

Running head: Ipfilters, Ipchains and Iptables

An Analysis of Ipfilters, Ipchains and Iptables

Charles Moore

East Carolina University

Dr. Lunsford / DTEC 6823

November 22, 2005

Abstract

This paper will analyze and contrast three public domain firewalls. Those firewalls are ipfilters, ipchains and iptables. A firewall is a requirement for any business or individual that interacts with the Internet. These products represent the most wide spread applications to date and to a degree the continued maturity of the open source community. By looking at these products, one can see how the open source community evolves driven by user requirements and developers trying to secure network resources.

An Analysis of Ipfilters, Ipchains and Iptables

Introduction

It is important for all to realize that information security is more than just strong passwords, policies or technologies. Information security is about developing and implementing a strategy for information security to protect technology resources. One such strategy to protect network resources is called defense in depth. The goal of a defense in depth strategy is to incorporate a wide array of information security tools to protect your company or network assets. Today one of the key components of a defense in depth strategy is a firewall. In simplest terms, a firewall is a piece of software or hardware or both that stands in front of your network protecting it from malicious threats that are present on the Internet. It performs these tasks by examining the flow of data into and out of your network. The firewall contains rules that determine what action to take based on the flow of packets. Several years ago, a firewall was considered adequate protection for protecting your network resources. However, today a firewall is part of a set of minimum requirements for using and operating on the Internet. Firewalls come in all different flavors and operating systems. This paper will examine and contrast three common types of public accessible Internet firewalls. The firewalls examined exist in the UNIX or Linux environment. This paper will neither consider Microsoft's solution called Internet Security and Acceleration Server (ISA) nor any appliance or vendor specific firewalls, but will examine and contrast the foundation of many firewalls today. The publicly available firewall solutions offer a more flexible approach to network security. Often one can get away with implementing one of these firewalls without the acquisition of additional hardware. The three firewalls examined in this paper are ipfilters, ipchains and iptables.

There are currently two broad classes of firewalls. The first class is a packet filter firewall and would be considered a first generation firewall. The second class of firewall is called a stateful inspection firewall and is deemed a fourth generation firewall. A packet filter firewall is an early elementary firewall where decisions by the firewall are made based on the source and destination addresses and source and destination ports of the packet arriving or leaving the network. The decisions are made on a packet-by-packet check. Using this method appeared, initially, to be very secure. For example, all web traffic is destined for port 80. If a company does not have a web server at their site this port, port 80, can be restricted to allow no access. Since the default policy of a firewall is to deny all incoming traffic this allows a person or company to keep out all traffic that has been determined to have no value for them. However, it became clear that hackers or attackers could take advantage of this weakness in packet filtering and direct packets to a company using a port that was allowed into the network and then, once behind the firewall exploit vulnerabilities at will. Due to the fact that the firewall rules allowed this particular traffic into the network, there were no further checks.

A stateful inspection firewall performs all of the same functionality of a packet filter firewall, but it now includes the ability to follow the state of the packet. By tracking the state of the packets, the firewall can examine the entire session and keeps track of the entire connection of a transmission control protocol (TCP) session. The firewall tracks the status of the TCP header flags (SYN, ACK, and FIN). The firewall also tracks the actual TCP sequencing number. This enables the firewall to be able to apply the firewall rule sets more granularly. An example of this state tracking can be seen in a domain name server (DNS) query. When a client makes a request for a web page, the computer must query a DNS. This is handled by the client initiating a user datagram protocol (UDP) request on a random high port (greater than 1023) to the DNS

server, which is listening on port 53. The DNS server will then respond with the ip address of the web site. In a stateful firewall, the firewall will understand that the response from the DNS server is a legitimate response to a session that was initiated by a client behind the firewall. Further, you can tighten your rules down to only accept DNS replies from a specific DNS server and limit potential malicious traffic getting into your network

One of the biggest issues people have in adopting either of these types of firewalls is that the rules and rule sets can be quite complex. The complexity of the rules is usually driven by one's organization. However, this complexity also makes analyzing and debugging very difficult. Here is a sample iptables rule; while ipchains and ipfilters may be a little bit different on syntax, the requirements are very similar:

Iptables -A input -t DENY -p tcp --destport telnet,ftp,shell

To understand fully this rule it is necessary to analyze it. The **iptables** is telling the system to use iptables, the **-A input** tells iptables to append a new rule on the input chain. The **-t DENY** will deny the action. The **-p tcp** refers to the TCP protocol. The **--destport telnet,ftp,shell** refers to the destination ports. In simplest terms, this rule is telling iptables to amend its rule set and deny any outgoing traffic destined for telnet, ftp (file transfer protocol) and the shell command. As one can see, working with the creation of these rules can be very tedious. It has possible that NMAP, www.insecure.org and NESSUS, www.nessus.org, can be good tools to have in your arsenal to debug rules. However, both of these tools can cause havoc on your network and network resources so care should be taken in their usage.

