IPTables

Further description of topics covered

Type of Systems: Linux, CentOS
Necessary Skills: Linux admin
Complete Goal: Enable a host based firewall to further secure your server to only necessary connections.
Relevant Info: If unfamiliar with firewalls and IPTables, refer to the section after steps, titled “Notes”
Important: This guide is designed to give you a better understanding of how to create rules. You should decide on your default action for traffic and design the firewall rules accordingly. Most of these rules are designed for accepting all outgoing traffic while blocking all incoming.
READ: Use of a firewall in the NECCDC should be done delicately. Attempting to block a certain IP or range of IP’s will do nothing but hurt the team as the ability for the red team to switch IP’s is very easy. This guide should only be done if you feel comfortable locking down your system to only certain services, as it is quite easy to block traffic necessary for your actual needs and wants.
Steps:

Initialization

1. iptables almost always comes pre-installed on any Linux distribution. To update/install it on Linux, just retrieve the iptables package with:
    a. `sudo apt-get install iptables`
    b. `sudo service iptables start`
2. If you are on CentOS, you may have to disable firewalld and install and enable iptables with the following:
    a. `systemctl disable firewalld`
    b. `systemctl stop firewalld`
    c. `yum install iptables-services -y`
    d. `systemctl enable iptables`
    e. `systemctl start iptables`
3. Before going in and configuring specific rules, you’ll want to decide what you want the default behavior of the three chains to be. To see what your policy chains are currently configured to do with unmatched traffic, run the following command:
    a. `sudo iptables -L | grep policy`
    b. If you are looking to have your default policy for incoming be DROP and default policy for outgoing be ACCEPT do the following:(NOTE: if you are looking to have a default action of drop for a certain policy, you may want to wait to apply

this until after your firewall configuration as you may kick yourself out of your system)

c. `iptables --policy INPUT DROP`
d. `iptables --policy OUTPUT ACCEPT`
e. `iptables --policy FORWARD ACCEPT`

4. If there are already some rules that you don't want to be there you can flush the rules by using the following command:
   a. `sudo iptables -F`
   b. Be careful with this command, especially if the default policy on your INPUT and OUTPUT chains is set to something other than ACCEPT because that could lock you out of your server.

Adding rules

5. Now to adding rules. The first firewall rule you need to add is the following one:
   a. `sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT`
   b. This basically tells your firewall to accept your current SSH connection if that is how you are connected to your server.
6. You may want to allow outgoing traffic of all established connections, which are typically the response to legitimate incoming connections. This command will allow that:
   a. `sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT`
7. The next step is to allow traffic on your loopback interface.
   a. `sudo iptables -A INPUT -i lo -j ACCEPT`
      `sudo iptables -A OUTPUT -o lo -j ACCEPT`
8. Now to add any other rules you would like to have. If you were looking to add a rule to allow traffic over tcp port 22 you would do the following:
   a. `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT`
   b. Repeat this step replacing 22 with any port you would like.
9. To make a rule that pertains to a specific IP address use the following -s command where x.x.x.x is the desired IP:
   a. `sudo iptables -A INPUT -p tcp --dport 22 -s x.x.x.x -j ACCEPT`
10. Allow all incoming HTTP and HTTPS traffic:
    a. `sudo iptables -A INPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`

b. `sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT`
c. The second command, which allows the outgoing traffic of established HTTP and HTTPS connections, is only necessary if the OUTPUT policy is not set to ACCEPT.

Deleting Rules

11. If you would like to delete a rule you can do the following command:
    a. `sudo iptables -L INPUT --line-numbers`
    b. `sudo iptables -D input x`
    c. Where X is the number of the line in which you want to delete.
12. If you would like to flush all rules within IPTables then do the following:
    a. `sudo iptables -F`

Logging

13. It may be useful to log dropped connections, especially when troubleshooting a problem. This can be done using a LOG statement immediately before the default DROP statement.
    a. `sudo iptables -D INPUT -j DROP`
    b. `sudo iptables -A INPUT -j LOG --log-prefix "FW-INPUT" --log-level 6`
    c. `sudo iptables -A INPUT -j DROP`

Saving IPTables Rules

14. IPTables by default does not save the firewall over persistence so therefore we must configure it this way. For **Ubuntu** do the following.
    a. `sudo apt-get install iptables-persistent`
    b. During the installation, you will asked if you want to save your current firewall rules.
    c. If you update your firewall rules and want to save the changes, run this command:
    d. `sudo invoke-rc.d netfilter-persistent save`

15. IPTables by default does not save the firewall over persistence so therefore we must configure it this way. For **CentOS** do the following.
    a. `sudo service iptables save`

Useful Commands:

`sudo iptables -L`
-this will list all rules
`sudo iptables -F`
-this will flush/remove all rules
`sudo invoke-rc.d iptables-persistent save`
-this will save current firewall rules

Notes:
Firewalls can be quite complex depending on how you make them. You may want to have a default policy of accepting certain directions of traffic or you may want to have a higher policy where you reject on incoming. If you are running an accept all incoming firewall then those rules in which you are looking to lock down certain people should be followed with a -j reject. If you are running a reject all incoming firewall and need to open up holes for certain people or services then the rules should be followed with a -j accept.

