what you are blocking. If an attacker can spoof enough IP addresses, you could end up blocking people you would rather not. But even then, with some tweaking, you could still work something out, thats the great thing with iptables. Anyway, have fun and learn.