SANS Security Essentials
GSEC Practical Assignment
Version 1.2d

4th Generation of Linux based firewalls

John Sutherland
April 11, 2001

Introduction

In consultations with different clients requiring security guidance, firewalls are often the subject of conversation. Web developers are usually under extreme pressure to put out an application quickly. When programmers are asked what they are doing to ensure a secure application the inevitable “deer in the headlights look” is what comes back. After a few moments of reflection the exclamation, “well, isn’t the firewall going to take care of that?” is usually heard. At this point our conversation usually turns to the basics of what a firewall can and can’t protect against.

Firewalls are a basic requirement for a secure system, but are no replacement for security at all layers in the OSI model. A firewall can control what traffic is allowed in and out of a network. For example, most companies want to provide at least a web server. This requires that we allow port 80, and 443 if SSL is a requirement. Other TCP/IP ports can be allowed or not depending on what services you want to allow. Here are a few popular service with their corresponding TCP port number: SMTP Port 25, Telnet Port 23, Secure Shell (SSH) Port 22, and DNS Port 53. In this paper we will explore the basics of firewalls with a focus on the new netfilter features in the new 2.4 Linux kernel.

History of Linux Firewalls

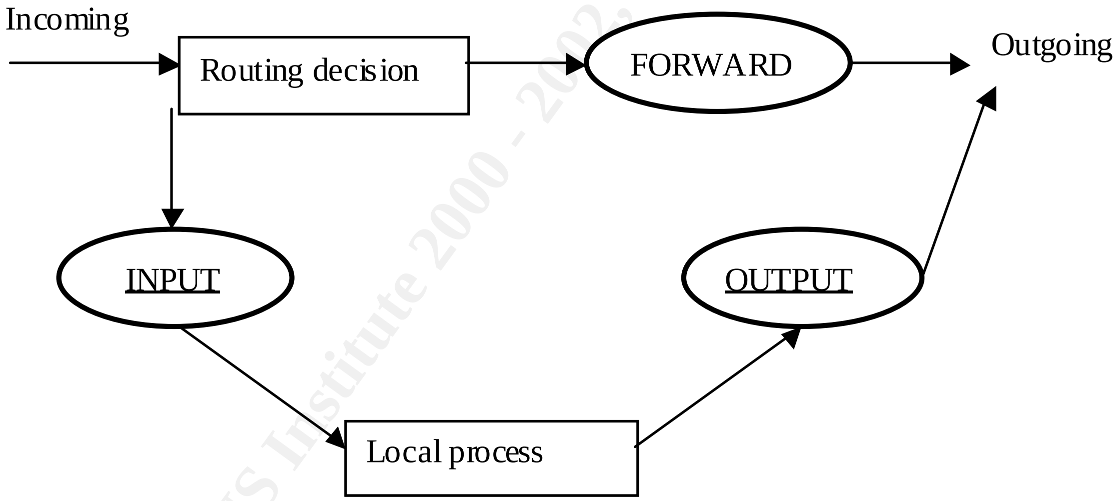
Linux has included support for implementing a firewall since the 1.1 kernel series. This first generation firewall was ported by Alan Cox from ipfw from BSD. Then in Linux 2.0 it was improved and a user tool call ipfwadm was introduced. This tool allows the user to control the kernel filtering rules. Enter Linux version 2.2, ipchains is born in 1998. Ipchains was developed by Rusty Russell and others. IPChains in kernel 2.2.x was the third generation of Linux firewalls. Now with netfilter and IPTables we have the 4th Generation of Linux based firewalls.

Netfilter is the name of the new packet handling code in Linux kernel 2.4 (actually 2.3.15 and later) and is a large redesign and improvement over previous versions of Linux. The good news for those currently using ipchains or ipfwadm is that netfilter provides backward compatible support for both. Iptables is the new command set that can take advantage of these new features.

