Linux IPMASQ and Firewalls for the Small Office or Home

Dillon Pyron
Sprint Paranet, Austin, TX
dmpyron@sprintparanet.com

What is IP Masquerading?

IP Masquerading is a technique that allows a private network to share one common, public IP address. This forms a "one to many" relationship with the Internet. This permits the private network to hide behind a router/gateway machine, but still have access to the Internet as if they were part of it with legitimate, assigned IP addresses. In fact, they do, but the nodes share the address in question. IP Masquerading (known as IP Masqing) is a form of Network Address Translation (NAT) unique to Linux.

An added benefit of IP Masqing is that its nature allows for a very secure firewall structure. When properly configured, an IP Masqed firewall is quite stout and resistant to most attacks. The combination of these two features makes IP Masqing very attractive to the home or small office environment. As more communities offer "continuous on" network connections such as cable modems or xDSL, the security feature alone is worth the cost (minimal) and labor (minimal).

IP Masqing does not typically require a high performance system. Because of its nature, Linux runs happily on what would normally (Windows-wise) be considered small, slow boxes. Although a 386 can support a modest IP Masq system, a 486/66 with 16Mbytes and a 1Gbyte disk will perform very well for some time to come. And, because of the overall requirements placed on it, the machine need not have a large number of "goo gaws".

How IP Masqing works

IP Masqing depends on two components that allow it to function. The first is the public to private network mapping, which is defined by RFC 1918. The second is IP forwarding, a function that works with the mapping to translate an address and forward the resulting packet to the proper location.

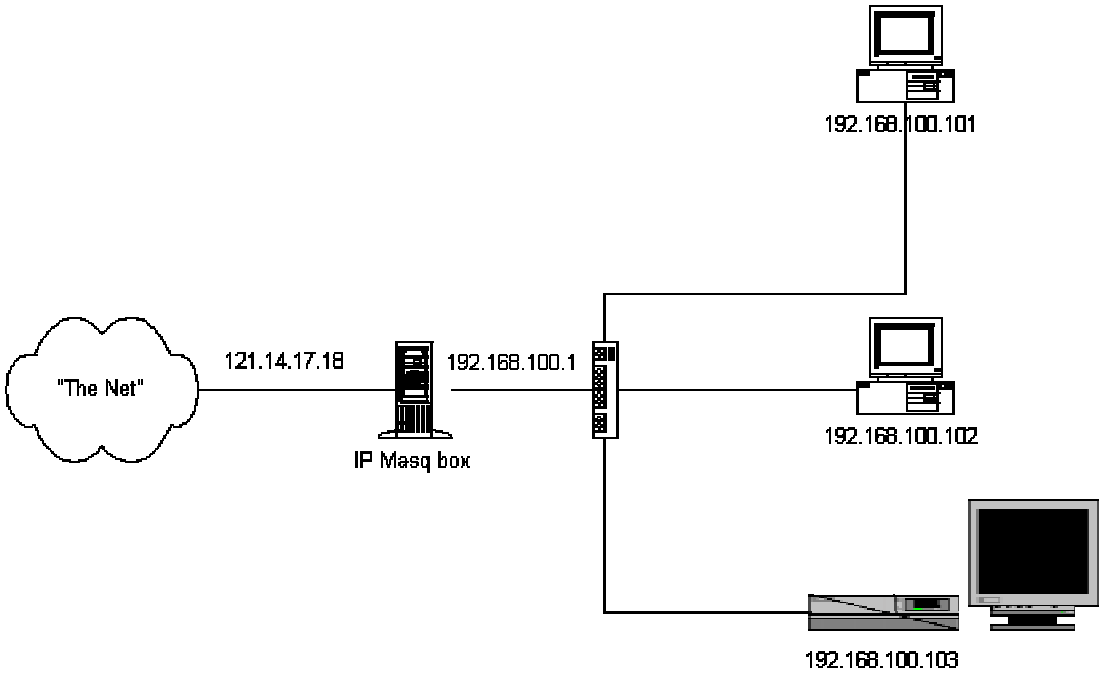
RFC 1918

Request For Comments (RFC) 1918 (Address Allocation for Private Internets) allocates a (pre-CIDR) Class A, Class B and Class C address space for use as private networks. By using these well defined networks behind a mapper such as IP Masq, a private network can be built and managed and still have access to the broader network. The three address spaces are 10.0.0.0 (10/8), 172.16.0.0 (172.16/12) and 192.168.0.0 (192.168/16).

IP forwarding

Step one, think of the IP Masquerade box as a "black box". Data goes in, something wonderful happens, and data comes out. Step two, look inside the box.

