Global Information Assurance Certification Paper

Ipchains and IP Masquerading
The Semi-Stateful Packet Filtering Firewall for All of Us Poor Folks
George Bakos
October 2, 2000

Now I know you have all heard the wonderful news about these things called "firewalls", no? Those miracle devices that will protect your home & loved ones from the bad old boogiemen just waiting for you to connect your network to the wild-wild-west of the Internet. Ahh yes, firewalls. $30,000.00 worth of steel & octets, keeping your business a'thriving, your coffee pot percolating and your children's French grades just above average. Well, maybe not French. The truth is, of course, this is all bunk. A firewall is not the silver bullet we have all been waiting to find. Instead, it is just one component in a multi-tiered security architecture. The military types like to call this a "defense-in-depth". Oh, and they don't need to be expensive; how does free sound for you? If you have an old 486, Pentium, Sparc or Alpha machine laying around and don't mind getting your fingers dirty under the hood of Linux, have I got a firewall for you.

It is pretty easy to identify the three prime factors that we all look for in computer security components:

1. They need to do what we need (or want)
2. They need to be easy to set up and administer
3. They shouldn't be too expensive.

The businessmen out there are grumbling to themselves right now: *"return on investment"* and *"total cost of ownership".* That's what I just said.

You can have any two of the above. In all reality, these three factors are each on their own sliding scale. Generally, if something comes with a really pretty interface (just point 'n' click and you're secure!) and is really expensive, it should do a good job. On the other hand, if a firewall is secure, and super cheap, you'll probably need to understand a little more about its internals to get it up and running, and keep things in ship shape. Every organization needs to evaluate any investment against these factors before committing time, money, and most importantly, security, to it. Ipchains has several of these on its side, the least of which is certainly not its purchase price. For many small networks, Ipchains is a pretty good balance of these three factors. Let's get down to business.

What is a firewall?

"Firewall" is an interesting term. In an automobile, it keeps you from getting burned, as long as the fire doesn't get too hot and its a well made firewall. The same holds true in computer networks. A firewall, if properly configured, will filter traffic based on a set of rules to allow only certain types of information to flow into, out of, and through it. The challenge, of course, is to identify those traffic types and keep your firewall ruleset(s) up to date. No firewall, however, even if properly configured, is impenetrable! Yes, the fire can and does get that hot. Don't put all your eggs in one basket.
In the glorious days of yore, we had nothing more than routers with a little more smarts that their non-filtering brethren. Think of data as a gift. To be transported from hither to yon, it needs to be packaged a number of times; first, it needs to be wrapped and labeled so that the recipient recognizes it as a gift, knows what it is for, and realizes that it is for them. Second, it needs to be packaged so that the post office will know what to do with it and it won't get damaged in transit. Then it goes in a mailbag, etc., etc. We call this process "encapsulation". At the other end, it comes out of the various layers of encapsulation until your nephew Eddie has his shiny new firetruck in his hands. And breaks it. I digress. Remember the routers I mentioned? They make decisions based on one layer of that encapsulation: the network layer. That's where the IP address resides. The router looks at that address, compares it with its routing tables, and repackages it to go back onto the wire to the next router along the route until it gets to its final destination. The earliest filtering routers, however, looked at another piece of information, the source (or return) address. They could decide that traffic coming from the sophomore chemistry lab could never find its way into the Dean's transcripts database. Cisco calls this "standard access lists". Good stuff, but not good enough.
If we peel off the layer with the addressing on it, the "network layer", we find the "Transport Layer": information that identifies the application that will process it (destination port), the application that sent the data or request for data (source port), some error checking and, in the case of TCP, sequence numbers that keep everything in the right order, as well as binary flags, or "code bits" that identify the purpose of each individual packet. For a good discussion of TCP/IP, go to Daryl's TCP/IP Primer. Modern packet filtering firewalls, including many good router software packages, can either permit or block data based on these ports & flags as well as source and destination address.

The first description of a modern firewall, although he didn't use that term, was by Bill Cheswick in 1990.

What is a "stateful" filter?

A simple packet filter, like ipchains, has a list of rules that all traffic, incoming and outgoing, is compared against. Here's an example of an ipchains ruleset:

input -i eth1 -s 0.0.0.0/0 1024:65535 -d 0.0.0.0/0 80 -p tcp -j ACCEPT

input -i eth0 -s 0.0.0.0/0 80 -d 0.0.0.0/0 1024:65535 -p tcp ! --syn -j ACCEPT

input -j REJECT -l

The first rule will evaluate traffic coming into the firewall on its inside (trusted) interface "eth1" from any source address, with a source port above 1023, most likely a client process. If it is headed for any address on the untrusted network listening on a port 80, and the Transport layer protocol is TCP, then the assumption is that this is a normal web request and the traffic will be accepted and passed along. The second rule is precisely its opposite. It will allow traffic into the firewall from the untrusted interface, "eth0" to come back through the firewall to the client, as long as the "syn" bit is not (!) set. A syn flag should only be set if the source is requesting an initial connection. The firewall in this example would not accept it; there might be someone trying to do bad things! The third rule then bounces away all other traffic, returning a very friendly ICMP message: "destination port unreachable", and making an entry in the system log (-l).

