Firewall and Proxy Server HOWTO

Table of Contents

Table of Contents

Firewall and Proxy Server HOWTO

Mark Grennan, `mark@grennan.com`

v0.80, Feb. 26, 2000

This document is designed to describe the basics of firewall systems and give you some detail on setting up both a filtering and proxy firewall on a Linux based system. An HTML version of this document is available at http://www.grennan.com/Firewall−HOWTO.html

---

5.Software requirements

6.Preparing the Linux system

7.IP filtering setup (IPFWADM)

8.IP filtering setup (IPCHAINS)

9.Installing a Transparent SQUID proxy

10.Installing the TIS Proxy server

11.The SOCKS Proxy Server

1.Introduction

David Rudder wrote this original version of this Firewall−HOWTO, these many moons ago, and I'd still like to thank him for allowing me to update his work.

I'd also like to thank Ian Gough for kindly assisting a this dislexic writer.

Firewalls have gained great popularity as the ultimate in Internet Security. Like most hot subject they are also often misunderstood. This HOWTO will go over the basics of what a firewall is and how to set one up.

I am using kernel 2.2.13 and RedHat 6.1 to develop this howto so the examples here are based on this distribution. If you find differences in your distribution, please email me and I'll update this howto.

1.1 Feedback

Any feedback is very welcome. **PLEASE REPORT ANY INACCURACIES IN THIS PAPER!!!** I am human, and prone to making mistakes. If you find a fix for anything please send it to me. I will try to answer all e−mail, but I am busy, so don't get insulted if I don't.

My email address is ***mark@grennan.com***

1.2 Disclaimer

I AM NOT RESPONSIBLE FOR ANY DAMAGES INCURRED DUE TO ACTIONS TAKEN BASED ON THIS DOCUMENT. This document is meant as an introduction to how firewalls and proxy servers work. I am not, nor do I pretend to be, a security expert. ;−) I am just some guy who has read too much and likes computers more than most people. Please, I am writing this to help people get acquainted with this subject, and I am not ready to stake my life on the accuracy of what is in here.

1.3 Copyright

If you have any questions, please email me. (See Above)

1.4 My Reasons for Writing this

Several years ago, while working for the State of Oklahoma as their "Internet Administrator" I was ask to "put the State on the Internet", with no budget. (Note: There was no such title at the time. I was just the guy doing all the work.) The best way to make this happen was to use as much free software and junk hardware as I could. Linux and a bunch of old 486s were all I had to work with.

Commercial firewalls are VERY over priced and the documentation on how they work is considered almost top secret. I found creating a firewall of my own was almost impossible.

At my next job, I was asked to put in a firewall. Linux had just added firewall code. So again with no budget I started building a firewall with Linux. Six months later my firewall was in place and this document was updated.

1.5 Further Readings

- The The Linux Networking Overview HOWTO
- The Ethernet HOWTO
- IPchains Firewalling made Easy!
- Linux Network Address Translation
- The Net−3 HOWTO
- The NET−PPP HOWTO
- The easiest way to create Virtual Tunnels over TCP/IP networks

[More URLS go here]

2.Understanding Firewalls

A firewall is a structure intended to keep a fire from spreading. Building have firewalls made of brick walls completely dividing sections of the building. In a car a firewall is the metal wall separating the engine and passenger compartments.

Internet firewalls are intended to keep the flames of Internet hell out of your private LAN. Or, to keep the members of your LAN pure and chaste by denying them access the all the evil Internet temptations. ;−)

The first computer firewall was a non−routing Unix host with connections to two different networks. One network card connected to the Internet and the other to the private LAN. To reach the Internet from the private network, you had to logon to the firewall (Unix) server. You then used the resources of the system to access the Internet. For example, you could use X−windows to run Netscape's browser on the firewall system and have the display on your work station. With the browser running on the firewall it has access to both networks.

This sort of dual homed system (a system with two network connections) is great if you can TRUST ALL of your users. You can simple setup a Linux system and give an account accounts on it to everyone needing Internet access. With this setup, the only computer on your private network that knows anything about the outside world is the firewall. No one can download to their personal workstations. They must first download a file to the firewall and then download the file from the firewall to their workstation.

BIG NOTE: 99% of all break−ins start with gaining account level access on the system being attacked. Because of this I don't recommend this type of firewall. It is also very limiting.

2.1 Firewall Politics

You shouldn't believe a firewall machine is all you need. *Set policies first*.

Firewalls are used for two purposes.

1. to keep people (worms / crackers) out.
2. to keep people (employees / children) in.

When I started working on firewalls I was surprised to learn the company I worked for were more interested in "spying" on their employees then keeping crackers out of their networks.

At least in my state (Oklahoma) employers have the right to monitor phone calls and Internet activity as long as they inform the employees they are doing it.

Big Brother is not government. Big Brother = Big Business.

Don't get me wrong. People should work, not play at work. And I feel the work ethic has been eroding. However, I have also observed that management types are the biggest abusers of the rules they set. I have seen hourly workers reprimanded for using the Internet to looking for bus routesto get to work while the same manager used hours of work time looking for fine restaurants and nightclubs to take prospective customers.

My fix for this type of abuse is to publish the firewall logs on a Web page for everyone to see.

The security business can be scary. If you are the firewall manager, watch your back.

How it create a security policy

I have seen some realy high folutin documentation on how to create a security policy. After many years of experence I know now say, don't believe a word of them. Create a security policy is simple.

1. describe what you need to service
2. describe the group of people you need to service
3. describe which service each group needs access to
4. for each service group describe how the service should be keep secure
5. write a statment making all other forms of access a vialation

Your policy will become more complicated with time but don't try to cover to much ground now. Make it simple and clear.

2.2 Types of Firewalls

There are two types of firewalls.

1. Filtering Firewalls – that block selected network packets.
2. Proxy Servers (sometimes called firewalls) – that make network connections for you.

Packet Filtering Firewalls

Packet Filtering is the type of firewall built into the Linux kernel.

A filtering firewall works at the network level. Data is only allowed to leave the system if the firewall rules allow it. As packets arrive they are filtered by their type, source address, destination address, and port information contained in each packet.

Many network routers have the ability to perform some firewall services. Filtering firewalls can be thought of as a type of router. Because of this you need a deep understanding of IP packet structure to work with one.

Because very little data is analyzed and logged, filtering firewalls take less CPU and create less latency in your network.

Filtering firewalls do not provide for password controls. User can not identify themselves. The only identity a user has is the IP number assigned to their workstation. This can be a problem if you are going to use DHCP (Dynamic IP assignments). This is because rules are based on IP numbers you will have to adjust the rules as new IP numbers are assigned. I don't know how to automate this process.

Filtering firewalls are more transparent to the user. The user does not have to setup rules in their applications to use the Internet. With most proxy servers this is not true.

Proxy Servers

Proxies are mostly used to control, or monitor, outbound traffic. Some application proxies cache the requested data. This lowers bandwidth requirements and decreases the access the same data for the next user. It also gives unquestionable evidence of what was transferred.

There are two types of proxy servers.

1. Application Proxies – that do the work for you.
2. SOCKS Proxies – that cross wire ports.

Application Proxy

The best example is a person telneting to another computer and then telneting from there to the outside world. With a application proxy server the process is automated. As you telnet to the outside world the client send you to the proxy first. The proxy then connects to the server you requested (the outside world) and returns the data to you.