How to implement IPTables

In order to use IPTables you need to build the kernel with netfilter support (must be kernel 2.3.15 or later) or include it as a module. Iptables is the user interface to the kernel and allows you to tell the kernel what packets to filter and what to do with them. You are actually inserting or deleting rules into the kernel's packet filtering table. Currently there are save and restore commands for IPTables as there was with ipchains, but there seems to be some problems with them, they don't work. This is a current topic on some of the IPTables related newsgroups and the issue is being worked on.

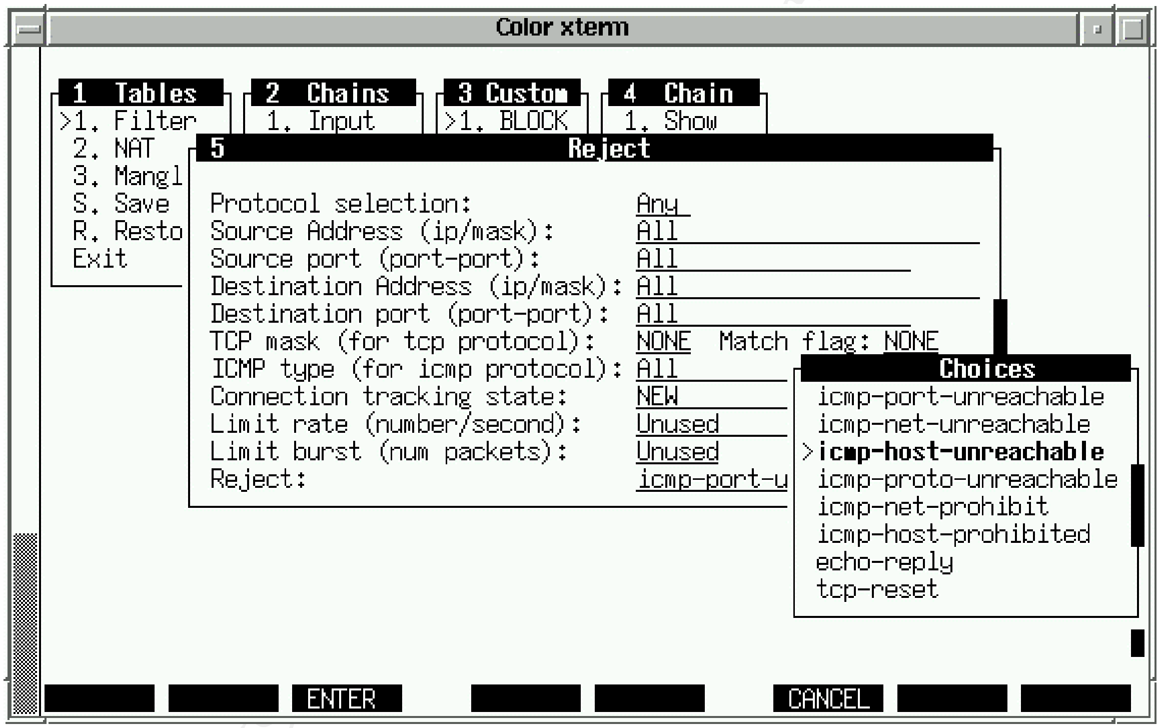
As in previous implementations of Linux firewalls, the kernel starts with three built in chains, INPUT, OUTPUT and FORWARD. A chain is a set of rules that determine what to do with a packet. When none of the rules match the packet, the kernel consults the chains default policy. If you care about security at all, the default policy should be to DROP the packet. Dropping a packet is as if you had never seen it, it just disappears into the ether. The problem with REJECTing or denying a packet is that it will still return a response to someone scanning your network informing them that you are alive, but are ignoring them. This can indicate to the person scanning you that you are running a firewall. The less information you give a potential hacker, the better off you are. Here is my version of the routing through the chains, this drawing is based on the Linux 2.4 Packet Filtering HOWTO by Rusty Russell.