Ipfilters

Darren Reed originally designed Ipfilters as a packet filtering firewall for UNIX. It can be modified to support FREEBSD, OPENBSD, NETBSD, SOLARIS, IRIX, and Linux among

other operating systems. In 2001, the creator of ipfilters made a move to clarify the licensing of ipfilters. The owner informed the public that ipfilters is not part of and was never part of open source software. By not allowing ipfilters to be part of the General Public License or GPL or any other similar software license, OPenBSD was forced to remove ipfilters from its distribution and replace it with something else. Apparently, the sticking point of the groups was that Darren Reed, the owner of ipfilters, wanted all changes to go through him prior to enhancements being released, and this directly contradicts the GPL (Weiss, 2001). From Mr. Reed's perspective and the user community there is a plus to being able to maintain this level of control over your product. You can take advantage of a large community of developers while insuring the functionality and code remain solid and robust. However, this was not the design of the GPL. There is apparently still a tremendous following for the product, although this decision certainly limits adoption by other vendors and independent development on the product. While ipfilters started off as a packet filter it, it currently does offer stateful technology. Ipfilters is a very robust and flexible firewall. It currently uses the last match principle, although a user can change it. This means that the last rule the packet matches is the rule that will be applied (Zwicky et al, 2000).

There are some graphical user interfaces (GUI) for Ipfilters. Those include ISBA and Firewall Builder. FreeBSD uses a replacement called IPFW. IPFW was rewritten in 2002 and called IPFW2 (Korff et al, 2005). OpenBSD has had to develop a new firewall add-in. The new add-in is called PF. It is possible to use ipfilters in a Linux environment, but it is not native. IPF 4.x can run on Linux with the 2.6.x kernel. Due to the lack of adoption by the public of the above-mentioned operating systems, this paper will focus on ipchains and iptables. The most up to date information on ipfilters can be found at <http://coombs.anu.edu.au/~avalon/>.

IPChains

On a separate path, Rusty Russell had worked on a Linux program that would provide the same functionality as ipfilters. This first product was called ipchains created in 1998 (Wikipedia Nefilter/iptables 2005). Ipchains got its name from the fact that it connects rules in the same manner as a chain (Shimonski et al. 2003). Ipchains is a packet filter firewall (Bueno, 1999), as stated above this type of firewall looks at the source and destination address or ports of the network activity. Ipchains can work as a network firewall or it can be configured as a host based firewall protecting just one machine. Ipchains has been included with the standard Linux operating system installation since Red Hat version 6 and the 2.2.x kernel (Liang, 2000). The premise behind ipchains is that there are three types of chains. They are input, forward, and output. The input chain examines incoming packets sent to the interface. The packets presented to this chain are forwarded to the next chain unless a rule match has occurred. The input chain can either drop or reject the packet. The forward chain decides what action to do with packets that are destined to another network or system. The forward chain only works if there are at least two network cards in the machine acting as a firewall. The final chain is the output chain. The output chain determines what to do with packets leaving the system. Any packet that is traversing an ipchains firewall will need to go through all of the chains. This is not very efficient.

At the time that ipchains was created there was a concern from many that the Internet would run out of IP addresses. While there are many that believe this is still any issue the original dire predictions has yet to happen. However, it is a matter of time when this will again become a problem. The Internet was created with basically three classes of addresses for public

consumption. At the time, the Internet started there was little thought to the number of addresses in the pool of available addresses. There were literally tens of millions of addresses and every one thought this would be plenty. Unfortunately, no one could see the explosive growth of the Internet and the distribution of addresses was not controlled very well. This led to a shortage of potential useable ip addresses. During this time there was a discussion concerning a new IP addressing scheme, which would allow almost an infinite amount of addresses. This new addressing scheme is called IPv6. The problem with this solution is that this new addressing scheme has to be backwards compatible and could not negatively affect the current operations of the Internet. IPv6 represented a great stride forward, but it became quite evident that this is a long-term solution and would take years to implement. A short-term solution was needed. The short-term solution had its roots in the original Internet, and it had been underutilized. This solution became known as network address translation or NAT. When the Internet was created there were three specific address ranges, 10.0.0.0, 172.16.0.0, 192.168.0.0, one in each class of numbers that were deemed private and any one could use them within their local network (Ippolito, 2004). The problem with these addresses was that the addresses were non-routable. This meant that the addresses were useable within a network, but one needs a public or routed address to go out onto the Internet. However, by using this technique a company could have thousands of machines behind a router or firewall and use only one public address. The router or firewall would keep a translation table of what internal or private address was destined to a particular Internet host. By incorporating NAT a company needs very few public addresses. Typically, a company would only need public IP addresses for their web sites, email servers, DNS servers or any other resources that some one on the Internet would need to access. This change led to a drastic reduction in the demand for public ip addresses.