Because proxy servers are handling all the communications, they can log everything they (you) do. For HTTP (web) proxies this includes very URL they you see. For FTP proxies this includes every file you download. They can even filter out "inappropriate" words from the sites you visit or scan for viruses.

Application proxy servers can authenticate users. Before a connection to the outside is made, the server can ask the user to login first. To a web user this would make every site look like it required a login.

SOCKS Proxy

A SOCKS server is a lot like an old switch board. It simply cross wires your connection through the system to another outside connection.

Most SOCKS server only work with TCP type connections. And like filtering firewalls they don't provide for user authentication. They can however record where each user connected to.

3.Firewall Architecture

There are lots of ways to structure your network to protect your systems using a firewall.

If you have a dedicated connections to the Internet through a router, you could plug the router directly into your firewall system. Or, you could go through a hub to provide for full access servers outside your firewall.

3.1 Dial−up Architecture

You may be using a dialup service like an ISDN line. In this case you might use a third network card to provide provide a filtered DMZ. This gives you full control over your Internet services and still separates them from your regular network.

```
                    __________
   _/\__/\_        |          |          _______________
  |        |       | Firewall |   (LAN)  |               |
 / Internet \----| System   |--(HUB)--| Workstation/s |
 \_  _  _  _/      |__________|          |_______________|
   \/ \/ \/             |
                      (DMZ)
```

```
                        (HUB)
```

3.2 Single Router Architecture

If there is a router or cable modem between you and the Internet. If you own the router you could setup some hard filter rules in the router. If this router is owned by your ISP so you may not the have the needed controls. You can ask your ISP to put in filters.

```
                      _________            __________
   _/\__/\_          | Router  |          |          |
  |        |         |   or    |   (DMZ)  | Firewall |   (LAN)   |  ________________
 / Internet \----|Cable Mdm|--(HUB)--|  System  |--(HUB)--|  Workstation/s  |
 \_  _  _  _/    |_________|     |     |__________|          |________________|
   \/ \/ \/                      |
                              (Outside)
                              (Server)
```

3.3 Firewall with Proxy Server

If you need to monitor where users of your network are going and your network is small, you can intergrate a proxy server into your firewall. ISP's some times do this to create interest list of their users to resell to marketing agencies.

```
                      __________
   _/\__/\_          | Proxy /  |           ________________
  |        |         | Firewall |   (LAN)  |                |
 / Internet \----|  System  |--(HUB)--| Workstation/s  |
 \_  _  _  _/    |__________|          |________________|
   \/ \/ \/
```

You can put the proxy server on your LAN as will. In this case the firewall should have rules to only allow the proxy server to connect to the Internet for the services it is providing. This way the users can get to the Internet only through the proxy.

```
                      __________
   _/\__/\_          |          |           ________________
  |        |         | Firewall |   (LAN)  |                |
 / Internet \----|  System  |--(HUB)--| Workstation/s  |
 \_  _  _  _/    |__________|     |     |________________|
   \/ \/ \/                        |      _______________
                                   |     |               |
                                   +----| Proxy Server  |
                                         |_______________|
```

3.4 Redundent Internet Configuration

If you are going to run a service like YAHOO or maybe SlashDot you may want to make your system by using redundant routers and firewalls. (Check out the High Availability HowTo.)

By using a round−robin DNS techniques to provide access to multipule web servers from one URL and multiple ISP's, routers and firewalls using High Avaibility technics you can create a 100% uptime service.

```
  _/\__/\_                                     _/\__/\_
 |        |                                   |        |
/  ISP #1  \______                 (WAN)_____/ Partners \
\_  _  _  _/      |                (HUB)     \_  _  _  _/
  \/ \/ \/        |               ___|____     \/ \/ \/
                __|___           |_______ |
  _/\__/\_     |______ |        |Firewall||          ______
 |        |   |      ||  (DMZ)  | System ||  (LAN)  |      |
/  ISP #2  \--|Router||--(HUB)--|  (VPN) ||--(HUB)--| WS/s |
\_  _  _  _/  |______|     |    |________|     |    |______|
  \/ \/ \/                 |         |         |     ______
         |             (Outside)  (Shared)     |    |      |
 ------  |             (Server)   (Server)     +----|Proxy |
| WS/s | |                                          |______|
| VPN  |-+
|______|
```

It is easy to let your network get out of hand. Keep control of every connection. It only takes a user with a modem to compromise your LAN.

4.Setting up the Linux Filtering Firewall

4.1 Hardware requirements

Filtering firewalls don't require fancy hardware. They are little more then simple routers.

All you need is:

1. a 486−DX66 with 32 meg of memory
2. a 250m hard disk (500 recommended)
3. network connections (LAN Cards, Serial Ports, Wireless?)
4. monitor and keyboard

With some systems by using a serial port console, you can even eliminate the monitor and keyboard.

If you need a proxy server that will handle lots of traffic, you should get the largest system you can afford. This is because for every user that connects to the system it will be creating another process. If you will have

50 or more concurrent users I'm guessing you will need:

1. a Pentium II with 64meg of memory
2. a two gig hard disk to store all the logs
3. two network connections
4. monitor and keyboard

The network connections can be any type (NIC cards, ISDN, even modems).

5.Software requirements

5.1 Selecting a Kernel

To create a filtering firewall, you don't need any special software. Linux will do. At the time of this writing I'm using RedHat 6.1.

The bilt in Linux firewall have changed several times. If you are using an old Linux kernel (1.0.x or older) geta new copy. These older used ipfwadm from **http://www.xos.nl/linux/ipfwadm/** and is no longer supported.

If you are using 2.2.13 or newer you will be using ipchaining as developed by **http://www.rustcorp.com/linux/ipchains/**

If you are using the newer 2.4 kernal there is a new firewall utility with more feachers. I will write about this soon.

5.2 Selecting a proxy server

If you want to setup a proxy server you will need one of these packages.

1. Squid
2. The TIS Firewall Toolkit (FWTK)
3. SOCKS

Squid is a great package and works with Linux's Transparent Proxy feature. I will be describing how to setup this server.

AT the time of this writing, Network Associates and Trusted Information System's (TIS) , have merged. So keep watching their web sites for more information about changes. Mean while, the Tool Kit can still be had at. **http://www.tis.com/research/software/**

Trusted Information System put out a collection of programs designed to facilitate firewalling. With this toolkit, you set up one daemon for each service (WWW, telnet ect.) you will be using.

6.Preparing the Linux system

Install as little of the Linux system as you can. My installation started with a server configuration and then I turn off ever un−needed service in /etc/inetd.conf. For more security you should uninstall the unneeded service.

Because most distributions don't dome with a kernel usefull to your perpose. You will need to compile your own kernal. It is best if you do this on a computer other then the firewall. If you do install a C compiler and utilities on your firewall, remove them after you have completed comfiguring your kernel.

6.1 Compiling the Kernel

Start with a clean minimal installation of your Linux distribution. The less software you have loaded the less holes, backdoors and/or bugs there will be to introduce security problems in your server.

Pick a stable kernel. I am using kernel 2.2.13 kernel for my system. So this documentation is based on it's settings.

You well need to recompile the Linux kernel with the appropriate options. If you haven't recompiled your kernel before you should read the Kernel HOWTO, the Ethernet HOWTO, and the NET−2 HOWTO.

Here are the network related setting I know work. I have marked some with a ?. If you will be using this feature, turn it on as well.

