IPTABLES - NON-BASIC SECURITY

I like iptables, I use iptables, I have written about iptables previously (see Basic Security and Stop Scanning) and I keep getting asked to put together some settings and checks that can be used in any iptables firewall setup. So I decided to give it a bash, I have broken down each section into a specific purpose which can be used by itself in any iptables firewall. I will also be going through some theory and some example firewall rules. Now I have tried to make the settings/rules as generic as possible in the order I would use them, but you should still doublecheck them as they relate to your setup. So without further ado...

Network Security Proc Settings

```
SETTING_0="/proc/sys/net/ipv4/conf/*/accept_redirects /proc/sys/net/ipv4/conf/*/accept_source_route /
proc/sys/net/ipv4/conf/*/send_redirects /proc/sys/net/ipv4/conf/*/secure_redirects /proc/sys/net/ipv4/
conf/*/proxy_arp"
SETTING_1="/proc/sys/net/ipv4/conf/*/log_martians"
for file_0 in $SETTING_0
 do
  for y in `ls $file_0`
   do
    echo 0 > $y
   done
 done
for file_1 in $SETTING_1
 do
  for i in `ls $file_1`
   do
    echo 1 > $i
   done
 done
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
echo 1 > /proc/sys/net/ipv4/ip_forward                  ##used if you are routing traffic##
echo 0 > /proc/sys/net/ipv4/ip_forward                  ##used if you are not routing traffic##
echo 0 > /proc/sys/net/ipv4/conf/*/proxy_arp            ##used if you are not bridging traffic##
echo 1 > /proc/sys/net/ipv4/conf/*/proxy_arp            ##used if you are bridging traffic##
```

Performance Related Proc Settings

```
echo 45 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 60 > /proc/sys/net/ipv4/netfilter/ip_conntrack_udp_timeout
echo 1 > /proc/sys/net/ipv4/tcp_rfc1337
echo 1800 > /proc/sys/net/ipv4/tcp_keepalive_intvl
echo 1280 > /proc/sys/net/ipv4/tcp_max_syn_backlog
echo 1 > /proc/sys/net/ipv4/tcp_window_scaling
echo 0 > /proc/sys/net/ipv4/tcp_sack
echo 1 > /proc/sys/net/ipv4/tcp_fack
echo 0 > /proc/sys/net/ipv4/tcp_timestamps
```

Flushing Old Rules

```
iptables -F
iptables -t nat -F
iptables -t mangle -F
iptables -X
```

```
iptables -t nat -X
iptables -t mangle -X
```

Restrictive Default Settings

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING DROP
iptables -t nat -P OUTPUT DROP
```

Ensure Loopback Functioning

```
ifconfig lo localhost up
iptables -A INPUT -i lo -p all -j ACCEPT
iptables -A OUTPUT -o lo -p all -j ACCEPT
```

Setup Stateful Packet Inspection

```
iptables -A INPUT   -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat -A PREROUTING -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat -A POSTROUTING -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Check For Port Scans

##If you use this ruleblock, you will need to use the *Adaptive Dropping* ruleblock as well. ##
##Otherwise remove the *-m recent --set* switches from this ruleblock##

```
iptables -N SCAN_CHK
iptables -A SCAN_CHK -m psd -m limit --limit 5/minute -m recent --set -j DROP
iptables -A SCAN_CHK -j RETURN
iptables -A INPUT -i <external_interface> -p tcp -j SCAN_CHK
iptables -A FORWARD -i <external_interface> -p tcp -j SCAN_CHK
```

Check For Bad Packet Flags

##If you use this ruleblock, you will need to use the *Adaptive Dropping* ruleblock as well. ##
##Otherwise remove the *-m recent --set* switches from this ruleblock##

```
iptables -N BAD_FLAGS
iptables -A BAD_FLAGS -p tcp --tcp-option 64 -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-option 128 -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ACK,FIN FIN -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ACK,PSH PSH -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ACK,URG URG -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags FIN,RST FIN,RST -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags SYN,FIN SYN,FIN -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags SYN,RST SYN,RST -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ALL ALL -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ALL NONE -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ALL FIN -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ALL FIN,PSH,URG -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ALL SYN,FIN,PSH,URG -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -m recent --set -j DROP
iptables -A BAD_FLAGS -p tcp --tcp-flags ALL URG,ACK,PSH,RST,SYN,FIN -m recent --set -j DROP
iptables -A BAD_FLAGS -j RETURN
iptables -A INPUT -p tcp -j BAD_FLAGS
```