Ipchains allowed the system administrator to easily incorporate network address translation. Ipchains use is limited to the following types of connections. The connections are point-to-point protocol (PPP), digital subscriber lines (DSL), cable modems, or T1 data circuits. There are a number of automated firewall scripts projects in the works. Often these scripts give the user the option of using a GUI without have to work on the command line. Some of these scripts are Firestarter, Mason, Knetfilter, Firewall Builder and Easy chains (Shimonski et al. 2003). For ipchains to work in the DNS example above it would be necessary to allow all ports above 1023 to be opened to be able to respond to the replies which is not a wise idea.

Ipchains and Ipfilters can offer the same similar functionality. The decision to use a specific program was usually based on the underlying operating system that the individual was using, as well as the users experience with a particular product. Ipchains does have some advantages over ipfilters. Ipchains network address translation is much stronger. Ipchains is part of the Linux. There are no integration issues. Ipfilters is not dynamically updateable. However, ipfilters does provide more filtering options and more flexible options and responses for TCP packets. Ipfilters uses a single configuration file. Ipfilters checks all rules in sequence and the last rule that matches determines the action to take for the packet. On the other hand ipchains applies rules in the order that the rules are read. Once a rule is match that action is taken. If no action then the next rule is read. This process goes on sequentially.

IPTables

Based on the previously discussed limitations with ipchains as well as a desire to add more functionality, Rusty Russell in 1998 began creating the next generation Linux firewall.

This product was called Netfilter/iptables. The product is licensed under GNU general public license and was merged into the Linux 2.3 core kernel in 2000. Iptables is standard in the 2.4.x and 2.6.x kernel (Bandel, 2001). Iptables is standard with Red Hat versions 7.1 to 9.0 and Fedora 1, 2, 3, 4 (Ippolito, 2004). You can find more information out on this firewall at www.netfilter.org. Iptables came out as a stateful firewall. The purpose of a stateful firewall is to monitor the state of connections and redirect, modify or stop packets based on the state of the connection not just the source or destination address, including port information (Suehring, 2005). Iptables also supports IPv6 in ip6tables. Iptables offers a tremendous amount of granularity in shaping the traffic that leaves and enters ones network.

Iptables also enhanced network address translation over ipchains. In ipchains source NAT was covered, but destination NAT was not handled natively. Another program had to be installed. Iptables provides these tasks natively. This is what allows iptables to forward address and ports to internal nonpublic addresses. This is called dynamic network address translation or DNAT. DNAT allows some load balancing to be performed.

Iptables can examine all six TCP flags (which include SYN, ACK, FIN, RST SYN, ACK), where ipchains is only aware of the SYN flag. Addresses can now be controlled based on the media access control address. Iptables also offers enhanced logging (Hoffman, 2003) and the ability to limit connection requests. This last functionality allows the end user to throttle bandwidth based on a potential denial of service or other suspicious activity. Iptables also offers type of service (TOS) prioritization. This becomes more and more important as more applications are pushed on the Internet that demand a minimum level of throughput to be effective. Examples of applications that need this type of robustness are the voice over Internet protocol (VOIP) and streaming media (H.323).

Iptables has expanded the ipchains and groups the chains into three tables. The tables are filter, nat and mangle (Bandel, 2001). Iptables also increased the number of chains from three in ipchains to five in iptables. Like ipchains, iptables has an input; output and forward chain, but prerouting, and postrouting were also added (Bandel, 2001). Each chain performs specific tasks that are particular to the table where it is located. All packets pass through the filter table. The filter table will block or allow packets through the device. All packets pass through one of the chains, input, and output or forward contained in the filter table. The nat table handles how network address translation is handled and sets up the packets for changes in the address or ports as required by the rules established. Only the first packet will pass through this table. This table uses the prerouting, postrouting and output chains. The final table is known as the mangle table and all packets will go through this table. This table is responsible for altering packet options such as quality of service. This table provides advanced options. The chains within this table are prerouting, input, forward, output, and postrouting chain.