I use "make menuconfig" to edit my kernel settings.

```
<*> Packet socket
[ ] Kernel/User netlink socket
[*] Network firewalls
[ ] Socket Filtering
<*> Unix domain sockets
[*] TCP/IP networking
[ ] IP: multicasting
[*] IP: advanced router
[ ] IP: kernel level autoconfiguration
[*] IP: firewalling
[?] IP: always defragment (required for masquerading)
[?] IP: transparent proxy support
[?] IP: masquerading
--- Protocol-specific masquerading support will be built as modules.
[?] IP: ICMP masquerading
--- Protocol-specific masquerading support will be built as modules.
[ ] IP: masquerading special modules support
[*] IP: optimize as router not host
< > IP: tunneling
```

```
< > IP: GRE tunnels over IP
[?] IP: aliasing support
[*] IP: TCP syncookie support (not enabled per default)
--- (it is safe to leave these untouched)
< > IP: Reverse ARP
[*] IP: Allow large windows (not recommended if <16Mb of memory)
< > The IPv6 protocol (EXPERIMENTAL)
---
< > The IPX protocol
< > Appletalk DDP
< > CCITT X.25 Packet Layer (EXPERIMENTAL)
< > LAPB Data Link Driver (EXPERIMENTAL)
[ ] Bridging (EXPERIMENTAL)
[ ] 802.2 LLC (EXPERIMENTAL)
< > Acorn Econet/AUN protocols (EXPERIMENTAL)
< > WAN router
[ ] Fast switching (read help!)
[ ] Forwarding between high speed interfaces
[ ] PU is too slow to handle full bandwidth
QoS and/or fair queueing  --->
```

After making all the setting you need you should recompile, reinstall the kernel and reboot.

I use the command:

make dep;make clean;make bzlilo;make modules;make modules_install;init 6 to accomplish all of this in one step.

6.2 Configuring two network cards

If you have two network cards in your computer, you may need to add an append statement to your /etc/lilo.conf file to describe the IRQ and address of both cards. My lilo append statement looks like this:

```
append="ether=12,0x300,eth0 ether=15,0x340,eth1"
```

6.3 Configuring the Network Addresses

Now we arrive at the fun part of our setup. I'm not going to go deep into how to setup a LAN. Read the Networking–HOWTO to solve your problems here.

Your goal is to provide two network connection to your filtering firewall system. One on the Internet (unsecured side) and one on the LAN (secure side).

Anyway, you have a few decisions to make.

1. Will you use Real IP number or Make some up for your LAN.

2. Will your ISP assign the number or will you be using static IP numbers?

Since you don't want the internet to have access to your private network, you don't need to use "real addresses". You could just makeup addresses for your private LAN. But this is not recommended. If data gets routed out of your LAN, it might end up at another systems port.

There are a number of Internet address ranges set aside for private networks. Of these, 192.168.1.xxx, is set aside and we will use it in our examples.

You will need to use IP masquerading to make this happen. With this process the firewall will forward packets and translate them into "REAL " " IP address to travel on the Internet.

Using these non−routable IP address makes your network is more secure. Internet routers will not pass packets with these addresses.

You may want to read the IP Masquerading HOWTO at this point.

```
        24.94.1.123  __________     192.168.1.1
  _/\__/\_        \ |          | /           _______________
 |        |        \| Firewall |/           |               |
/ Internet \--------|  System  |------------| Workstation/s |
\_  _  _  _/        |__________|            |_______________|
  \/ \/ \/
```

You must have a "real" IP address to assign to your Internet network card. This address can be permanently assigned to you. (A static IP address) or it can be assigned at network connect time by the PPP process.

You assign your inside IP numbers. Like 192.168.1.1 to the LAN card. This will be your gateway IP address. You can assign all the other machines in the protected network (LAN) a number in the 192.168.1.xxx range. (192.168.1.2 through 192.168.1.254)

I use RedHat Linux. To configure the network at boot time I added a ifcfg−eth1 file in the /etc/sysconfig/network−scripts directory. You may also find a ifcfg−ppp0 or ifcfg−tr0 in this directory. These 'ifcfg−' files are used by RedHat to configure and enable your network devices at boot time. The are named after the connection type.

Here is the ifcfg−eth1 (second ehternet card) for our example;

```
DEVICE=eth1
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
GATEWAY=24.94.1.123
ONBOOT=yes
```

If you are going to use a dialup connection you will need to look at the ifcfg−ppp0 and the chat−ppp0 file. These control your PPP connection.

This ifcfg file might look like;

```
DEVICE="ppp0"
ONBOOT="yes"
USERCTL="no"
MODEMPORT="/dev/modem"
LINESPEED="115200"
PERSIST="yes"
DEFABORT="yes"
DEBUG="yes"
INITSTRING="ATZ"
DEFROUTE="yes"
HARDFLOWCTL="yes"
ESCAPECHARS="no"
PPPOPTIONS=""
PAPNAME="LoginID"
REMIP=""
NETMASK=""
IPADDR=""
MRU=""
MTU=""
DISCONNECTTIMEOUT=""
RETRYTIMEOUT="5"
BOOTPROTO="none"
```

6.4 Testing your network

Start by using the ifconfig and route commands. If you have two network cards ifconfig should look something like:

```
#ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:1620 errors:0 dropped:0 overruns:0
          TX packets:1620 errors:0 dropped:0 overruns:0
          collisions:0 txqueuelan:0

eth0      Link encap:10Mbps Ethernet  HWaddr 00:00:09:85:AC:55
          inet addr:24.94.1.123 Bcast:24.94.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1000 errors:0 dropped:0 overruns:0
          TX packets:1100 errors:0 dropped:0 overruns:0
          collisions:0 txqueuelan:0
          Interrupt:12 Base address:0x310

eth1      Link encap:10Mbps Ethernet  HWaddr 00:00:09:80:1E:D7
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1110 errors:0 dropped:0 overruns:0
          TX packets:1111 errors:0 dropped:0 overruns:0
          collisions:0 txqueuelan:0
          Interrupt:15 Base address:0x350
```

and your route table should look like:

```
#route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
24.94.1.0       *               255.255.255.0   U     1500   0       15 eth0
192.168.1.0     *               255.255.255.0   U     1500   0        0 eth1
127.0.0.0       *               255.0.0.0       U     3584   0        2 lo
default         24.94.1.123     *               UG    1500   0       72 eth0
```

Note: 24.94.1.0 is the Internet side of this firewall and 192.168.1.0 is the private (LAN) side.

You should start by making sure every computer on your LAN can ping the inside address of your firewall system. (192.168.1.1 in this example) If not, go over the NET−2 HOWTO again and work on the network some more.

Next, from the firewall, try to ping a Internet system. I use www.internic.net as my test point. If it doesn't work, try a server at your ISP. If this doesn't work some part of your Internet connection is wrong. You should be able to connect to the anywhere on the Internet from the firewall. Try looking at your default gateway setting. If you are using a dialup connection double check your user ID and Password. Reread the Net−2 HOWTO, and try again.

Now try to ping the outside address of the firewall (24.94.1.123) from a computer on your LAN. This shouldn't work. If it does, you have masquerading or IP Forwarding turned on, or you already have some packet filtering set. Turn them off and try again. You need to know the filtering is in place.

For kernels newer then 2.1.102 you can issue the command;

```
echo "0" > /proc/sys/net/ipv4/ip_forward
```

If you are using an older kernel (WHY) you will need to re−compile your kernel with forwarding turned off. (Just upgrade.)

Try pinging the outside address of the firewall (24.94.1.123) again. It shouldn't work.

Now turn on IP forwarding and/or masquerading. You should be able to ping the anywhere on the Internet from any system on your LAN.

