HOST FIREWALLS - LINUX MACHINES

I have, in the past, spoken about firewalls in a general sense (see here), and in that article I spoke about "personal firewalls". Firewalls which are not present on some network choke point or network border, but rather on each and every machine in your network. This is a good thing because it means that any attackers job becomes that much more difficult, as not only does he need to get past your main enterprise firewall, but also each firewall on the internal hosts. In this article I am going to go through the script I use on my linux machines to setup a firewall on each host....and yes I am using iptables.

Making Life Easier
The first part of my script is the section where I setup the variables which I am going to use throughout the rest of the script. Please replace the relevant parts with the details specific to your machine..

```
#SETUP VARIABLES
INT=<network card name>
INTIP="<internal ip address range>"
INTBC="<broadcast address for the internal range>"
HOSTIP="<the ip address of the host this script will run on>"
IPT=/sbin/iptables
MPROBE=/sbin/modprobe
```

Needful Things
Next is the section where the needed modules are loaded and any kernel switches are set..

```
#SETUP KERNEL MODULES
$MPROBE -v ip_tables
$MPROBE -v ip_conntrack
$MPROBE -v iptable_filter
$MPROBE -v iptable_mangle
$MPROBE -v iptable_nat
$MPROBE -v ipt_LOG
$MPROBE -v ipt_REJECT
$MPROBE -v ipt_limit
$MPROBE -v ipt_state
$MPROBE -v ip_conntrack_ftp

#SETUP KERNEL OPTIONS
echo 0 > /proc/sys/net/ipv4/ip_forward
```

Setting the Base
The next section is to setup the basic starting state of the host firewall. This is the getting rid of old rules, setting up the loopback address, etc ...

```
#FLUSH IPTABLES RULESET
$IPT -F
$IPT -t nat -F
$IPT -X
$IPT -t nat -X

#DEFAULT POLICIES
$IPT -P INPUT DROP
$IPT -P FORWARD DROP
$IPT -P OUTPUT DROP
$IPT -t nat -P PREROUTING DROP
$IPT -t nat -P POSTROUTING DROP
$IPT -t nat -P OUTPUT DROP

#SETUP LOCAL INTERFACE
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT

#SETUP FIREWALL RULES
$IPT -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t nat -A PREROUTING -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t nat -A POSTROUTING -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
$IPT -t nat -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Security Checks

I then add in some security checks, if you want you can use all the ones mentioned here, but the the following section is the least you should do..

```
#INTERNAL INTERFACE SANITY CHECKS
$IPT -A INPUT -s ! $INTIP -i $INT -j DROP

#BAD PACKET FLAGS
$IPT -N BAD_FLAGS
#$IPT -A BAD_FLAGS -p tcp --tcp-option 64 -j DROP
#$IPT -A BAD_FLAGS -p tcp --tcp-option 128 -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ACK,FIN FIN -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ACK,PSH PSH -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ACK,URG URG -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ALL ALL -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ALL NONE -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ALL FIN -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ALL FIN,PSH,URG -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ALL SYN,FIN,PSH,URG -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP
$IPT -A BAD_FLAGS -p tcp --tcp-flags ALL URG,ACK,PSH,RST,SYN,FIN -j DROP
$IPT -A BAD_FLAGS -j RETURN
$IPT -A INPUT -p tcp -j BAD_FLAGS

#FRAGMENT PACKET CHECK
$IPT -N FRAG_CHK
$IPT -A FRAG_CHK -j DROP
$IPT -A INPUT -p ip -f -j FRAG_CHK
```

Are You There?

The following section is used to allow your machine to ping others, and to allow others in your network to ping it. You will notice that there is a limit on the the number of incoming ping packets to will service, this is just a good idea to stop any intentional or unintentional DOS attacks. If you do not want people to ping your machine, or do not want to ping other machines just remove the relevant sections..