iptables -A FORWARD -p tcp -j BAD_FLAGS

Check For Bad Packet Options
##If you use this ruleblock, you will need to use the *Adaptive Dropping* ruleblock as well. ##
##Otherwise remove the *-m recent --set* switches from this ruleblock##
iptables -N BAD_OPT
iptables -A BAD_OPT -m ipv4options --ssrr -m recent --set -j DROP
iptables -A BAD_OPT -m ipv4options --lsrr -m recent --set -j DROP
iptables -A BAD_OPT -m ipv4options --rr -m recent --set -j DROP
iptables -A BAD_OPT -j RETURN
iptables -A INPUT -i <external_interface> -p tcp -j BAD_OPT
iptables -A FORWARD -i <external_interface> -p tcp -j BAD_OPT

Check For Bad Connection States
##If you use this ruleblock, you will need to use the *Adaptive Dropping* ruleblock as well. ##
##Otherwise remove the *-m recent --set* switches from this ruleblock##
iptables -N BAD_CONN
iptables -A BAD_CONN -m conntrack --ctstate INVALID -m recent --set -j DROP
iptables -A BAD_CONN -j RETURN
iptables -A INPUT -j BAD_CONN
iptables -A FORWARD -j BAD_CONN

Check For Packets of the Wrong Size
iptables -N BAD_SMALL
iptables -A BAD_SMALL -p udp -m length --length 0:27 -m recent --set -j DROP
iptables -A BAD_SMALL -p tcp -m length --length 0:39 -m recent --set -j DROP
iptables -A BAD_SMALL -p 30 -m length --length 0:31 -m recent --set -j DROP
iptables -A BAD_SMALL -p 47 -m length --length 0:39 -m recent --set -j DROP
iptables -A BAD_SMALL -p 50 -m length --length 0:49 -m recent --set -j DROP
iptables -A BAD_SMALL -p 51 -m length --length 0:35 -m recent --set -j DROP
iptables -A BAD_SMALL -m length --length 0:19 -m recent --set -j DROP
iptables -A BAD_SMALL -j RETURN
iptables -A INPUT -p tcp -j BAD_SMALL
iptables -A FORWARD -p tcp -j BAD_SMALL

Checks for Peer-2-Peer Network Traffic
iptables -N P2P_CHK
iptables -A P2P_CHK -m ipp2p --edk -j DROP
iptables -A P2P_CHK -m ipp2p --dc -j DROP
iptables -A P2P_CHK -m ipp2p --kazaa -j DROP
iptables -A P2P_CHK -m ipp2p --gnu -j DROP
iptables -A P2P_CHK -m ipp2p --bit -j DROP
iptables -A P2P_CHK -m ipp2p --apple -j DROP
iptables -A P2P_CHK -m ipp2p --winmx -j DROP
iptables -A P2P_CHK -m ipp2p --soul -j DROP
iptables -A P2P_CHK -m ipp2p --ares -j DROP
iptables -A P2P_CHK -j RETURN
iptables -A INPUT -j P2P_CHK
iptables -A FORWARD -j P2P_CHK

Sanity Checks for External Traffic
##In this ruleblock replace the *$EXT* with the name of your external interface##
##If for some reason you are using one of these network address externally, uncomment it##
iptables -N EXT_SANITY

iptables -A EXT_SANITY -s 192.168.0.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 192.168.1.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 192.168.2.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 192.168.3.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 192.168.254.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 199.196.196.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 0.0.0.0/8 -i $EXT -j DROP
iptables -A EXT_SANITY -s 255.0.0.0/8 -i $EXT -j DROP
iptables -A EXT_SANITY -s 255.255.255.255 -i $EXT -j DROP
iptables -A EXT_SANITY -s 127.0.0.0/8 -i $EXT -j DROP
iptables -A EXT_SANITY -s 224.0.0.0/4 -i $EXT -j DROP
iptables -A EXT_SANITY -s 240.0.0.0/5 -i $EXT -j DROP
iptables -A EXT_SANITY -s 169.254.0.0/16 -i $EXT -j DROP
iptables -A EXT_SANITY -s 192.0.0.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 192.0.34.0/24 -i $EXT -j DROP
iptables -A EXT_SANITY -s 10.0.0.0/8 -i $EXT -j DROP
iptables -A EXT_SANITY -s 172.16.0.0/12 -i $EXT -j DROP
iptables -A EXT_SANITY -s 192.168.0.0/16 -i $EXT -j DROP
iptables -A EXT_SANITY -s <firewalls_own_ip> -i $EXT -j DROP
iptables -A EXT_SANITY -j RETURN
iptables -A INPUT -p tcp -j EXT_SANITY
iptables -A FORWARD -p tcp -j EXT_SANITY