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

BIG NOTE: If you are using "REAL" IP addresses on your LAN (not 192.168.1.*) and you can't ping the internet but you CAN ping the Internet side of your firewall, make sure your ISP is routing packets for your private network address.

A test for this problem is to have someone else on the Internet (say a friend using a local provider) use traceroute to your network. If the trace stops at your providers router, then they are not forwarding your traffic.

It works? Great. The hard part is done. :−)

6.5 Securing the Firewall

A firewall isn't any good if the system it is build on is left wide open to attacks. A "bad guy" could gain access to the through a non firewall service and modify it for their own needs. You need to turning off any unneeded services.

Look in your /etc/inetd.conf file. This file configures inetd also known as the "super server". It controls a bunch of the server daemons and starts them as they are requested by a packet arriving at a "well known" port.

You should turn off echo, discard, daytime, chargen, ftp, gopher, shell, login, exec, talk, ntalk, pop−2, pop−3, netstat, systat, tftp, bootp, finger, cfinger, time, swat and linuxconfig if you have one.

To turn a service off, put # as the first character of the service line. When your done, send a SIG−HUP to the process by typing **"kill −HUP <pid>"**, where <pid> is the process number of inetd. This will make inetd re−read its configuration file (inetd.conf) and restart without taking your system down.

Test this by telneting to port 15 (netstat) on firewall. If you get any output you have not turned these services off.

telnet localhost 19

You can also create the file /etc/nologin. Put a few line of text in it like (BUZZ OFF). When this file exists, login will not allow user to logon. They will see the contents of this file and their logins refused. Only root can logon.

You can also edit the file /etc/securetty. If the user is root, then the login must be occurring on a tty listed in /etc/securetty. Failures will be logged with the syslog facility. With both of these controls in place the only way to logon to the firewall will be as root from the console.

NEVER EVER TELNET to a system and log IN AS ROOT. If you need remote root access SSH (Secure Shell). You might even turn off telnet.

If you are really paranoid you need to be using lids (Linux Intrusion Detect System). It is an intrusion detection system patch for the Linux kernel; it can protect important files from being changed. When it's in effect, no one (including root) can change the protected files or directories and their sub−directories. You have to reboot the system with a security=1 LILO setting to modify secure files. (I'd also boot into single user mode.)

7.IP filtering setup (IPFWADM)

If you are using kernel 2.1.102 or newer skip to the next section on IPCHAINS.

In older kernels IP Forwarding is turned on by default in the kernel. Because of this, your network should start by denying access to everything and flushing any ipfw rules in place from the last time it was run. This script fragment should go in your network startup script. (/etc/rc.d/init.d/network)

```
  #
  # setup IP packet Accounting and Forwarding
  #
  #   Forwarding
  #
  # By default DENY all services
  ipfwadm -F -p deny
  # Flush all commands
  ipfwadm -F -f
  ipfwadm -I -f
  ipfwadm -O -f
```

Now we have the ultimate firewall. Nothing can get through.

Now create the file /etc/rc.d/rc.firewall. This script should allow email, Web and DNS traffic through. ;–)

```
#! /bin/sh
#
# rc.firewall
#
# Source function library.
. /etc/rc.d/init.d/functions

# Get config.
. /etc/sysconfig/network

# Check that networking is up.
if [ ${NETWORKING} = "no" ]
then
        exit 0
fi
case "$1" in
  start)
  echo -n "Starting Firewall Services: "
  # Allow email to got to the server
  /sbin/ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -D 192.1.2.10 25
  # Allow email connections to outside email servers
  /sbin/ipfwadm -F -a accept -b -P tcp -S 192.1.2.10 25 -D 0.0.0.0/0 1024:65535
  # Allow Web connections to your Web Server
  /sbin/ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -D 192.1.2.11 80
  # Allow Web connections to outside Web Server
  /sbin/ipfwadm -F -a accept -b -P tcp -S 192.1.2.* 80 -D 0.0.0.0/0 1024:65535
  # Allow DNS traffic
  /sbin/ipfwadm -F -a accept -b -P udp -S 0.0.0.0/0 53 -D 192.1.2.0/24
  ;;
  stop)
  echo -n "Stooping Firewall Services: "
  ipfwadm -F -p deny
  ;;
  status)
  echo -n "Now do you show firewall stats?"
  ;;
  restart|reload)
        $0 stop
        $0 start
        ;;
  *)
        echo "Usage: firewall {start|stop|status|restart|reload}"
        exit 1
esac
```

NOTE: In this example we have the email (smtp) server running at 192.1.2.10 that must be able to send and receive on port 25. The web server running at 192.1.2.11. We are allowing anyone on the LAN to get to outside web and DNS servers.

This is not perfectly secure. Because port 80 doesn't have to used as a web port, a smart hacker might use this port to create a virtual private network (VPN) through the firewall. The way around this is to setup a web proxy. and only allow the proxy through the firewall. Users on the LAN will have to go through the proxy to get to outside web servers.

You might also be interested in accounting for traffic going through your firewall. This script will count ever packet. You could add a line or two to account for packets going to just a single system.

```
# Flush the current accounting rules
ipfwadm -A -f
# Accounting
/sbin/ipfwadm -A -f
/sbin/ipfwadm -A out -i -S 192.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A out -i -S 0.0.0.0/0 -D 192.1.2.0/24
/sbin/ipfwadm -A in -i -S 192.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A in -i -S 0.0.0.0/0 -D 192.1.2.0/24
```

If all you need is a filtering firewall you can stop here. Test it and Enjoy.

8.IP filtering setup (IPCHAINS)

Linux ipchains is a rewrite of the Linux IPv4 firewalling code and a rewrite of ipfwadm, which was a rewrite of BSD's ipfw, I believe. It is required to administer the IP packet filters in Linux kernel versions 2.1.102 and above.

The older code doesn't deal with fragments, has 32−bit counters (on Intel at least), doesn't allow specification of protocols other than TCP, UDP or ICMP, can't make large changes atomically, can't specify inverse rules, has some quirks, and can be tough to manage (making it prone to user error). Or so the author says.

I'm not going to get real deep into how to control an IPChains firewall because there is a GREAT!! HOWTO on it at http://www.rustcorp.com/linux/ipchains/HOWTO.html. I'd just end up duplicating it here. Here are the basics.

You work with chains by name. You start with three built−in chains input, output and forward which you can't delete. You can create chains of your own. Rules can then be added and deleted from these rule sets.

The operations to work on entire chains are;

1. Create a new chain (−N).

2. Delete an empty chain (−X).
3. Change the policy for a built−in chain. (−P).
4. List the rules in a chain (−L).
5. Flush the rules out of a chain (−F).
6. Zero the packet and byte counters on all rules in a chain (−Z).

There are several ways to manipulate rules inside a chain:

1. Append a new rule to a chain (−A).
2. Insert a new rule at some position in a chain (−I).
3. Replace a rule at some position in a chain (−R).
4. Delete a rule at some position in a chain (−D).
5. Delete the first rule that matches in a chain (−D).

There are a few operations for masquerading, which are in ipchains for want of a good place to put them:

1. List the currently masqueraded connections (−M −L).
2. Set masquerading timeout values (−M −S).

There are some timing issues involved in altering firewall rules. If you are not careful, you can let packets through while you are half−way through your changes. A simplistic approach is to do the following:

```
# ipchains −I input 1 −j DENY
# ipchains −I output 1 −j DENY
# ipchains −I forward 1 −j DENY
```

... make changes ...

```
# ipchains −D input 1
# ipchains −D output 1
# ipchains −D forward 1
#
```

This drops all packets for the duration of the changes.

