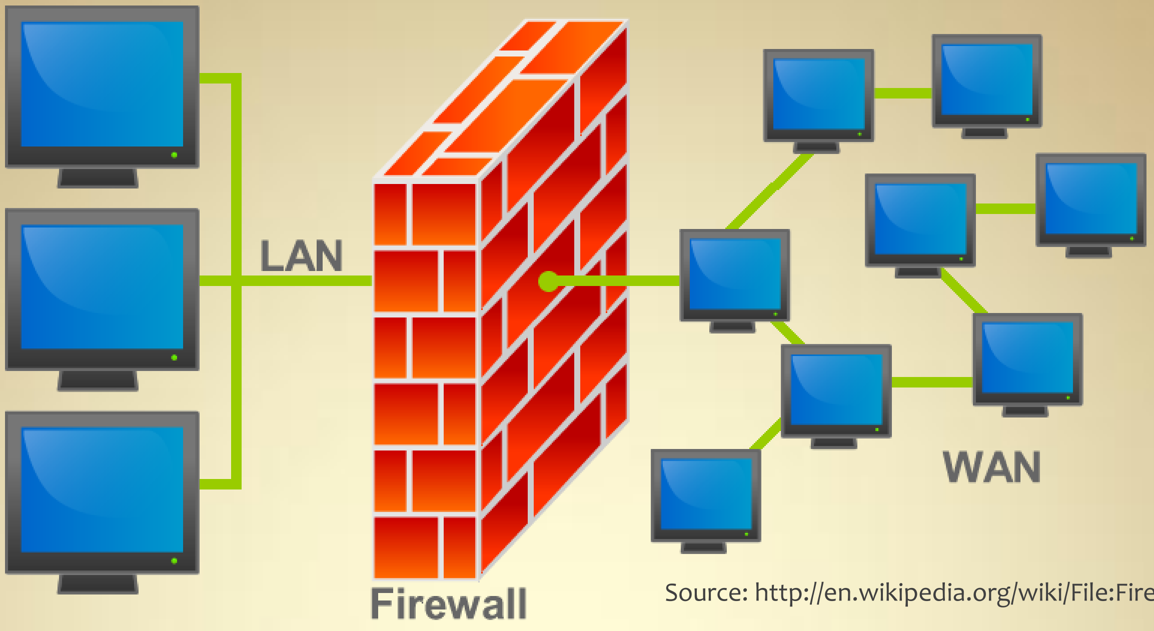


Source: http://en.wikipedia.org/wiki/File:Firewall.png

Linux Security:

Firewall

This material is based on work supported by the National Science Foundation under Grant No. 0802551

National Science Foundation
WHERE DISCOVERIES BEGIN

Lesson Overview

Networked computers are under constant threat from unauthorized users, rogue programs, and malicious scripts intent on exploiting holes in the network, protocol, or computer systems. A compromised computer can be used to create havoc on a company's network and may be used to steal valuable data. Consequently, system administrators must ensure that data requests coming to their computers are legitimate and all legitimate responses are routed to the requesting computer using the correct protocols.

One tool that system administrators use to help with network security is a firewall. A computer firewall may exist as a software-based tool, a hardware-based tool, or a combination of both. A firewall is a security tool that examines data packets as they pass through the network and routes, allows, or denies certain packets based on a set of pre-existing rules.

Understanding firewalls and the rules that allow them to be effective is a necessary skill for any aspiring administrator. A misconfigured firewall can block legitimate traffic and bring the company's network to a halt. Conversely, a well-configured firewall can minimize the risk of malicious users exploiting the network. In this lesson, you will learn to use and configure a firewall securely.

Objective

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives. Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.