```
#ALLOW INTERNAL RESPONSIBLE PINGING OF HOST FROM INTERNAL NETWORK
ICMPOPT="-m limit --limit 1/second --limit-burst 10"
$IPT -t mangle -A PREROUTING -i $INT -s $INTIP -d $HOSTIP -p icmp $ICMPOPT -j ACCEPT
$IPT -t nat -A PREROUTING -i $INT -s $INTIP -d $HOSTIP -p icmp $ICMPOPT -j ACCEPT
$IPT -A INPUT -i $INT -s $INTIP -d $HOSTIP -p icmp $ICMPOPT -j ACCEPT

#ALLOW HOST TO PING OTHER HOSTS
$IPT -t mangle -A OUTPUT -o $INT -s $HOSTIP -p icmp -j ACCEPT
$IPT -t nat -A OUTPUT -o $INT -s $HOSTIP -p icmp -j ACCEPT
$IPT -A OUTPUT -o $INT -s $HOSTIP -p icmp -j ACCEPT
$IPT -t nat -A POSTROUTING -o $INT -s $HOSTIP -p icmp -j ACCEPT
```

Incoming

Here is the ruleblock used for setting the ports allowed to access the host. You will notice that there is both a TCP and a UDP section, if you want to add or remove a port which your host will work with, then just add or remove the port number from the list specified in the for loops. You will also see that each section has a variable where you can place options which will be put into your rules..

```
#TCP PORTS TO HOST
INOPT="-m state --state NEW"
for TCPIN in 22 80 21 25 139
 do
  $IPT -t mangle -A PREROUTING -i $INT -s $INTIP -d $HOSTIP $INOPT -p tcp --syn --dport $TCPIN -j
ACCEPT
  $IPT -t nat -A PREROUTING -i $INT -s $INTIP -d $HOSTIP $INOPT -p tcp --syn --dport $TCPIN -j
ACCEPT
  $IPT -A INPUT -i $INT -s $INTIP -d $HOSTIP $INOPT -p tcp --syn --dport $TCPIN -j ACCEPT
```

```
 done

#UDP PORTS TO HOSTS
UDPOPT="-m state --state NEW"
for UDPIN in 67 137 138
 do
  $IPT -t mangle -A PREROUTING -i $INT -p udp --dport $UDPIN $UDPOPT -j ACCEPT
  $IPT -t nat -A PREROUTING -i $INT -p udp --dport $UDPIN $UDPOPT -j ACCEPT
  $IPT -A INPUT -i $INT -p udp --dport $UDPIN $UDPOPT -j ACCEPT
 done
```

Outgoing

Much as the above ruleblock is used for incoming protocols, the following (also structured the same way) is used for outgoing protocols..

```
#TCP PORTS FROM HOST
OUTOPT="-m state --state NEW"
for TCPOUT in 21 22 80
 do
  $IPT -t mangle -A OUTPUT -o $INT -s $HOSTIP $OUTOPT -p tcp --syn --dport $TCPOUT -j ACCEPT
  $IPT -t nat -A OUTPUT -o $INT -s $HOSTIP $OUTOPT -p tcp --syn --dport $TCPOUT -j ACCEPT
  $IPT -A OUTPUT -o $INT -s $HOSTIP $OUTOPT -p tcp --syn --dport $TCPOUT -j ACCEPT
  $IPT -t nat -A POSTROUTING -o $INT -s $HOSTIP $OUTOPT -p tcp --syn --dport $TCPOUT -j ACCEPT
 done

#UDP PORTS FROM HOSTS
UDPIPT="-m state --state NEW"
for UDPIN in 161
 do
  $IPT -t mangle -A OUTPUT -o $INT -p udp --dport $UDPIN $UDPIPT -j ACCEPT
  $IPT -t nat -A OUTPUT -o $INT -p udp --dport $UDPIN $UDPIPT -j ACCEPT
  $IPT -OUTPUT -o $INT -p udp --dport $UDPIN $UDPIPT -j ACCEPT
  $IPT -t nat -A PREROUTING -o $INT -p udp --dport $UDPIN $UDPIPT -j ACCEPT
 done
```

Final Thoughts

The above script is what I use for my host firewalls because it is very easy to adapt to each server's needs. Just add or remove the ports you want coming in or going out for the relevant protocols. As with all my scripts, please feel totally free to improve upon it in any way you want. Have fun and learn.