Here a duplicate of the above firewall rules in IPChains.

```
#!/bin/sh
#
# rc.firewall
#
## Flush everything, start from scratch
  /sbin/ipchains −F input
  /sbin/ipchains −F output
  /sbin/ipchains −F forward

## Redirect for HTTP Transparent Proxy
  #$IPCHAINS  −A input −p tcp −s 192.1.2.0/24 −d 0.0.0.0/0 80 −j REDIRECT 8080

## Create your own chain
```

```
  /sbin/ipchains -N my-chain
  # Allow email to got to the server
  /sbin/ipchains -A my-chain -s 0.0.0.0/0 smtp -d 192.1.2.10 1024:-j ACCEPT
  # Allow email connections to outside email servers
  /sbin/ipchains -A my-chain -s 192.1.2.10 -d 0.0.0.0/0 smtp -j ACCEPT
  # Allow Web connections to your Web Server
  /sbin/ipchains -A my-chain -s 0.0.0.0/0 www -d 192.1.2.11 1024: -j ACCEPT
  # Allow Web connections to outside Web Server
  /sbin/ipchains -A my-chain -s 192.1.2.0/24 1024: -d 0.0.0.0/0 www -j ACCEPT
  # Allow DNS traffic
  /sbin/ipchains -A my-chain -p UDP -s 0.0.0.0/0 dns -d 192.1.2.0/24 -j ACCEPT

## If you are using masquerading
  # don't masq internal-internal traffic
  /sbin/ipchains -A forward -s 192.1.2.0/24 -d 192.1.2.0/24 -j ACCEPT
  # don't masq external interface direct
  /sbin/ipchains -A forward -s 24.94.1.0/24 -d 0.0.0.0/0 -j ACCEPT
  # masquerade all internal IP's going outside
  /sbin/ipchains -A forward -s 192.1.2.0/24 -d 0.0.0.0/0 -j MASQ

## Deny everything else
  /sbin/ipchains -P my-chain input DENY
```

Don't stop here. This is not a great firewall and I'm sure you have other services you will be providing. Again, read the IPCHAINS–HOWTO.

9.Installing a Transparent SQUID proxy

The squid proxy is available at **http://squid.nlanr.net/**.

The SQUID developers provide RedHat and Debian packages. If you can, use one of these.

10.Installing the TIS Proxy server

10.1 Getting the software

The TIS FWTK is available at **http://www.tis.com/research/software/**.

Don't make the mistake I did. When you ftp files from TIS, READ THE README's. The TIS fwtk is locked up in a hidden directory on their server.

TIS requires you read their agreement at http://www.tis.com/research/software/fwtk_readme.html and then

send email to fwtk−request@tislabs.com with only the word **accepted** in the body of the message to learn the name of this hidden directory. No subject is needed in the message. Their system will then mails you back the directory name (good for 12 hours) to download the source.

As of this writing, the current version of FWTK is 2.1.

10.2 Compiling the TIS FWTK

Version 2.1 of the FWTK compiles much easier then any of the older versions.

EXPLAIN HERE!!!

Now run **make**.

10.3 Installing the TIS FWTK

Run **make install**.

The default installation directory is /usr/local/etc. You could change this (I didn't) to a more secure directory. I chose to change the access to this directory to 'chmod 700'.

All last is left now is to configure the firewall.

10.4 Configuring the TIS FWTK

Now the fun really begins. We must teach the system to call theses new services and create the tables to control them.

I'm not going to try to re−write the TIS FWTK manual here. I will show you the setting I found worked and explain the problems I ran into and how I got around them.

There are three files that make up these controls.

- /etc/services
    - Tells the system what ports a services is on.
- /etc/inetd.conf
    - Tells inetd what program to call when someone knocks on a service port.
- /usr/local/etc/netperm−table

- Tells the FWTK services who to allow and deny service to.

To get the FWTK functioning, you should edit these files from the bottom up. Editing the services file without the inetd.conf or netperm−table file set correctly could make your system inaccessible.

The netperm−table file

This file controls who can access the services of the TIS FWTK. You should think about the traffic using the firewall from both sides. People outside your network should identify themselves before gaining access, but the people inside your network might be allowed to just pass through.

So people can identify themselves, the firewall uses a program called **authsrv** to keep a database of user IDs and passwords. The authentication section of the netperm−table controls where the database is keep and who can access it.

I had some trouble closing the access to this service. Note the premit−hosts line I show uses a '*' to give everyone access. The correct setting for this line is " `authsrv: premit-hosts localhost` if you can get it working.

```
#
# Proxy configuration table
#
# Authentication server and client rules
authsrv:      database /usr/local/etc/fw-authdb
authsrv:      permit-hosts *
authsrv:      badsleep 1200
authsrv:      nobogus true
# Client Applications using the Authentication server
*:            authserver 127.0.0.1 114
```

To initialize the database, su to root, and run **./authsrv** in the /var/local/etc directory to create the administrative user record. Here is a sample session.

Read the FWTK documentation to learn how to add users and groups.

```
#
# authsrv
authsrv# list
authsrv# adduser admin "Auth DB admin"
ok - user added initially disabled
authsrv# ena admin
enabled
authsrv# proto admin pass
changed
authsrv# pass admin "plugh"
Password changed.
authsrv# superwiz admin
set wizard
authsrv# list
Report for users in database
user   group  longname           ok?    proto   last
```

```
------ ------ ----------------- -----  ------  -----
admin         Auth DB admin      ena    passw   never
authsrv# display admin
Report for user admin (Auth DB admin)
Authentication protocol: password
Flags: WIZARD
authsrv# ^D
EOT
#
```

The telnet gateway (tn−gw) controls are straight forward and the first you should set up.

In my example, I permit host from inside the private network to pass through without authenticating themselves. (permit−hosts 19961.2.* −passok) But, any other user must enter their user ID and password to use the proxy. (permit−hosts * −auth)

I also allow one other system (192.1.2.202) to access the firewall directly without going through the firewall at all. The two inetacl−in.telnetd lines do this. I will explain how these lines are called latter.

The Telnet timeout should be keep short.

```
# telnet gateway rules:
tn-gw:              denial-msg      /usr/local/etc/tn-deny.txt
tn-gw:              welcome-msg     /usr/local/etc/tn-welcome.txt
tn-gw:              help-msg        /usr/local/etc/tn-help.txt
tn-gw:              timeout 90
tn-gw:              permit-hosts 192.1.2.* -passok -xok
tn-gw:              permit-hosts * -auth
# Only the Administrator can telnet directly to the Firewall via Port 24
netacl-in.telnetd: permit-hosts 192.1.2.202 -exec /usr/sbin/in.telnetd
```

The r−commands work the same way as telnet.

```
# rlogin gateway rules:
rlogin-gw:    denial-msg      /usr/local/etc/rlogin-deny.txt
rlogin-gw:    welcome-msg     /usr/local/etc/rlogin-welcome.txt
rlogin-gw:    help-msg        /usr/local/etc/rlogin-help.txt
rlogin-gw:    timeout 90
rlogin-gw:    permit-hosts 192.1.2.* -passok -xok
rlogin-gw:    permit-hosts * -auth -xok
# Only the Administrator can telnet directly to the Firewall via Port
netacl-rlogind: permit-hosts 192.1.2.202 -exec /usr/libexec/rlogind -a
```

You shouldn't have anyone accessing your firewall directly and that includes FTP so don't put an FTP, server on you firewall.