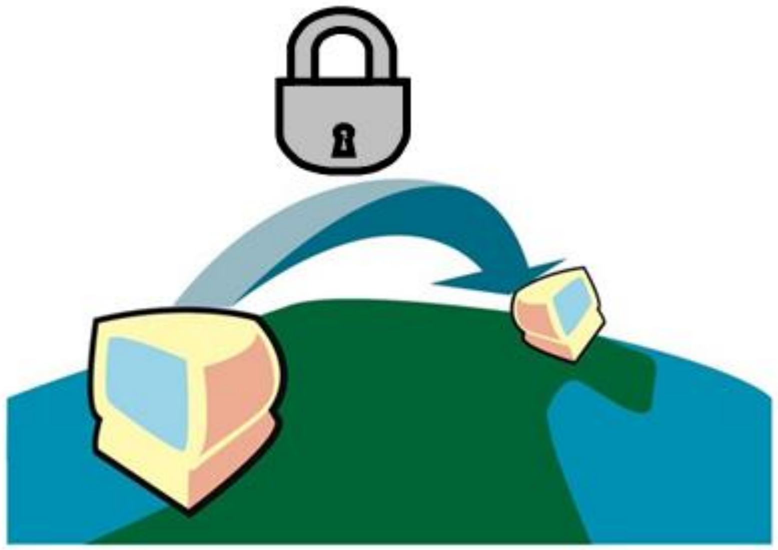
Lesson Objective:

Given the need to create a secure Linux firewall, the student will be able to compose a BASH script that successfully implements a working IPtables firewall on system boot.

Lesson Outline

In this lesson, you will explore:

- Firewalls
- Firewall principles
- IPtables
- IPtables and BASH scripting



Resources and Notes

This lesson uses Fedora Linux for demonstration. To complete this lesson successfully, you must have access to:

- ❖ Fedora Linux on bare metal or as a virtual install
- ❖ 10 Gb of hard drive space dedicated to the operating system's use
- ❖ A command shell
- ❖ Internet for research
- ❖ Word processor

Use the resources on the right to configure your system for Fedora.

Resources:
- Download Virtualbox
- Virtualbox for Linux Hosts
- Install Fedora to Virtualbox
- Virtualbox manual
- Using Virtualbox with Ubuntu (process similar to Fedora)

What is a Firewall?

The term "firewall" was originally coined as a protective wall that prevented fire from spreading from one side of a building to another. While the original definition still holds for commercial and residential buildings today, there is now another definition of firewall for the computer industry. For our purposes, a firewall is a device or application, designed to protect a network, or part of a network from tampering and information theft.

A firewall has a simple purpose—keep unauthorized users out of the network, and keep those within a network safe from outside interference or damage.

There are various types of firewalls:

- ❖ Network Layer and Packet Filters
- ❖ Application Layer
- ❖ NAT
- ❖ Proxies

Each has a different purpose, strength, and use and will be described in the next set of screens.

Recommended Reading

- Firewalls

Network Layer & Packet Filter

The network layer and packet layer firewalls are set at the Kernel level or at a low level of the TCP/IP stack. The TCP/IP stack is the portion of the Linux kernel that handles all TCP/IP communication. The TCP/IP stack is part of the network stack in the kernel.

The firewall works as the data traffic passes through the network card and into the kernel's network modules. From there, the data is either stopped or allowed to pass. The network layer and packet firewalls are the strongest and most efficient firewalls because they monitor traffic as it is coming into the network.

In the network layer firewall, traffic is not allowed to pass unless it matches a set of pre-established rules. The firewall can filter and pass traffic based on port, destination address, origin address, and protocol.

Recommended Reading
- Understanding Firewalls
- Firewall Basics

Application Layer Firewall

The application layer firewall works on specific applications. For example, if an application layer firewall was setup for "www" (web traffic) it would be up to Apache—the web server—to monitor that traffic and drop or allow it to pass based on pre-established rules.

It is rare to find an application firewall without an associated packet filtering firewall.

Application layer firewalls provide filtering (that allows or blocks data) based on application rather than on the port of the packet or network layer firewall.

Recommended Reading

- Application Layer Firewall

Proxy

A proxy is a device or software application that acts as a firewall by responding to input packets from or to an application while blocking other packets. Proxies are designed to make tampering with an internal system from an external network more difficult because all traffic must first go through the proxy prior to being routed to the internal service.

The other use of proxies is to provide a gateway to the external network while providing filtering and security rules. Many large companies use a proxy server to filter outbound TCP/IP traffic and only allow access to certain web sites, or more frequently, block traffic from web sites that are considered unacceptable in the work environment.

Finally, many academic institutions use proxy servers to allow off-campus access to restricted resources such as electronic libraries and journals. The student or faculty member would log into a proxy server, and from there, he or she would be allowed to link to the resources that are usually available on-campus only.