There are a few key differences between IPChains and IPTables and how they handle an IP datagram. In IPChains all datagrams would be applied to the input chain, whether or not they were destined for the local host or just being routed to another host. Netfilter, on the other hand, applies the input chain against only datagrams destined for the local host and the forward chain only applies to datagrams destined for another host. IP Chains applies the output chain against all datagrams leaving the local host even if the datagram originated on the local host. Netfilter on the other hand the output chain applies only to datagrams originating from the local host and not to datagrams being routed from another host. This results in simpler firewall configurations, which are more secure since they are simpler and, therefore, easier to write, maintain and troubleshoot.

Consider the example of a configuration with a default policy of the input, output and forward chains is deny. In IP chains we need six rules. Two for each of the three chains. You need one for each path, forward and reverse. With iptables you would need two rules, one for the forward and reverse direction in the forward chain. Much easier, this is the way to configure firewalls!

Command line interface can be a real obstacle to many users. Ipchains has several GUIs available, gfcc - GTK+ Firewall Control Center, FCT - Firewall Configuration Tool, and a few others. IPTables also has a few GUI interface, IPMenu and Knetfilter, a KDE GUI interface, are two that I have seen. IPMenu is available from http://users.pandora.be/stes/ipmenu.html. This is the first GUI that I found for IPTables and looks very promising; here is a sample screen shot.



```
Protocol selection:             Any
Source Address (ip/mask):       All
Source port (port-port):        All
Destination Address (ip/mask):  All
Destination port (port-port):   All
TCP mask (for tcp protocol):    NONE  Match flag: NONE
ICMP type (for icmp protocol):  All
Connection tracking state:      NEW
Limit rate (number/second):     Unused
Limit burst (num packets):      Unused
Reject:                         icmp-port-u
```

```
 icmp-port-unreachable
 icmp-net-unreachable
>icmp-host-unreachable
 icmp-proto-unreachable
 icmp-net-prohibit
 icmp-host-prohibited
 echo-reply
 tcp-reset
```

The nice thing about IPMenu is that it does NOT require you to run X-windows on a firewall machine. This allows those of us that are command line challenged to also maintain better security by not running X-windows on a firewall machine.

PACKET vs. STATEFUL Firewalls

IPTables allows us to implement a stateful firewall, which provides greater control and greater security than the previous 2.2 and earlier versions of Linux packet filtering. So you say to yourself great, we can do stateful firewalling, so what? What's the difference? What good does it do a firewall to maintain the state? As we all know, TCP connections are made up of a series of packets. These packets carry information such as the source address, the destination address, and, if a packet gets out of order, a sequence number so the packets can be put back together. TCP connections are considered connection-oriented instead of connectionless like UDP. Hang in there with me, we are getting to the why is a stateful firewall important.

Consider that you have a stateless firewall and intend to not allow external computers to originate a connection to your internal resources. How can one determine whether a packet is part of an ongoing connection or is not? A stateless firewall can only determine a packet that is part of an existing connection versus one that is part of a new connection by looking at the SYN flag. Now let's say a bad guy is sending us packets, is it possible he might alter the SYN flag or other parts of a packet? Hmmm? This is exactly how some scanners exploit this to bypass the firewall. A stateful firewall will keep track of all its connections. When a packet shows up, claiming to be part of an existing connection, it is checked. When it is found to not be part of an existing connection, it is discarded. So, all of those nmap users that use nmap's ability to do "stealth scans" can be stopped at your new netfilter enabled firewall. We can go on with additional examples but it is not necessary, we can see that having the ability to maintain state with a firewall gives us additional control and can make our rule sets simpler and smaller.

Rate Limiting and Denial of Service Attacks

Netfilter, the new packet-filtering module in 2.4, provides several additional features over the packet filtering capabilities of previous kernels. We have seen that stateful packet filtering is effective against common attacks that exploit the SYN flag for example. An additional feature that netfilter includes is the ability to limit bandwidth through rate limiting. There are several uses for this. One possible use is to limit the amount of bandwidth that a particular server uses so it can't hog all your bandwidth. We can use this to protect us from some common network attacks.