A stateful filter however, needs no such rules that allow return traffic back in. When the outgoing request is made from the trusted side of the network, the firewall then begins listening for a sensible reply. If it doesn't come within a certain time, the dynamically created rule is dropped and no traffic is allowed through. If the reply does come back, and the connection exits either gracefully or abruptly (FIN exchange or RST), the incoming rule is torn down and again, we have no holes from the outside in. This is a great improvement over a static or "dumb" packet filter; a static ruleset may allow traffic to penetrate the firewall if it is crafted in such a way as to appear normal. Given the above example ruleset, a "remote administration" trojan horse residing on the trusted network could listen for incoming SYN/ACK (normal response to a connection request), instead of traditional SYN, packets on oh, say, port 23456. If the attacker sent these crafted packets from his port 80 they would meet all the requirements of the second firewall rule and pass right on through.

Stateful filters, such as the Cisco PIX and Lucent Managed Firewall build a "state table" of connections precisely identifying the addresses, ports, tcp sequence numbers and flags associated with each new outgoing request. Incoming packets are then analyzed against this table to verify that a corresponding connection exists before they are allowed to pass. The above example trojan traffic would not match any existing state table entry, and as such would be blocked and, hopefully, logged.

But I thought ipchains was a dumb packet filter.

It is, but there is a twist. There is a technology known as NAT, Network Address Translation, which allows a router or firewall to intercept all traffic incoming or outgoing and translate all private trusted-side network addresses into publicly accessible ones. This serves a number of functions. One, it effectively hides a network's architecture from prying eyes (and scanners). Two, it allows either one-to-one mapping of internal to external addresses, or one-to-many. This one-to-many allows an entire network, or portion of it, to appear to the rest of the world as if it were only one host - the firewall/router! Now let's take this one-to-many thing a little further: NAT benefit number three, by using appearing as only one address to the outside world, and dynamically setting up and tearing down this one-to-many relationship, the likelihood of an attacker accessing the private internal address space without first root compromising the firewall is greatly reduced.
Linux IP Masquerading is a subset of NAT that provides no mechanism for one-to-one "static" mapping of internal to external addresses. The only available mapping is dynamic. The firewall presents one address to the untrusted network and mangles the packets as they pass through so as to keep the trusted network hidden.

How does it work?

Ipchains, unlike many firewalls, provides a mechanism for acting on packets as they traverse the firewall internally, from one interface to another. Thus, instead of being limited to merely filtering on input and output, there is an additional "chain" of rules known as FORWARD. This is where we configure the automagic packet mangling. Let's try this:

NOTE: ALL SYNTAX HERE IS ABBREVIATED FOR CLARITY. READ THE MAN PAGES

AND IPMASQ HOWTO BEFORE ATTEMPTING TO ROLL YOUR OWN!

:input REJECT ****

:forward REJECT **)---Here we set our policies in case a packet falls off the end of a chain.**

:output ACCEPT **/**

:evil - **|---Now we create some custom chains to help**

:nice - **| keep things organized and easy to follow**

input -i ppp0 -j evil **|---If it is coming in over the DSL (untrusted), jump to the "evil" chain**

input -i eth0 -j nice **|---If it is coming from the inside (trusted), jump to "nice"**

input -p tcp -j REJECT -l **\ If a packet isn't matched in the custom chains,**

input -p udp -j REJECT -l **)---the pattern matching returns back here so we**

input -p icmp -j REJECT -l **/ can kill it and log it!**

forward -s 192.168.1.0/255.255.255.0 -j MASQ **|---Once input is done, apply IP Masquerading**

evil -s 128.59.64.60 123 -d 0/0 123 -p udp -j ACCEPT **|---NTP traffic is OK from these two servers**

evil -s 199.212.17.35 123 -d 0/0 123 -p udp -j ACCEPT **|**

evil --destination-port 22 -p tcp -j ACCEPT **|---SSH is OK coming in**

evil --source-port 53 --destination-port 1024:65535 -p udp -j ACCEPT **|---DNS replies are fine**

evil --source-port 53 --destination-port 1024:65535 -p tcp -j ACCEPT ! -y **|---TCP DNS replies are OK too**

evil --source-port 25 --destination-port 1024:65535 -p tcp -j ACCEPT ! -y **|---SMTP replies are OK**

evil --destination-port 61000:65096 -p tcp -j ACCEPT ! -y **|---TCP replies to the Masq. ports are OK**

evil --destination-port -p icmp -j ACCEPT ! -f **|---Non-fragmented ICMP is (usually) normal. OK here**

nice -s 192.168.1.0/24 -i eth0 -j ACCEPT **|---I trust myself (Please don't flame me over this!)**