Required Reading

- Proxy Firewall

Network Address Translation

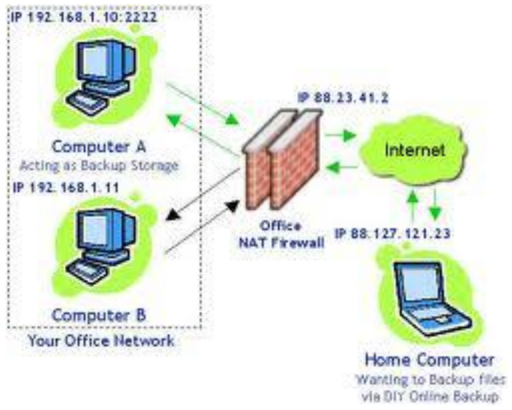
NAT (Network Address Translation) is part of most firewalls. The machines on the internal network behind the firewall have addresses in the "private address range" and these actual addresses are hidden from the external network.

IP Numbers in the private address range are only used within private networks on the internal side of a firewall. These addresses have been reserved and are duplicated in many networks. You will not find any domain names mapped to these private networks. The private address range is defined as IP addresses within the following ranges:

10.0.0.0 - 10.255.255.255 (10/8 prefix)
172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
192.168.0.0. - 192.168.255.255 (192.168/16 prefix)

The NAT functionality of the firewalls allow the machines on the internal network to access the external network. Additionally, the NAT function identifies the internal machine to the external network by substituting the actual originating IP address with the public IP address of the firewall. The public IP address is the one assigned by the Internet Service Provider (ISP).

Recommended Reading
- NAT



Firewalls Summary

In summary, the firewall has three uses:

- ❖ Prevent unauthorized traffic on the external network from reaching the internal network
- ❖ Prevent machines on the external network from "mapping" the true identity of the machines on the internal network
- ❖ Protect individual machines by filtering (denying) unauthorized traffic

In the remainder of this lesson, we are going to setup two types of firewalls. The first will be used to protect a single machine (for example a desktop) from external access. The second will be a network gateway server that will protect an entire network from unauthorized access via the NAT.

Recommended Reading

- Firewall Types

Configuring a Firewall

IPTables, Scripts & Firewalls

IPTables

The IPTables program comes with all Linux distributions and allows administrators to configure the operating system to allow certain applications and clients to connect through the network and stop unwanted applications and clients from accessing, communicating with, and sometimes, corrupting the operating system.

To protect a computer, Linux administrators will begin by denying or blocking all access to its ports. Then, they will open only the ports, protocols, and destination addresses that are needed for normal operation. All other packets (or traffic) are “dropped.” By using the word “dropped,” we mean that data packets are never seen again. The computer will simply not accept data traffic or acknowledge its existence.

IPTables is the utility we use to create filtering and access rules. Its configuration files is found in **/etc/sysconfig/iptables** on most distributions. Additionally, the configuration commands can be added to a script in **/etc/init.d** or in one of the init directories and will run at system boot.

We will be using the IPtables tool to configure the firewalls.

Recommended Reading
- What is IPTables?
- IPTables Man page

Firewall Configuration

You have been asked by your management team to put a firewall on all desktop computers in your organization. The firewall must block all incoming traffic but allow all outgoing traffic. You explain to your supervisor that you need SSH on all the computers to provide help-desk support and routine maintenance. Your supervisor agrees and modifies the request to block all incoming traffic except SSH and allow all outbound traffic. You know from your training that you are going to use IPtables to create this firewall.

Follow these instructions to create and test your firewall:

1. Start your virtual desktop machine. (In this example, we will be using Fedora, but IPtables work identically on any Linux distribution. Be sure to set your network adapter in VirtualBox to "Bridged" rather than “NAT” so that your virtual machine gets its own IP address rather than sharing the IP address of the host machine.
2. Open a terminal window and update the software using either the **sudo apt-get** or **yum upgrade** (as root).
3. On a second machine on the same network, make sure NMAP is installed. Use **sudo apt-get install nmap** from an Ubuntu or Debian machine, or **yum install nmap** from a RedHat or Fedora machine.

Recommended Reading
- NMAP Tutorial

Creating IPTables Script

Our next step is to create the IPtables script. While it is possible to always type these commands directly into the operating system by hand, the script is the only way to ensure these rules are applied at boot on multiple machines.