Sanity Checks for Internal Traffic
##In this ruleblock replace *$INT* with the name of your internal interface##
iptables -N INT_SANITY
iptables -A INT_SANITY -d 255.255.255.255 -i $INT -j DROP
iptables -A INT_SANITY -s 255.255.255.255 -i $INT -j DROP
iptables -A INT_SANITY -s ! $INTIP -i $INT -j DROP
iptables -A INT_SANITY -j RETURN
iptables -A INPUT -p tcp -j INT_SANITY
iptables -A FORWARD -p tcp -j INT_SANITY

Check for Fragmented Traffic
iptables -N FRAG_CHK
iptables -A FRAG_CHK -j DROP
iptables -A INPUT -p ip -f -j FRAG_CHK
iptables -A FORWARD -p ip -f -j FRAG_CHK

Check for Proper Connection Status and Traffic Flooding
##If you use this ruleblock, you will need to use the *Adaptive Dropping* ruleblock as well. ##
##Otherwise remove the *-m recent --set* switches from this ruleblock##
Also the values of 50 and 120 work for me, you may need to change them in your setup##
iptables -N TCP_CHK
iptables -A TCP_CHK -p tcp --syn -m conntrack --ctstate NEW -m limit --limit 50/second --limit-burst 120 -j RETURN
iptables -A TCP_CHK -m recent --set -j DROP
iptables -A INPUT -p tcp -j TCP_CHK
iptables -A FORWARD -p tcp -j TCP_CHK

Adaptive Dropping
##Only use this ruleblock if you left used the *-m recent --set* switches in other ruleblocks##
##It must also go in after the other ruleblocks using the *-m recent --set* switches##
iptables -N SOD_OFF

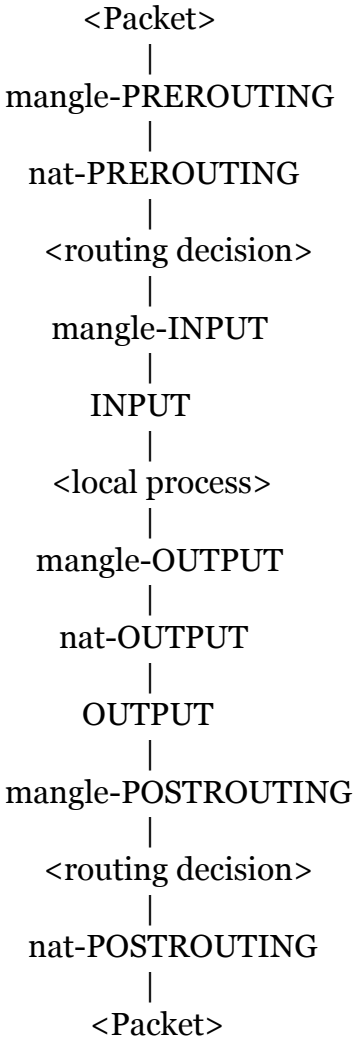
```
iptables -A SOD_OFF -p tcp -j TARPIT       ## use this only if you feel nasty
iptables -A SOD_OFF -j DROP
iptables -A INPUT -m recent --rcheck --seconds 300 -j SOD_OFF
iptables -A FORWARD -m recent --rcheck --seconds 300 -j SOD_OFF
```

Reject Identd Traffic Politely

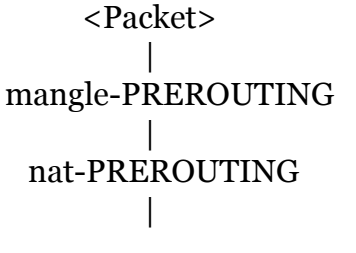
```
iptables -A INPUT -p tcp -m conntrack --ctstate NEW -m tcp --dport 113 --syn -j REJECT --reject-with tcp-reset
iptables -A FORWARD -p tcp -m conntrack --ctstate NEW -m tcp --dport 113 --syn -j REJECT --reject-with tcp-reset
```

Uhhhh... What About the "Normal Rules"?