It is also a good thing to know that firewall rules typically operate in a top down approach meaning that when IPTables is checking traffic against your list of rules, it will work from the top down so if you want to allow traffic from a certain range to a given port then that should come first where the overall block will follow.

Appendix:
Below are examples of iptables which were found through the University of Maine's IT confluence page.

Minimum Policy - IPv4
The following policy is the recommended minimum stateful policy for a Linux system. Note that SSH is limited to 3 connections per minute per IP address to mitigate brute force attempts, and HTTP and HTTPS are permitted as example exceptions.

```
iptables -P INPUT ACCEPT
iptables -F INPUT
iptables -A INPUT -i lo -j ACCEPT
```

```
iptables -A INPUT -p icmp -m icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW -m recent
--set --name DEFAULT --rsource
iptables -A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW -m recent
--update --seconds 60 --hitcount 4 --name DEFAULT --rsource -j DROP
iptables -A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 443 -m conntrack --ctstate NEW -j
ACCEPT
iptables -A INPUT -j DROP
```

Outgoing Policy

Creating a policy to filter outbound traffic can be very effective at mitigating the chance of success for malicious command and control, particularly for web application servers.

The following outbound policy limits outbound traffic to processes run as root and DNS traffic. Note that if you require external connections to resources such as LDAP or database servers, you would need to create additional exceptions.

```
iptables -P OUTPUT ACCEPT
iptables -F OUTPUT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate INVALID -j DROP
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate NEW -m owner --uid-owner root -j
ACCEPT
iptables -A OUTPUT -m conntrack --ctstate NEW -m owner --gid-owner root -j
ACCEPT
iptables -A OUTPUT -m conntrack --ctstate NEW -p udp -m udp --dport 53 -j
ACCEPT
iptables -A OUTPUT -m conntrack --ctstate NEW -p tcp -m tcp --dport 53 -j
ACCEPT
iptables -A OUTPUT -j DROP
```

Malformed TCP

Many system scans make use of malformed TCP packets (combinations of TCP flags that should never occur during normal operation). While not specifically recommended, the following rules can be used to drop such packets. Ideally placed before the connection state check for INVALID.

```
iptables -A INPUT -p tcp -m tcp --tcp-flags ALL ALL -j DROP
iptables -A INPUT -p tcp -m tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
```

```
iptables -A INPUT -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A INPUT -p tcp -m tcp --tcp-flags ALL FIN,URG,PSH -j DROP
iptables -A INPUT -p tcp -m tcp --tcp-flags ALL NONE -j DROP
```

Minimum Policy - IPv6

Even if you are not using IPv6 on a server, you should create an IPv6 firewall policy to protect against link-local access to the system. It is not recommended to disable IPv6 completely.

```
ip6tables -P INPUT ACCEPT
ip6tables -F INPUT
ip6tables -A INPUT -i lo -j ACCEPT
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 1 -j ACCEPT
# Destination Unreachable [RFC4443]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 2 -j ACCEPT
# Packet Too Big [RFC4443]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 3 -j ACCEPT
# Time Exceeded [RFC4443]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 4 -j ACCEPT
# Parameter Problem [RFC4443]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 128 -j ACCEPT
# Echo Request [RFC4443]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 130 -j ACCEPT
# Multicast Listener Query [RFC2710]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 131 -j ACCEPT
# Multicast Listener Report [RFC2710]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 132 -j ACCEPT
# Multicast Listener Done [RFC2710]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 134 -m hl --hl-eq 255 -j
ACCEPT  # Router Advertisement [RFC4861]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 135 -m hl --hl-eq 255 -j
ACCEPT  # Neighbor Solicitation [RFC4861]
ip6tables -A INPUT -p ipv6-icmp -m icmp6 --icmpv6-type 136 -m hl --hl-eq 255 -j
ACCEPT  # Neighbor Advertisement [RFC4861]
ip6tables -A INPUT -p udp -m udp --sport 547 --dport 546 -m hl --hl-eq 255 -j
ACCEPT    # DHCPv6

ip6tables -A INPUT -m conntrack --ctstate INVALID -j DROP
ip6tables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
ip6tables -A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW -m
recent --set --name DEFAULT --rsource
ip6tables -A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW -m
recent --update --seconds 60 --hitcount 4 --name DEFAULT --rsource -j DROP
ip6tables -A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW -j
ACCEPT
ip6tables -A INPUT -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW -j
```

```
ACCEPT
ip6tables -A INPUT -p tcp -m tcp --dport 443 -m conntrack --ctstate NEW -j
ACCEPT
ip6tables -A INPUT -j DROP
```