To create your script, follow these directions:

1. Open a terminal window and log in as root.
2. Type: **vi /etc/rc.firewall**
3. Add the following to your empty file:

```
#!/bin/sh
SERVER_IP=`ifconfig | grep 'inet addr:'|grep –v
   \'127.0.0.1'|cut -d: -f2|awk '{ print $1}'`
echo $SERVER_IP

IPTABLES=/sbin/iptables
MODPROBE=/sbin/modprobe

echo "[+] Flushing existing iptables rules....."
$IPTABLES -F
$IPTABLES -F -t nat
$IPTABLES -X
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

echo "[+] Install connection tracking modules ..... "
$MODPROBE iptable_nat
$MODPROBE ip_nat_ftp

# AT THIS POINT NO SEND AND RECIEVE FOR
ANYTHING!
```

4. Press ESC and then save the file and exit vi.

Select **PLAY** below for a video on flushing the firewall.

View Video
Videolesson8FirewallFlush
(C5L8S15).mp4

Explaining the Script

The script you just wrote is a Bash script. First, we find our own IP address. For client machines, this step is important because client machines often receive their addresses via DHCP, and DHCP-based IP addresses change frequently. We need an IP address to begin our security. You will notice that the GREP command at the end of the SERVER_IP= line excludes 127.0.0.1 because 127.0.0.1 is a "loopback" address that allows internal communication on the machine.

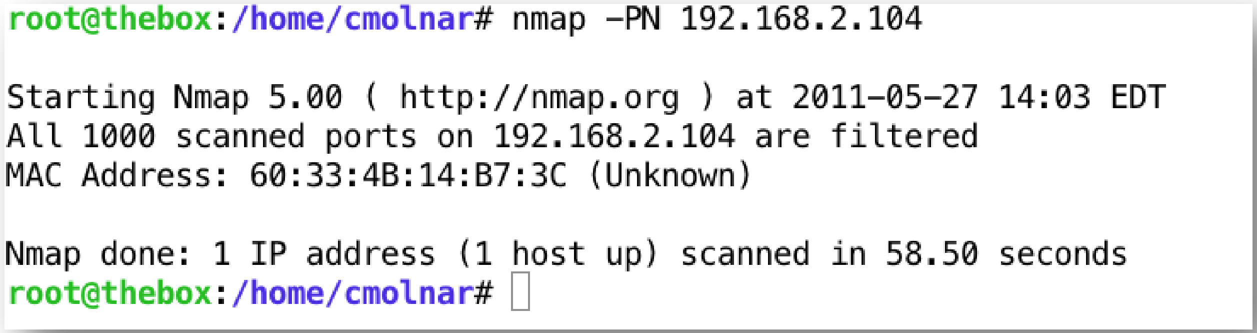
The next part of the script begins to use the IPTables command to set the basics. We always begin a new firewall script by flushing the existing configuration. In other words, we remove all existing firewall rules and start fresh. It is safer to create a new firewall rule rather than modify an existing firewall configuration.

The next section of the script sets our default policies using the iptables –P INPUT, iptables –P OUTPUT, and iptables –P FORWARD rules. We set all of these policies to "DROP" so that any traffic we do not want gets dropped (or rejected).

The IPTables command has two parts. The first is the chain (portion of the IPTables) to which the rule will be applied. The second part is the rule itself. In the above IPTables commands, the INPUT, OUTPUT, and FORWARD are the chains. If the command required it, the rules would follow.

Now, let's test what we have done so far.

Testing the Firewall



```
root@thebox:/home/cmolnar# nmap -PN 192.168.2.104

Starting Nmap 5.00 ( http://nmap.org ) at 2011-05-27 14:03 EDT
All 1000 scanned ports on 192.168.2.104 are filtered
MAC Address: 60:33:4B:14:B7:3C (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 58.50 seconds
root@thebox:/home/cmolnar# 
```

To test the firewall, follow these steps as a “root” user, or by inserting **sudo** in front of each line:

Type: **chmod u+x /etc/rc.firewall**
Type: **/etc/init.d/apache2 start**
Type: **/etc/rc.firewall**

Once the command completes:
1. Attempt to access the machine from another machine’s web browser.
2. Try to access the machine using SSH from another machine.
3. From your test machine, try to ping other machines on the network, or try to surf a web page.
4. From another machine on the same network type:
   nmap –PN [IP ADDRESS OF TEST MACHINE]

You should NOT be able to do anything on your machine; it should be acting like a black hole on the network. NMAP, in the final step, should not show any ports available on the test machine.