Again, the permit−hosts line allows anyone in the protected network free access to the Internet and all others must authenticate themselves. I included logging of every file sent and received to my controls. (−log { retr stor })

The ftp timeout controls how long it will take to drop a bad connections as well as how long a connection will stay open with out activity.

```
# ftp gateway rules:
ftp-gw:              denial-msg      /usr/local/etc/ftp-deny.txt
ftp-gw:              welcome-msg     /usr/local/etc/ftp-welcome.txt
ftp-gw:              help-msg        /usr/local/etc/ftp-help.txt
ftp-gw:              timeout 300
ftp-gw:              permit-hosts 192.1.2.* -log { retr stor }
ftp-gw:              permit-hosts * -authall -log { retr stor }
```

Web, gopher and browser based ftp are contorted by the http−gw. The first two lines create a directory to store ftp and web documents as they are passing through the firewall. I make these files owned by root and put the in a directory accessible only by root.

The Web connection should be kept short. It controls how long the user will wait on a bad connections.

```
# www and gopher gateway rules:
http-gw:      userid          root
http-gw:      directory       /jail
http-gw:      timeout 90
http-gw:      default-httpd   www.afs.net
http-gw:      hosts           192.1.2.* -log { read write ftp }
http-gw:      deny-hosts      *
```

The ssl−gw is really just a pass anything gateway. Be carefully with it. In this example I allow anyone inside the protected network to connect to any server outside the network except the addresses 127.0.0.* and 192.1.1.* and then only on ports 443 through 563. Ports 443 through 563 are known SSL ports.

```
# ssl gateway rules:
ssl-gw:         timeout 300
ssl-gw:         hosts           192.1.2.* -dest { !127.0.0.* !192.1.1.* *:443:563 }
ssl-gw:         deny-hosts      *
```

Here is an example of how to use the plug−gw to allow connections to a news server. In this example I allow anyone inside the protected network to connect to only one system and only to it's news port.

The seconded line allows the news server to pass its data back to the protected network.

Because most clients expect to stay connected while the user read news, the timeout for a news server should be long.

```
# NetNews Pluged gateway
plug-gw:        timeout 3600
plug-gw: port nntp 192.1.2.* -plug-to 24.94.1.22 -port nntp
plug-gw: port nntp 24.94.1.22 -plug-to 192.1.2.* -port nntp
```

The finger gateway is simple. Anyone inside the protected network must login first and then we allow them to use the finger program on the firewall. Anyone else just gets a message.

```
# Enable finger service
netacl-fingerd: permit-hosts 192.1.2.* -exec /usr/libexec/fingerd
netacl-fingerd: permit-hosts * -exec /bin/cat /usr/local/etc/finger.txt
```

I haven't setup the Mail and X−windows services so I'm not including examples. If anyone has a working example, please send me email.

The /etc/services file

This is where it all begins. When a client connects to the firewall it connects on a known port (less then 1024). For example telnet connects on port 23. The inetd deamon hears this connection and looks up the name of these service in the /etc/services file. It then calls the program assigned to the name in the /etc/inetd.conf file.

Some of the services we are creating are not normally in the /etc/services file. You can assign some of them to any port you want. For example, I have assigned the administrator's telnet port (telnet−a) to port 24. You could assign it to port 2323 if you wished. For the administrator (YOU) to connect directly to the firewall you will need to telnet to port 24 not 23 and if you setup your netperm−table file, like I did, you will only be able to this from one system inside your protected network.

```
telnet-a        24/tcp
ftp-gw          21/tcp           # this named changed
auth            113/tcp   ident    # User Verification
ssl-gw          443/tcp
```

11.The SOCKS Proxy Server

11.1 Setting up the Proxy Server

The SOCKS proxy server available from **http://www.socks.nec.com/**.

Uncompressed and untar the files into a directory on your system, and follow the instructions on how to make it. I had a couple problems when I made it. Make sure that your Makefiles are correct.

One important thing to note is that the proxy server needs to be added to /etc/inetd.conf. You must add a line:

```
socks  stream  tcp  nowait  nobody  /usr/local/etc/sockd  sockd
```

to tell the server to run when requested.

11.2 Configuring the Proxy Server

The SOCKS program needs two separate configuration files. One to tell the access allowed, and one to route the requests to the appropriate proxy server. The access file should be housed on the server. The routing file should be housed on every UNIX machine. The DOS and, presumably, Macintosh computers will do their own routing.

The Access File

With socks4.2 Beta, the access file is called "sockd.conf".It should contain 2 lines, a permit and a deny line. Each line will have three entries:

- The Identifier (permit/deny)
- The IP address
- The address modifier

The identifier is either permit or deny. You should have both a permit and a deny line.

The IP address holds a four byte address in typical IP dot notation. I.E. 192.168.1.0.

The address modifier is also a typical IP address four byte number. It works like a netmask. Envision this number to be 32 bits (1s or 0s). If the bit is a 1, the corresponding bit of the address that it is checking must match the corresponding bit in the IP address field. For instance, if the line is:

```
permit 192.168.1.23 255.255.255.255
```

it will permit only the IP address that matches every bit in 192.168.1.23, eg, only 192.168.1.3. The line:

```
permit 192.168.1.0 255.255.255.0
```

will permit every number within group 192.168.1.0 through 192.168.1.255, the whole C Class domain. One should not have the line:

```
permit 192.168.1.0 0.0.0.0
```

as this will permit every address, regardless.

So, first permit every address you want to permit, and then deny the rest. To allow everyone in the domain 192.168.1.xxx, the lines:

```
permit 192.168.1.0 255.255.255.0
deny 0.0.0.0 0.0.0.0
```

will work nicely. Notice the first "0.0.0.0" in the deny line. With a modifier of 0.0.0.0, the IP address field does not matter. All 0's is the norm because it is easy to type.

More than one entry of each is allowed.

Specific users can also be granted or denied access. This is done via ident authentication. Not all systems support ident, including Trumpet Winsock, so I will not go into it here. The documentation with socks is quite adequate on this subject.

The Routing File

The routing file in SOCKS is poorly named "socks.conf". I say "poorly named" because it is so close to the name of the access file that it is easy to get the two confused.

The routing file is there to tell the SOCKS clients when to use socks and when not to. For instance, in our network, 192.168.1.3 will not need to use socks to talk with 192.168.1.1, firewall. It has a direct connection in via Ethernet. It defines 127.0.0.1, the loopback, automatically. Of course you do not need SOCKS to talk to yourself. There are three entries:

- deny
- direct
- sockd

Deny tells SOCKS when to reject a request. This entry has the same three fields as in sockd.conf, identifier, address and modifier. Generally, since this is also handled by sockd.conf, the access file, the modifier field is set to 0.0.0.0. If you want to preclude yourself from calling any place, you can do it here.

The direct entry tells which addresses to not use socks for. These are all the addresses that can be reached without the proxy server. Again we have the three fields, identifier, address and modifier. Our example would have

```
direct 192.168.1.0 255.255.255.0
```

Thus going direct for any on our protected network.

The sockd entry tells the computer which host has the socks server daemon on it. The syntax is:

```
sockd @=<serverlist> <IP address> <modifier>
```

Notice the @= entry. This allows you to set the IP addresses of a list of proxy servers. In our example, we only use one proxy server. But, you can have many to allow a greater load and for redundancy in case of failure.