As described earlier, the configuration of the IP Masq environment has an internal network, with each machine assigned a unique IP address consistent with RFC 1918, a gateway box with both an internal address and a valid external address, and "the world". One configuration might look like this:



The two PCs and the HP 715 are connected to the IP Masq box via a simple hub (the “hub” can be anything from an inexpensive hub to a large, managed switch). The IP Masq box is connected to “The Net” by any one of a variety of options from dial-up 56K to xDSL or cable modem.

To actually communicate with the outside world, the node uses the IP Masq box as a router. The Masq system takes the packet, replaces the original source IP address with its own. It then substitutes a unique port number and stores the original address and port in a table for future reference. When the packet comes back from the other end, the Masq box replaces the port and source address with the originals. As far as the original node (inside the private network) knows, or cares, it has had a direct connection to the network.

IP Masq configuration in Linux

IP Masquerading requires no additional software, although you will need to configure both the Linux kernel and the actual IP Masqing will need to be setup for both IP forwarding and firewalling.

Compiling Linux to support IP Masq

Although IP Masquerading has been supported in Linux since 1.3.x, version 2.2 has been a stable release and will be the basis for this discussion. Due to a bug, it is recommended that you use a 2.2.11 or later. When you compile the kernel, you will need to answer the following questions (in addition to others). If you’ve never built a Linux kernel before, don’t panic. It’s easy to do.

Answer YES to the following questions:

```
Prompt for development and/or incomplete code/drivers
(CONFIG_EXPERIMENTAL) [Y/n/?]
Enable loadable module support (CONFIG_MODULES) [Y/n/?]
Networking support (CONFIG_NET) [Y/n/?]
Packet socket (CONFIG_PACKET) [Y/m/n/?]
Kernel/User netlink socket (CONFIG_NETLINK) [Y/n/?]
Routing messages (CONFIG_RTNETLINK) [Y/n/?]
Network firewalls (CONFIG_FIREWALL) [Y/n/?]
TCP/IP networking (CONFIG_INET) [Y/n/?]
IP: verbose route monitoring (CONFIG_IP_ROUTE_VERBOSE) [Y/n/?]
```

```
IP: firewalling (CONFIG_IP_FIREWALL) [Y/n/?]
IP: firewall packet netlink device (CONFIG_IP_FIREWALL_NETLINK) [Y/n/?]
IP: always defragment (required for masquerading)
(CONFIG_IP_ALWAYS_DEFRAG) [Y/n/?]
IP: masquerading (CONFIG_IP_MASQUERADE) [Y/n/?]
IP: ICMP masquerading (CONFIG_IP_MASQUERADE_ICMP) [Y/n/?]
IP: masquerading special modules support (CONFIG_IP_MASQUERADE_MOD)
[Y/n/?]
IP: ipportfw masq support (EXPERIMENTAL)
(CONFIG_IP_MASQUERADE_IPPORTFW) [Y/m/n/?]
IP: optimize as router not host (CONFIG_IP_ROUTER) [Y/n/?]
IP: TCP syncookie support (not enabled per default)
(CONFIG_SYN_COOKIES) [Y/n/?]
Network device support (CONFIG_NETDEVICES) [Y/n/?]
Dummy net driver support (CONFIG_DUMMY) [M/n/y/?]
/proc filesystem support (CONFIG_PROC_FS) [Y/n/?]
```

Answer NO to the following (although you may need to evaluate whether you need them for other support, such as the GRE tunneling):

```
IP: advanced router (CONFIG_IP_ADVANCED_ROUTER) [Y/n/?]
IP: ipautofw masq support (EXPERIMENTAL)
(CONFIG_IP_MASQUERADE_IPAUTOFW) [N/y/m/?]
IP: ip fwmark masq-forwarding support (EXPERIMENTAL)
(CONFIG_IP_MASQUERADE_MFW) [Y/m/n/?]
IP: GRE tunnels over IP (CONFIG_NET_IPGRE) [N/y/m/?]
```

Next, you will need to compile the kernel and install the modules:

```
make modules; make modules_install
```

Host name translation

Remember, your internal nodes are private, with IP addresses that are not visible (or had better not be!) to the public network. There are several options on how to translate those names to IP addresses. In theory, the easiest way to do this is to keep hostname tables on each machine. However you have to keep all of the files in sync. NIS would be an option, if all of your computers supported it (hey, Redmond!!!). That leaves DHCP and DNS. Since I have little actual experience in setting up a DHCP server, I use DNS. As a little extra paranoia, my DNS server is not my IP Masquerade system.

Building a firewall with IP Masq

As interesting as the routing and NAT capabilities of IP Masq, for most people the firewall aspects are the most attractive part. A powerful, high performance firewall can be built with relatively low performance and even “obsolete” computers. Although not in the same category as dedicated firewalls such as a Cisco Pix, they provide a sufficiently stout front door to encourage all but the most dedicated attacker to “go elsewhere”.