Testing the Firewall (Contd)

The NMAP command is a port-scanner. It will check every available port on a destination machine and tell you which ones are open and closed. For some functions of NMAP, you will be told you need to log in as root. This is because NMAP will point out vulnerabilities in network security, and in the past, it has always been assumed that root users have a higher level of responsibility and knowledge than non-root users.

This is not a useful configuration because traffic can not enter nor leave the machine. In fact, this configuration is extremely insecure because the package managers cannot update any of the packages with security updates and bug fixes.

In our next example, we are going to give your firewall some flexibility so the user and the machine are able to perform basic functions.

Required Reading

- Firewall Using IPTables

Allowing Traffic Through Firewall

In our earlier example, your manager asked you to create a firewall that would allow all outbound traffic and block all inbound traffic, except for SSH. We have now blocked all inbound and outbound traffic, so now we need to open SSH.

Follow these directions to add to your script:

1. Type: **vi rc.firewall**
2. Type **I** to access insert mode
3. Put your cursor on the last line of the file and add the following code:

```
## INPUT CHAIN ##
echo "[+] Setting up input chain...."

$IPTABLES -A INPUT -m state --state INVALID -j LOG
 \  --log-prefix "DROP INVALID" --log-ip-options
 \  --log-tcp-options
$IPTABLES -A INPUT -m state --state INVALID -j DROP
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED
  \-j ACCEPT

$IPTABLES -A INPUT -i eth0 -p tcp -s 0/0 --dport 22
  \ --syn -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A INPUT -i ! lo -j LOG --log-prefix "DROP "
  \ --log-ip-options --log-tcp-options
```

4. When complete, press ESC and then save the file by typing **:x**
5. Exit vi.
6. Type **./rc.firewall**

The new rules are in place. Let's review what these options did.

Allowing Traffic Through Firewall

Select **PLAY** below for a video on opening SSH.

View Video VideoLesson8OpeningSSH (C5L8S20)).mp4

```
## INPUT CHAIN ##
echo "[+] Setting up input chain...."

$IPTABLES -A INPUT -m state --state INVALID -j LOG
\  --log-prefix "DROP INVALID" --log-ip-options
\  --log-tcp-options
$IPTABLES -A INPUT -m state --state INVALID -j DROP
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED
 \-j ACCEPT

$IPTABLES -A INPUT -i eth0 -p tcp -s 0/0 --dport 22
  \ --syn -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A INPUT -i ! lo -j LOG --log-prefix "DROP "
  \ --log-ip-options --log-tcp-options
```

Script Description

The input rules we just added are the lines that begin with $IPTABLES. Each line has a purpose:

- Line 1: Tells the system to log any invalid packet.
- Line 2: Tells the firewall to drop any invalid packet coming into the system.
- Line 3: Tells the system to accept any packet from an established (known) connection or a related connection.
- Line 4: Tells the system to accept any connection from any source on TCP port 22. This is the port used for SSH.
- Line 5: Tells the system to accept "echo requests" or pinging.
- Line 6: Tells the system to log any other input with the exception of the traffic on the loopback (internal communication) device.

Next, let's test our new changes.

Testing Input Rules

To test the firewall, follow these steps as a "root" user, or insert **sudo** in front of each line. (See the graphic on the next page.)

1. Type: **/etc/rc.firewall**
2. Once the command completes, attempt to access the machine from another machine's web browser.
3. Try to access the machine using SSH from another machine.
4. From your test machine try to ping other machines on the network or try to surf a web page.
5. From another machine on the same network type:
   nmap –PN [IP ADDRESS OF TEST MACHINE]

You should not be able to do anything on your machine; it should be acting like a black hole on the network.