Resource starvation is a method that Denial of Service (DoS) attacks are based on. One of the most common is the SYN flood. This is called a SYN flood, because it attempts to "flood" the target machine with connections that are left open, or unanswered. This exploits a weakness or "design feature" in the TCP/IP protocol.

As previously mentioned, the Transport Control Protocol (TCP) is a connection-oriented protocol that uses handshaking to start transmission of data. When two hosts X and Y want to make a TCP connection they use a three-way handshake. Here is an example of the beginning of a conversation between Hosts X and Y:

```
  1  X --------------SYN--------------> Y

  2  X <----------SYN/ACK-----------  Y

  3  X --------------ACK---------------> Y
```

On line 1 the X is telling Y that it wants to establish a connection. X tells Y that the sequence number field is valid, and should be checked. X sets the sequence number field in the TCP header to its ISN (Initial Sequence Number). Y, upon receiving this segment on line 2 responds with it's own ISN and an acknowledgement (ACK) of X's first segment (which is X's ISN+1). X then sends ACK Y, depicted on line 3. The handshake is now completed and the connection established and the data transfer may take place.

Essentially an attacker sends a number of TCP SYN packets. These SYN packets are used to start a TCP connection. When your computer responds to the SYN the evil computer merely refuses to answer any of your replies, or the evil computer will send SYN packets (TCP packets with the SYN flag set) with a bogus IP address. Your computer waits for the response, which never comes. Depending on the timeout settings in your system, this connection could stay open for 30 seconds, 60 seconds or longer. Your connection table has a limit. If a bad guy sends enough of these half open SYN connections he/she can keep your machine completely occupied and unable to communicate with anyone.

Here is where Netfilters rate limiting feature pays off, simply limit the number of SYN packets from a single source. Netfilter allows just such a defense because it implements basic rate limiting. Here is an example of the command:

```
iptables -A INPUT -p tcp --syn -m limit --limit 3/s --dport 21 -j ACCEPT
```

This rate-limits syns (incoming connections) to port 21 on this server to three per second. This restricts the effect of syn floods on your resources. The rate limiting can also be used to limit bandwidth usage, not only by ip but types of services.

Summary

There are many differences in Netfilter/IPTables when compared to ipchains. Illustrated above in the section on 'How to Implement IPTables', we saw how a packet is routed in kernel 2.4. This is a key difference over previous versions of the packet handling routines such as ipchains. Netfilter provides a raw framework for manipulating packets as they traverse through various parts of the kernel. There are many new features of which we have only covered a few. This new framework includes support for masquerading, standard packet filtering, and even complete network address translation. We saw just a glimpse of rate limiting which could be used to support load-balancing requests for a particular service among a group of servers behind the firewall. This difference is in the kernel design and is at the heart of the improved performance of netfilter/IPTables.

Linux has been a popular network platform and is growing in popularity every day. IPTables is an addition to the Linux toolbox that will help it become a more viable option for many individuals and companies.

References

1. Russell, Paul 'Rusty'. "Linux 2.4 Packet Filtering HOWTO"
v1.0.1 Mon May 1 18:09:31 CST 2000.
URL: http://netfilter.kernelnotes.org/unreliable-guides/packet-filtering-HOWTO/index.html
(8 April 2001)

2. Kirch, Olaf & Dawson, Terry. "Linux Network Administrators Guide", 2nd Edition June 2000.
URL: http://www.linuxdoc.org/LDP/nag2/x-087-2-firewall.howto.html (8 April 2001)

3. Andreasson, Oskar. "IPTables Tutorial"
URL: http://www.boingworld.com/workshops/linux/iptables-tutorial/ (8 April 2001)

4. Beale, Jay. "Linux Gets Stateful Firewalling Introducing Netfilter (iptables)"
URL: http://securityportal.com/cover/coverstory20010122.html (8 April 2001)

5. Weltehttp, Harald. "The netfilter framework in Linux 2.4"
URL: http://www.franken.de/kongress/00/flat4/netfilter.pdf (8 April 2001)