Firewall basics

IP Masquerading firewalls can be thought of as a set of rules that control how and which packets are forwarded and which packets are accepted and passed into the private net. The command `ipchains` is used to define and control this process. A few of the commands are listed below, along with what they do. The general concept in this is something referred to as “strong rulesets”. The purpose is to establish a set of

rules that are initially very restrictive, and then relax access as required. To do this, you first need to set a default policy of rejection, then allow where needed. So, to start, go with:

```
ipchains –P input REJECT
ipchains –P output REJECT
```

The next critical step is to prevent unwanted access from the outside. Cut off spoofing but allow access to certain other nodes. The commands are

```
ipchains –A <rulename> –i <interface> -s <source address> -d <dest> \
ACCEPT|REJECT

For instance:
ipchains –A input –i eth0 –s 208.25.106.43 –d 192.168.100.102 ACCEPT

appends to a ruleset call “input” and allows the IP address
208.25.106.43 (the Sprint Paranet WWW server) to have input access to
one of the local machines, the PC with IP address of 192.168.100.102
```

Determining allowable ports and services

One facet that must be allowed for is the use of specific ports to a machine. This can be accomplished by appending the port to the IP address, as in 192.168.100.102:23 provide telnet access to node 192.168.100.102. However, this access path can defeat the entire purpose of your firewall if incorrectly constructed. Because IP Masquerading is intended to be a transparent system, port translation needs to be undertaken very carefully.

Comparing IP Masq to other proxy/router/firewall products

IP Masquerading is far from the only product to offer the services here. What follows is a small list of products, with a comparison of the pros and cons of each.

Dedicated hardware

Performance-wise, IP Masq is not comparable to dedicated firewall products such as Cisco’s PIX. However, the cost and configuration expenses are much lower with IP Masq. Other products, such as Checkpoint’s Firewall-1, offer a much higher level of security, at a significant cost. While not unreasonable for a moderate to large corporation, but well out of reach of most small companies or individuals.

Proxy servers

A proxy server uses only (1) public IP address, like IP MASQ, and acts as a translator to clients on the private LAN (WWW browser, etc.). This proxy server receives requests from the private network on one interface. It then in turn, initiates these requests as if someone on the local box was making the requests. Once the remote Internet server sends back the requested information, it then re-translates the TCP/IP addresses back to the internal MASQ client and sends traffic to the internal requesting host. This is why it is called a PROXY server.

Unfortunately, each application that uses a proxy server must be cognizant of the service and configured to use it. Applications such as Netscape do, in fact, do this. But not all applications are written with this in mind. Also, unless it is otherwise built in, proxy servers do not provide any firewall security.

Some of the superior proxy servers also cache certain types of requests (specifically, WWW traffic). This can optimize some applications performance because the server does not need to go out to the Internet to complete the transaction

NAT servers

Network Address Translation is a name for a box that has a pool of valid IP addresses on the Internet interface that it can use. When a node on the internal network wants to go to the Internet, the NAT box associates an available valid IP address from the Internet interface to the original requesting private IP address. After that, all traffic is re-written from the NAT public IP address to the NAT private address. Once the associated public NAT address becomes idle for some pre-determined amount of time, the public IP address is returned back into the public NAT pool. The problem with this is that there is a one to one relationship between publicly known (ie, valid) IP addresses, and the number of simultaneously active Internet connections. If this number is exceeded, the requesting nodes are unable to get to the Internet. In addition, there is not consideration for security built into the NAT service.

Specialized versions of IP Masq

Finally, there are specialized versions of IP Masq. One of these is a dedicated Linux that boots and runs off of a single floppy. The product, from Share The Net (www.sharethenet.com) is a fully configured IP Masq system with a minimal Linux environment. One of the interesting things that results from this is that a computer can run both the IP Masq (Share The Net) system and another OS, by simply rebooting and changing the status of the floppy.

Special Note

Of course, as time goes by, things change. Since I started writing this paper, the 2.4 kernel has been released, with a replacement for IPCHAINS. This paper was written based on the 2.3.11 kernel and IPCHAINS. NETFILTER (the replacement) has a similar syntax although the changes are worth noting before making the conversion.

References

The Linux IP Masquerade HOW-TO (http://members.home.net/ipmasq/ipmasq-HOWTO-1.80-3.html)
David Ranch's Home Page (http://www.ecst.csuchico.edu/%7Edranch/LINUX/index-linux.html#ipmasq)
The Official IP Masquerade Home Page (http://members.home.net/ipmasq)
The IP Masq mailing list