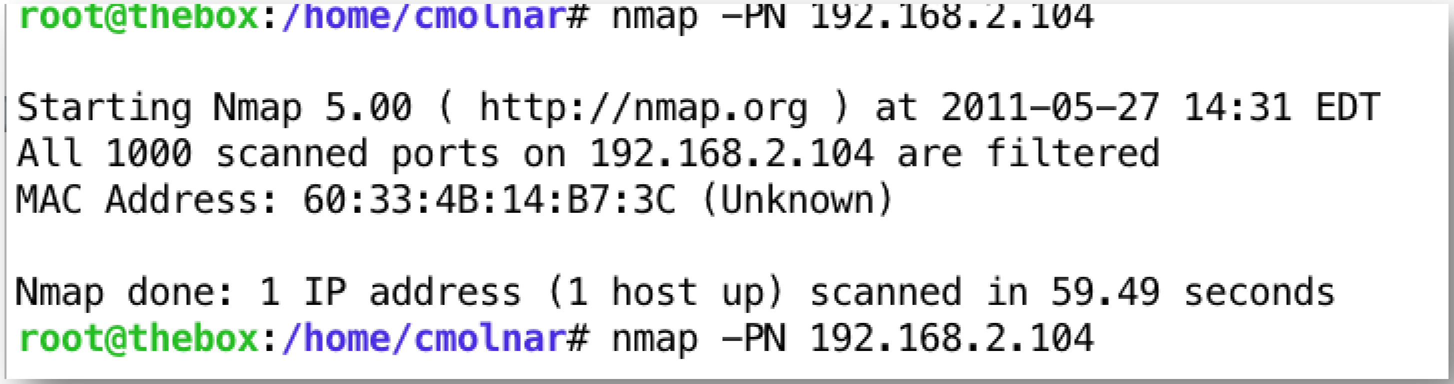
The inbound rules are in place, but the machine is still useless because most network functions require both inbound and outbound traffic.

Next, we are going to set our outbound rules in place.

Required Reading

- Linux Firewall Tutorial

Testing Input Rules (Contd)



```
root@thebox:/home/cmolnar# nmap -PN 192.168.2.104

Starting Nmap 5.00 ( http://nmap.org ) at 2011-05-27 14:31 EDT
All 1000 scanned ports on 192.168.2.104 are filtered
MAC Address: 60:33:4B:14:B7:3C (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 59.49 seconds
root@thebox:/home/cmolnar# nmap -PN 192.168.2.104
```

Screen grab showing the display during testing of the firewall Input rules

Setting Outbound Firewall Rules

While our inbound rules are in-place, we are still unable to use any network functions because most network functions are a combination of inbound and outbound rules. We now need to add the outbound rules to the script.

Follow these directions to add to your script:

1. Type: **vi rc.firewall**
2. Type **I** to access insert mode
3. Put your cursor on the last line of the file and add the code shown on the next two slides
4. When finished, press ESC, save your file, and exit vi.
5. Now, as with the INPUT rules, we need to test our firewall script, but first let's look at what this code does.

Code instructions continued on next slide . . .

Select **PLAY** below for a video on output rules.

View Video VideoLesson8OutputRules (C5L8S24).mp4

Setting Outbound Firewall Rules

```
## OUTPUT CHAIN ##
echo "[+] Setting up output chain...."

$IPTABLES -A OUTPUT -m state --state INVALID -j LOG
    \ --log-prefix "DROP INVALID " --log-ip-options
    \ --log-tcp-options
$IPTABLES -A OUTPUT -m state --state INVALID -j DROP
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED
     \ -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 21 --syn -m state
     \ --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 22 --syn -m state
     \ --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 25 --syn -m state
     \ --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 43 --syn -m state
     \ --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 80 --syn -m state
     \ --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 443 --syn -m state
     \ --state NEW -j ACCEPT
```

Code continues on next slide . . .

Setting Outbound Firewall Rules

Continued from previous slide . . .

```
$IPTABLES -A OUTPUT -p tcp --dport 4321 --syn -m state
    \ --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p udp --dport 53 -m state  --state NEW
    \ -j ACCEPT
$IPTABLES -A OUTPUT -p icmp --icmp-type echo-request
    \ -j ACCEPT
$IPTABLES -A OUTPUT -o ! lo -j LOG --log-prefix "DROP "
    \ --log-ip-options --log-tcp-options
```

Script Description

As with the input rules, each line of the outbound rules has a purpose. Let’s look at them:

- Line 1 and 2: We want to log any invalid packets and then drop them.
- Line 3: We want to allow any packets from established and related sessions to pass through.
- Line 4: We want to allow any outbound connections on port 21 (FTP).
- Line 5: We want to allow outbound connections on the SSH port.
- Line 6-10: We allow outbound connections for additional known services.
- Line 11: We must allow outbound connections on port 53 or the domain name resolver will not work.
- Line 12: We allow outbound “echo requests” or pings.
- Line 13: We log all other outbound requests unless it originates from the loopback device (lo).

Now, as with the INPUT rules we need to test our firewall script.