This is a simple ruleset for home use across a DSL that is always up. When bad guys want to attack the internal network, they hit a brick wall unless there is an existing connection that they can hijack. Let's take a look at the masq (NAT) table with a few connections:

```
[root@bunta /root]# /sbin/ipchains -L -M -n

IP masquerading entries

prot expire   source         destination       ports

TCP  01:18.99 192.168.1.4    209.100.46.7      4300 (61882) -> 31632

TCP  14:15.89 192.168.1.4    216.33.46.174     4298 (61880) -> 80

TCP  00:15.57 192.168.1.4    216.33.46.173     4297 (61879) -> 80

ICMP 00:33.46 192.168.1.12   208.225.201.200   256 (61883) -> 8

TCP  14:18.68 192.168.1.4    209.100.46.7      4299 (61881) -> 21

ICMP 00:59.96 192.168.1.12   208.184.74.98     256 (61884) -> 8

TCP  00:22.87 192.168.1.4    167.216.133.33    4288 (61862) -> 80

TCP  01:13.48 192.168.1.4    167.216.133.33    4289 (61864) -> 80

TCP  01:13.55 192.168.1.4    209.100.46.1      4293 (61874) -> 80

TCP  01:12.27 192.168.1.4    209.100.46.1      4292 (61872) -> 80

TCP  00:26.34 192.168.1.12   192.88.209.22     1047 (61865) -> 80
```

In the first entry, we see a TCP entry that will keep listening for another 1 minute and 18.99 seconds before it closes down. The Source host that initiated the connection is 192.168.222.4 and its src port is 4300. When the packet was intercepted by the firewall, the source port and address were stripped off the packet and replaced by the firewall's public address, 123.45.67.8, and a port from the available pool; in this case 61882. Here is what the packets look like:

192.168.222.4:4300	->	209.100.46.7:31632
	becomes	
123.45.67.8: 61882	->	209.100.46.7:31632

Now when the untrusted partner replies, the firewall first applies the input (& evil) ruleset, if it passes the input test, it then compares the incoming packet's address/port pairs to the masq table and decides whether or not to de-masquerade it. If so, the following happens:

209.100.46.7:31632	->	123.45.67.8: 61882
	becomes	
209.100.46.7:31632	->	192.168.222.4:4300

Thus, the traffic requested by our trusted host makes it safe and sound, and the masq table entry is flushed. Pretty slick, eh? A few more details need to be mentioned. IP Masq lacks the more advanced features available in some commercial stateful filters such as sequence number checking; a framework is in place for this, but it was never fully implemented. Additionally, be aware of the timeout values and adjust them for your site's needs. This is important as they will keep a hole open for that duration unless something signals the end of a transaction. No such mechanism exists in IP Masq for UDP or ICMP! That means that a hole in your perimeter defense will stick around for the duration of the timeout value every time a UDP or TCP packet leaves through the firewall. Hmmmmm. We'll get back to that.

Is it a sure thing?

Absolutely! Oh, while you're listening; can I interest you in some prime real estate? Actually, for many installations this is a very effective approach and extremely cost effective. Remember that *"return on investment"* stuff? I have personally had a production environment pushing 22 very busy workstations and 8 even busier servers across a T1 through a single PPro200 Ipchains / IP box with 64 Mb of RAM and no noticeable loss or corruption. According to the HOWTO maintainer, David Ranch, "MASQ has also be known run quite well on 386SX-16s with 8MB of RAM". There are additional modules available to give IP Masq the smarts to handle many "negotiated port" protocols such as non-PASV FTP, RealAudio, CuSeeme, VDOLive, etc. By default, when IP Masq is compiled into the Linux kernel, there are only 4096 ports defined for masquerading. This limits it to small networks, as a busy host can easily tie up over one hundred of those, particularly if connections are left hanging open. If you want to push beyond those limits and recompile your kernel to support that, please let me know how far you can stretch your milage. I've heard that IP Masq should be happy with up to 32,000 ports, but at over 500 NAT table entries, it gets rather skittish.

How can it be defeated?

All firewalls, like I mentioned earlier, are only as good as their configuration. Networks are in the same boat. Unfortunately, the greatest threat to a secure network behind the best of firewalls, is that ever-challanging "keyboard actuator mechanism", otherwise known as the "physical mouse driver". Users launching email attachments, failing to adhere to good password policies & practices, plugging modems in while connected to the LAN, using floppies that they got at the library, etc. can negate your beautiful security architecture in a flash. But we already know this and have fixed our policies, made an example out of a token offender, and these problems are no more, right? And we have sufficiently hardened our firewall platform, too? Yes you do need to get that SUID Xbill off of there, too! Good, now we can discuss a few issues particular to Ipchains & IP Masq.