Unfortunatly this section requires a bit more thought. While this is generally where I place the firewall rules for allowed traffic, I cannot hope to cover all the possible variations needed by everyone. So we'll try a quick crash course on rule writing. The first thing to understand is the way that a packet travels the iptables tables/chains. Lets start with a packet going to or from the firewall;



Now lets look at how a packet travelling through a firewall (meaning that the firewall forward the packet between interfaces) goes
through the iptables tables/chains;



<routing decision>
|
FORWARD
|
<routing decision>
|
nat-POSTROUTING
|
<Packet>

This is stuff you need to know so that you can construct rules for use by your firewall. But to make rules you need to know some basic (there is a lot more) syntax;

Switch	*Desciption*
-t	this specific the tablespace where the tables/chains reside which you want to work with
-A	this is used to append the rule to the targeted table/chain
-p	this is used to specify the protocol the rule must match
-i	this is used to specify the incoming network interface the rule must match
-o	this is used to specify the outgoing network interface the rule must match
-s	this is used to specify the source address the rule must match
-d	this is used to specify the destination address the rule must match
-j	this specifies the action the rule must take if the rule matches the packet

Now lets look at writing some rules (bear in mind these need the stateful packet inspection rules in place), lets start with enabling SSH (port 22) to the firewall..

iptables -t nat -A PREROUTING -p tcp -m tcp --syn --dport 22 -d <firewall_ip> -j ACCEPT
iptables -A INPUT -p tcp -m tcp --syn --dport 22 -d <firewall_ip> -j ACCEPT

Now lets enable our firewall to SSH (port 22) out from itself..

iptables -t nat -A OUTPUT -p tcp -m tcp --syn --dport 22 -s <firewall_ip> -j ACCEPT
iptables -A OUPUT -p tcp -m tcp --syn --dport 22 -s <firewall_ip> -j ACCEPT
iptables -t nat -A POSTROUTING -p tcp -m tcp --syn --dport 22 -s <firewall_ip> -j ACCEPT

Those were rules for packets going to and from the firewall, lets try one for allowing http (port 80) through the firewall..

iptables -t nat -A PREROUTING -i <incoming> -p tcp -m tcp --syn --dport 80 -d <web_server_ip> -j ACCEPT
iptables -A FORWARD -i <incoming> -o <outgoing> -p tcp -m tcp --syn --dport 80 -d <web_server_ip> -j ACCEPT
iptables -t nat -A POSTROUTING -o <outgoing> -p tcp -m tcp --syn --dport 80 -d <web_server_ip> -j ACCEPT

Lets try one for source network address translation, say for people accessing ftp (port 21)..

iptables -t nat -A PREROUTING -i <incoming> -p tcp -m tcp --syn --dport 21 -j ACCEPT
iptables -A FORWARD -i <incoming> -o <outgoing> -p tcp -m tcp --syn --dport 21 -j ACCEPT
iptables -t nat -A POSTROUTING -o <outgoing> -p tcp -m tcp --syn --dport 21 -d -j SNAT --to-source <external_ip>

Let try one for allowing smtp (port 25) into our network..

iptables -t nat -A PREROUTING -i <incoming> -p tcp -m tcp --syn --dport 25 -d <ext_smtp_ip> -j DNAT --to-dest <int_smtp_ip>

iptables -A FORWARD -i <incoming> -o <outgoing> -p tcp -m tcp --syn --dport 25 -d <int_smtp_ip> -j ACCEPT
iptables -t nat -A POSTROUTING -o <outgoing> -p tcp -m tcp --syn --dport 25 -d <int_smtp_ip> -j ACCEPT

Tarpit Them All

##Only use this ruleblock as the very last rule after all others##
##Also only use this rule if you feel like being nasty##
iptables -A INPUT -i $EXT -p tcp -j TARPIT
iptables -A FORWARD -i $EXT -p tcp -j TARPIT

Final Words

Iptables really is a good firewall to learn, stateful, extensible and free. So take the time to learn it a bit better. I hope the examples and snippets here help with that learning process. As always, have fun.