The IP address and modifier fields work just like in the other examples. You specify which addresses go where through these. 6.2.3. DNS from behind a Firewall

Setting up Domain Name service from behind a firewall is a relatively simple task. You need merely to set up the DNS on the firewalling machine. Then, set each machine behind the firewall to use this DNS.

11.3 Working With a Proxy Server

Unix

To have your applications work with the proxy server, they need to be "sockified". You will need two different telnets, one for direct communication, one for communication via the proxy server. SOCKS comes with instructions on how to SOCKify a program, as well as a couple pre−SOCKified programs. If you use the SOCKified version to go somewhere direct, SOCKS will automatically switch over to the direct version for you. Because of this, we want to rename all the programs on our protected network and replace them with the SOCKified programs. "Finger" becomes "finger.orig", "telnet" becomes "telnet.orig", etc. You must tell SOCKS about each of these via the include/socks.h file.

Certain programs will handle routing and sockifying itself. Netscape is one of these. You can use a proxy server under Netscape by entering the server's address (192.168.1.1 in our case) in the SOCKs field under Proxies. Each application will need at least a little messing with, regardless of how it handles a proxy server.

MS Windows with Trumpet Winsock

Trumpet Winsock comes with built in proxy server capabilities. In the "setup" menu, enter the IP address of the server, and the addresses of all the computers reachable directly. Trumpet will then handle all outgoing packets.

Getting the Proxy Server to work with UDP Packets

The SOCKS package works only with TCP packets, not UDP. This makes it quite a bit less useful. Many useful programs, such as talk and Archie, use UDP. There is a package designed to be used as a proxy server for UDP packets called UDPrelay, by Tom Fitzgerald <fitz@wang.com>. Unfortunately, at the time of this writing, it is not compatible with Linux.

11.4 Drawbacks with Proxy Servers

The proxy server is, above all, a `security device`. Using it to increase internet access with limited IP addresses will have many drawbacks. A proxy server will allow greater access from inside the protected network to the outside, but will keep the inside completely inaccessible from the outside. This means no servers, talk or archive connections, or direct mailing to the inside computers. These drawbacks might seem slight, but think of it this way:

- You have left a report you are doing on your computer inside a firewall protected network. You are at home, and decide that you would like to go over it. You can not. You can not reach your computer because it is behind the firewall. You try to log into `firewall` first, but since everyone has proxy server access, no one has set up an account for you on it.

- Your daughter goes to college. You want to email her. You have some private things to talk about, and would rather have your mail sent directly to your machine. You trust your systems administrator completely, but still, this is private mail.
- The inability to use UDP packets represents a big drawback with the proxy servers. I imagine UDP capabilities will be coming shortly.

FTP causes another problem with a proxy server. When getting or doing an `ls`, the FTP server opens a socket on the client machine and sends the information through it. A proxy server will not allow this, so FTP doesn't particularly work.

And, proxy servers run slow. Because of the greater overhead, almost any other means of getting this access will be faster.

Basically, if you have the IP addresses, and you are not worried about security, do not use a firewall and/or proxy servers. If you do not have the IP addresses, but you are also not worried about security, you might also want to look into using an IP emulator, like Term, Slirp or TIA. Term is available from **`ftp://sunsite.unc.edu`**, Slirp is available from **`ftp://blitzen.canberra.edu.au/pub/slirp`**, and TIA is available from marketplace.com. These packages will run faster, allow better connections, and provide a greater level of access to the inside network from the internet. Proxy servers are good for those networks which have a lot of hosts that will want to connect to the internet on the fly, with one setup and little work after that.

12.Advanced Configurations

There is one configuration I would like to go over before wrapping this document up. The one I have just outlined will probably suffice for most people. However, I think the next outline will show a more advanced configuration that can clear up some questions. If you have questions beyond what I have just covered, or are just interested in the versatility of proxy servers and firewalls, read on.

12.1 A large network with emphasis on security

Say, for instance, you are the leader of millisha and you wish to network your site. You have 50 computers and a subnet of 32 (5 bits) IP numbers. You need various levels of access within your network because you tell your followers different things. Therefore, you'll need to protect certain parts of the network from the rest.

The levels are:

1. The external level. This is the level that gets shown to everybody. This is where you rant and rave to get new volunteers.
2. **Troop** This is the level of people who have gotten beyond the external level. Here is where you teach them about the evail government and how to make bombs.
3. **Mercenary** Here is where the *real* plans are keep. In this level is stored all the information on how the 3rd world government is going to take over the world, your plans involving Newt Gingrich,

Oklahoma City, lown care products and what really is stored in that hangers at area 51.

The Network Setup

The IP numbers are arranged as:

- 1 number is 192.168.1.255, which is the broadcast address and is not usable.
- 23 of the 32 IP addresses are allocated to 23 machines that will be accessible to the internet.
- 1 extra IP goes to a Linux box on that network
- 1 extra goes to a different Linux box on that network.
- 2 IP #'s go to the router
- 4 are left over, but given domain names paul, ringo, john, and george, just to confuse things a bit.
- The protected networks both have the addresses 192.168.1.xxx

Then, two separate networks are built, each in different rooms. They are routed via infrared Ethernet so that they are completely invisible to the outside room. Luckily, infrared ethernet works just like normal ethernet.

These networks are each connected to one of the Linux boxes with an extra IP address.

There is a file server connecting the two protected networks. This is because the plans for taking over the world involves some of the higher Troops. The file server holds the address 192.168.1.17 for the Troop network and 192.168.1.23 for the Mercenary network. It has to have different IP addresses because it has to have different Ethernet cards. IP Forwarding on it is turned off.

IP Forwarding on both Linux boxes is also turned off. The router will not forward packets destined for 192.168.1.xxx unless explicitly told to do so, so the internet will not be able to get in. The reason for turning off IP Forwarding here is so that packets from the Troop's network will not be able to reach the Mercenary network, and vica versa.

The NFS server can also be set to offer different files to the different networks. This can come in handy, and a little trickery with symbolic links can make it so that the common files can be shared with all. Using this setup and another ethernet card can offer this one file server for all three networks.

The Proxy Setup

Now, since all three levels want to be able to monitor the network for their own devious purposes, all three need to have net access. The external network is connected directly into the internet, so we don't have to mess with proxy servers here. The Mercenary and Troop networks are behind firewalls, so it is necessary to set up proxy servers here.

Both networks will be setup very similarly. They both have the same IP addresses assigned to them. I will throw in a couple of parameters, just to make things more interesting though.

1. No one can use the file server for internet access. This exposes the file server to viruses and other nasty things, and it is rather important, so its off limits.

2. We will not allow troop access to the World Wide Web. They are in training, and this kind of information retrieval power might prove to be damaging.

So, the sockd.conf file on the Troop's Linux box will have this line:

```
deny 192.168.1.17 255.255.255.255
```

and on the Mercenary machine:

```
deny 192.168.1.23 255.255.255.255
```

And, the Troop's Linux box will have this line

```
deny 0.0.0.0 0.0.0.0 eq 80
```

This says to deny access to all machines trying to access the port equal (eq) to 80, the http port. This will still allow all other services, just deny Web access.

Then, both files will have:

```
permit 192.168.1.0 255.255.255.0
```

to allow all the computers on the 192.168.1.xxx network to use this proxy server except for those that have already been denied (ie. The file server and Web access from the Troop network).