Fragments, what lovely little beasties. When an IP datagram is passed down to the next layer, the Link layer, there's a chance that the Link layer may insist that it is too large to frame. This limit is known as the Maximum Transmission unit, or MTU. The IP will then, unless instructed otherwise, break the datagram into fragments to be independantly fired down the wire. Upon arrival at the ultimate destination, they are reassembled at the Network (IP) Layer before being passed up to the Transport Layer. Wait a minute, up top we said that the packetfiltering firewall performed its examination at the Network AND Transport layers, no? If the final destination for a packet is on the trusted network, and the fragment is small enough so that it doesn't contain the entire transport header, then our firewall won't have the opportunity to see if it is a nasty or not! Ipchains gives us the opportunity to match fragments with a "-f" condition, but what does that really mean? From IPCHAINS(8): "the rule only refers to second and furthur fragments of fragmented packets." That's all well and good, but what if some realy mean person were to craft two fragments, neither appearing to be "second and further", that would overlap upon reassembly on the target machine? It may result in a packet with transport layer settings that would have normally been blocked. This is precisely what came to light in July of 1999 with Linux kernel 2.2.10. There is a patch available, but better still, you should compile the kernel on the firewall with CONFIG_IP_ALWAYS_DEFRAG, or in kernels 2.2.12 and later just include the following line in your firewall.rc script:
echo "1" > /proc/sys/net/ipv4/ip_ip_always_defrag . Thus all fragments get reassembled before being examined and passed inside.

OK, that pertained to packets addressed to boxes behind the firewall. That can't happen with IP Masq, right? Wrong! Remember earlier we talked about those timeout values? This is where they bite you in the behind. Envision another nasty attacker (my Goodness there are a lot of them!) who has come to discover that you are masquerading. Its not too hard; just look for the boxes making client requests with source ports between 61000 and 65096. Hmmmm, maybe you might want to change that. OK, he knows what you've got, what can he do with those excessive UDP timeouts? Well, believe it or not, IP Masqerading has a problem here. According to a Bugtraq post by H.D.Moore in March 2000, it is fairly simple for an attacker to identify which masq port is currently open and then hijack it, overwriting the NAT table entry with his own address and port. This would allow him an avenue of approach to an internal host. The Bugtraq thread is archive at Neohapsis, as well as plenty of other sites. There has been much discussion regarding this, but no exploit has yet been demonstrated. If you don't do ICQ, you should be able to safely throttle back your UDP timeout to 30 seconds, minimizing your vulnerability.

OK, it sounds pretty nifty. Where can I get one?

Run down to your local McD's and order a really big burger, since you don't need to spend any of your hard-earned sheckles on this one. Linux and Ipchains/IP Masquerading are free, unless you want to pay a vendor for support and cds. If you are unfamiliar with Linux, go to http://www.linux.org and fix that. Ipchains and IP Masquerading are kernel implementaions and should be enabled by default in most modern distributions. Be sure to visit the links below and get a solid grounding from the experts before trusting your organization's network to anything. And remember, if

you doubt the virtue of those who provide such gifts, "Read the source, Luke."

Additional resources

Banttari, Daryl. "Daryl's TCP/IP Primer.", 7 APR 2000
URL - http://ipprimer.windsorcs.com (2 Oct 2000)

Cisco Systems. "Cisco's PIX Firewall and Stateful Firewall Security White Paper", 1996
URL - http://www.cisco.com/warp/customer/cc/pd/fw/sqfw500/tech/nat_wp.htm (2 Oct 2000)

Cheswick, Bill. "The Design of a Secure Internet Gateway", 1990 USENIX Summer Conference
URL - http://www.alw.nih.gov/Security/FIRST/papers/firewall/gateway.ps (2 Oct 2000)

"COAST Hotlist - Internet Firewalls" Purdue University
URL - http://www.cerias.purdue.edu/coast/firewalls (2 Oct 2000)

Au, Ambrose. and Ranch, David. "Linux IP Masquerade Resource"
URL - http://ipmasq.cjb.net (2 Oct 2000)

Hasenstein, Michael. "IP NETWORK ADDRESS TRANSLATION", 1997
URL - http://www.suse.de/~mha/HyperNews/get/linux-ip-nat.html (2 Oct 2000)

Moore, H.D. "Security Problems with Linux 2.2.x IP Masquerading", 27 Mar 2000
URL - http://archives.neohapsis.com/archives/bugtraq/2000-03/0284.html (2 Oct 2000)