Test Outbound Rules

To test the firewall, follow these steps as a “root” user, or by inserting **sudo** in front of each line.

1. Type: **/etc/rc.firewall**
2. Once the command completes, attempt to access the machine from another machine’s web browser.
3. Now try to access the machine using SSH from another machine.
4. From your test machine, try to ping other machines on the network or try to surf a web page.
5. From another machine on the same network type:
   nmap –PN [IP ADDRESS OF TEST MACHINE]

At this point, you should find that you have full functionality on your test machine. Domain names should resolve, you should be able to ping, and you should be able to SSH out to other machines. The NMAP port scan should return a single port open for SSH, and all other ports should be filtered.

When you try to access SSH from other machines, you should be able to connect as well.

Congratulations, your machine and firewall script now meets the requirements of your manager!

Now, how do we change this script from a regular script to the format found in the **/etc/sysconfig/iptables** configuration file?

Export & Save IPTables Policy

At this point, the Linux administrator has two choices to make sure that the IPTables rules are set at boot time for the desktop. The first is to move the *rc.firewall* script to the */etc/init.d* directory and place a link in each of the rc#.d directories so that it executes at boot time. If you wantto use that approach, it is as simple as:

1. Type: **cp ./rc.firewall /etc/rc.firewall**
2. Type: **ln –s /etc/rc.firewall /etc/rc2.d/S99rc.firewall**
3. Type: **shutdown –r 0**
4. After a reboot, the new IPTables data should be in place.

The other, and better way to ensure the script loads at boot is to place the firewall rules in the **/etc/sysconfig/iptables** file. However, we need the script to be in a specific format. Follow these directions:

1. Make sure your firewall is working as specified by the script.
2. Type: **iptables-save > iptables.save**
3. Type: **cp ./iptables.save /etc/sysconfig/iptables**
4. Type: **/etc/init.d/iptables restart**

Your new firewall rules will be in place and should return after each system restart. The *iptables.save* file can be copied to a server or USB drive and used to configure additional machines to the same settings.

Required Reading
- IPTables Config

Configuring NAT

At this point, the computer or device is protected. Specific traffic can enter through the firewall (SSH only) and traffic on the machine can go out (for all the services defined in the firewall). If you need to allow additional services, you (as the administrator) would add them to the firewall. This policy works well for a desktop computer.

On the other hand, if the firewall settings were configured on a computer or device that served as a gateway or router between an internal network and an external network, we would need to make additional changes. While we are not going to discuss routing tables in this lesson, we will need to make some firewall changes to allow NAT (Network Address Translation) to occur.

A network router has two network ports. The first port is eth0, which in our case, is assigned to 192.168.2.104. The second port (eth1) would be tied to a second network card with an IP address of 192.168.56.10 or something similar. This second port would be used for the internal network, and all machines on the internal network would use its IP address (192.168.56.10 or similar IP) as the gateway.

Recommended Reading

- NAT with IPTables

Configuring NAT (Contd)

Network address translation says that all traffic going from the 182.168.56.* network will appear as though it is coming from 192.168.2.104. None of the original IP addresses will be visible in the packets.

To add network address translation, follow these directions:

1. Type: **vi rc.firewall**
2. Go to the end of the code and add the following:

```
iptables -A FORWARD -i ppp+ -p all -m state --state
NEW,ESTABLISHED,RELATED
          \    -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

3. Exit vi and saving your file.

As root, type **./rc.firewall** to load your new configuration.

Test this file by trying to access a site off your network from a second machine on the local network. When finished, you can export this file and load into your */etc/sysconfig/iptables* file and the next time you reboot, your new firewall with NAT will be in place.

Recommended Reading

- NAT with IPTables

Lesson Summary

In this lesson, we used the IPTables command to configure a network layer or packet firewall. We first flushed all existing firewall rules, then we configured our inbound rules, then we configured our outbound rules. This process left us with a machine that was secured by the TCP stack in the kernel which is at the lowest level of network communication.

Finally, we added network address translation (NAT) to the firewall. NAT is helpful when the machine with the firewall is used as a network router or gateway machine. NAT allows the internal network to interact with the external network without the ability of others to identify the source IP address. All the NAT-based traffic would seem to come from the gateway machine, even when each machine has its own local IP address.

Recommended Reading

- Debian Linux Firewall
- Firewall Scripts