While the input, output and forward chains perform the exact same functionality as exists in ipchains, there are several new options. It is necessary to better understand these particular new chains. The chains are prerouting and postrouting. Within the NAT table the prerouting chain analyzes incoming packets to modify the packets before going to the local routing table. This is used primarily for destination network address translation or DNAT (Napier, 2005). DNAT is a process that writes the packets destination address or port (Bandel, 2001). This can be useful when a company has an internal web server with a private address that it wants to allow the public to be able to access. The postrouting chain performs the opposite tasks and packets are presented to this chain after the routing path has been determined. This chain is used for source network address translation or SNAT. SNAT enables the firewall to rewrite the

source address or source port (Bandel 2001). The SNAT would allow users on the internal network to use private ip addresses, but access the Internet when they desire.

To show the improvement of iptables the command referenced above, **Iptables -A input -t DENY -p tcp --destport telnet,ftp,shell** would have needed to be separated for each destination port if used in ipchains (Napier, 2001). The user must use either iptables or ipchains. They will not work together.

There are several graphical user interfaces for iptables. They include fwbuilder, (<http://www.fwbuilder.org>), the turtle firewall project (<http://www.turtlefirewall.com>), IPMenu (<http://users.pandora.be/stes/ipmenu.html>), and easy firewall generator (<http://easyfwgen.morizot.net/>) (Andreasson, 2005).

Conclusion

This paper has focused on several different types of publicly available firewalls and to some degree on the evolution and the maturity of the products. From a user prospective it is possible to see the directions these products are taking. There are really two choices and each comes with pluses and minuses. The two choices are ipfilters and iptables, while outside the scope of this paper PF for FreeBSD is a possibility. Ipfilters is also a stateful inspection firewall, but has less acceptance and support. Iptables is a robust, widely accepted and supported stateful inspection firewall. It is built into almost any of Linux distributions found on the Internet today. It well documented and the user can use command line rules or a graphical user interface. Each should serve the end user well.

References

- Andreasson, Oskar, (2001-2005) *Iptables Tutorial 1.2.0*. Retrieved September 19, 2005, from <http://iptables-tutorial.frozentux.net/iptables-tutorial.html>.
- Bandel, David (2001), Taming the Wild Netfilter, *Linux Journal*, page 1
- Bueno, Peter, (1999) Building a Firewall with IPChains, *Linux Journal*, ,page 1
- Hoffmann, Daniel, & Prabhakar, Durga, 2003, IBM Centre for Advanced Studies Conference, Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative Research, page 2
- Ippolito, Greg, (2000 – 2004) *YoLinux: Using Linux iptables or ipchains to set up an internet gateway / firewall / router for home or office*. Retrieved October 7, 2005, from <http://www.yolinux.com/Tutorials/LinuxTutorialIptablesNetworkGateway.html>.
- Korff, Yanek, Hope, Paco, Potter, Bruce, (2005). *Mastering Free BSD and Open BSD Security*. O'Reilly Media, Inc, page 314.
- Liang, Chuck, (2000), *A Course on TCP/IP Networking with Linux*, Consortium for Computing Sciences in Colleges, Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges, pages 256-264
- McKenna, Adam, *The unofficial Linux ipchains-HOWTO (beta)*. Retrieved September 19, 2005 from <http://www.flounder.net/ipchains/ipchains-howto.html>.

Napier, Duncan (2001) Iptables/Netfilter – Linux’s Next-Generation Stateful Packet Filter, *SysAdmin Magazine*, retrieved September 23, 2005 from

www.samag.com/documents/s=1769/sam0112a/0112a.htm

Netfilter/iptables. Retrieved September 19, 2005. from

<http://www.en.wikipedia.org/wiki/iptables>.

Shimonski, Robert, Littlejohn Shinder, Debra, Shinder, Dr. Thomas, Carasik-Hemmi, Anne, (2003). *Best Damn Firewall Book Period*. Syngress Publishing, pp 128, 143, 146, 148, 149, 151, 171.

Suehring, Steve, Ziegler, Robert, (2005). *Linux Firewalls Third Edition*. Novell Press, page 64.

Weiss, Todd (2001). OpenBSD drops firewall program in licensing dispute. *Computerworld*.

October 20, 2005, from <http://www.computerworld.com>.

Zwicky, Elizabeth, & Cooper, Simon, & Chapman, D. Brent, (2000), *Building Internet Firewalls, second edition*, O’Reilly & Associates, pages 203-223