The Troop's sockd.conf file will look like:

```
deny 192.168.1.17 255.255.255.255
deny 0.0.0.0 0.0.0.0 eq 80
permit 192.168.1.0 255.255.255.0
```

and the Mercenary file will look like:

```
deny 192.168.1.23 255.255.255.255
permit 192.168.1.0 255.255.255.0
```

This should configure everything correctly. Each network is isolated accordingly, with the proper amount of interaction. Everyone should be happy.

13.Making Management Easy

13.1 Firewall tools

There are several software packages that will make managing your firewall easier.

Be carefull, don't use these tools unless you can do without them. These scripts make it just as easy to make a misstake as they do to help you get it wright.

Both graphical and web based interfaces are being developed to work with the Linux filtering rules. Some companies have even create commercial firewalls based on Linux by putting it in their own box with their own management code. (nice)

I'm not realy a GUI guy. However, I have been using firewalls with GUI interfaces for some time. I've found they help by providing a nice report of all the rules in one easy glance.

gfcc (GTK+ Firewall Control Center) is a GTK+ application which can control Linux firewall policies and rules, based on ipchains package. Go to http://icarus.autostock.co.kr and get your copy. This is a realy good tool.

I have included RC scripts in appendex A. These scripts work with and without gfcc.

There a lots of scripts avaible to setup a firewall. One very complete script is avaible at http://www.jasmine.org.uk/~simon/bookshelf/papers/instant−firewall/instant−firewall.html. Another will done script is at http://www.pointman.org/.

Kfirewall is a GUI frontend for ipchains or ipfwadm (depending on your kernel version). http://megaman.ypsilonia.net/kfirewall/

FCT is an HTML based tool for the configuration of a firewall. It features automatic script−generation for IP−filtering commands (ipfwadm) on a firewall for multiple interfaces and any internet services. http://www.fen.baynet.de/~ft114/FCT/firewall.htm

13.2 General tools

WebMin is a general system admin package. It will not help you manage the firewall rules but it will help you with turning on and off damons and processes. This program is VERY good, I'm hoping the J. Cameron will include a IPCHAINS module. http://www.webmin.com/

If you are an ISP, you will want to know about IPFA (IP Firewall Acounting) http://www.soaring−bird.com/ipfa/. It can do Per−Month/Per−day/per−min/ logs and has a Web based GUI administation.

---

httptunnel. httptunnel creates a bidirectional virtual data path tunnelled in HTTP requests. The HTTP requests can be sent via an HTTP proxy if so desired. Or, on their system they install a Virtual Private Network (vpn). See: http://sunsite.auc.dk/vpnd/ Or, Maybe this user simply puts a modem on their NT system and turns on routing. Finally, on the workstation, on the private LAN, change the default gateway to point to the new route to the Internet. Now, from this workstation, you can go anywhere. The only thing the firewall admin might see is one connect with nowill see is a really long DNS lookup. Now, take over the world!

15.APPENDEX A – Example Scripts

15.1 RC Script useing GFCC

```
#!/bin/bash
#
# Firewall Script - Version 0.9.1
#
# chkconfig: 2345 09 99
# description: firewall script for 2.2.x kernel
# Set for testing
# set -x
#
# NOTES:
#
#  This script is written for RedHat 6.1 or better.
#
#  Be careful about offering public services like web or ftp servers.
#
# INSTALLATION:
#  1. place this file in /etc/rc.d/init.d  (you'll have to be root..)
#     call it something like "firewall"    :-)
#     make it root owned -->  "chown root.root (filename)"
#     make it executable -->  "chmod 755 (filename)"
#
#  2. use GFCC to create your firewall rules and export them to a file
#     named /etc/gfcc/rules/firewall.rule.sh.
#
#  3. add the firewall to the RH init structure --> "chkconfig --add (filename)"
#     next time the router boots, things should happen automagically!
#     sleep better at night knowing you are *LESS* vulnerable than before...
#
# RELEASE NOTES
#   30 Jan, 2000 - Changed to GFCC script
#   11 Dec, 1999 - updated by Mark Grennan <mark@grennan.com>
#   20 July, 1999 - initial writing - Anthony Ball <tony@LinuxSIG.org>
#

################################################

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
```

```
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

# See how we are called
case "$1" in

  start)
        # Start providing access
        action "Starting firewall: " /bin/true
        /etc/gfcc/rules/firewall.rule.sh
        echo
        ;;

  stop)
        action "Stoping firewall: " /bin/true
        echo 0 > /proc/sys/net/ipv4/ip_forward
        /sbin/ipchains -F input
        /sbin/ipchains -F output
        /sbin/ipchains -F forward

        echo
        ;;

  restart)
        action "Restarting firewall: " /bin/true
        $0 stop
        $0 start

        echo
        ;;

  status)
        # List out all settings
        /sbin/ipchains -L
        ;;

  test)
        action "Test Mode firewall: " /bin/true
        /sbin/ipchains -F input
        /sbin/ipchains -F output
        /sbin/ipchains -F forward
        echo 1 > /proc/sys/net/ipv4/ip_forward
        /sbin/ipchains -A input -j ACCEPT
        /sbin/ipchains -A output -j ACCEPT
        /sbin/ipchains -P forward DENY
        /sbin/ipchains -A forward -i $PUBLIC -j MASQ

        echo
        ;;

  *)
        echo "Usage: $0 {start|stop|restart|status|test}"
        exit 1

esac
```

15.2 GFCC script

This script was generated by the Graphical Firewall program (GFCC). This is not the working rule set. This is the exported rules set.

```
#!/bin/sh
# Generated by Gtk+ firewall control center

IPCHAINS=/sbin/ipchains


localnet="192.168.1.0/24"
firewallhost="192.168.1.1/32"
localhost="172.0.0.0/8"
DNS1="24.94.163.119/32"
DNS2="24.94.163.124/32"
Broadcast="255.255.255.255/32"
Multicast="224.0.0.0/8"
Any="0.0.0.0/0"
mail_grennan_com="192.168.1.1/32"
mark_grennan_com="192.168.1.3/32"

$IPCHAINS -P input DENY
$IPCHAINS -P forward ACCEPT
$IPCHAINS -P output ACCEPT

$IPCHAINS -F
$IPCHAINS -X

# input rules
$IPCHAINS -A input -s $Any -d $Broadcast -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any netbios-ns -j DENY
$IPCHAINS -A input -p tcp -s $Any -d $Any netbios-ns -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any netbios-dgm -j DENY
$IPCHAINS -A input -p tcp -s $Any -d $Any netbios-dgm -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any bootps -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any bootpc -j DENY
$IPCHAINS -A input -s $Multicast -d $Any -j DENY
$IPCHAINS -A input -s $localhost -d $Any -i lo -j ACCEPT
$IPCHAINS -A input -s $localnet -d $Any -i eth1 -j ACCEPT
$IPCHAINS -A input -s $localnet -d $Broadcast -i eth1 -j ACCEPT
$IPCHAINS -A input -p icmp -s $Any -d $Any -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any -j ACCEPT ! -y
$IPCHAINS -A input -p udp -s $DNS1 domain -d $Any 1023:65535 -j ACCEPT
$IPCHAINS -A input -p udp -s $DNS2 domain -d $Any 1023:65535 -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any ssh -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any telnet -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any smtp -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any pop-3 -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any auth -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any www -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any ftp -j ACCEPT
$IPCHAINS -A input -s $Any -d $Any -j DENY -l

# forward rules
$IPCHAINS -A forward -s $localnet -d $Any -j MASQ

# output rules
```

15.3 RC Script without GFCC This is the firewall rules set built my hand. It does not use GFCC.

```
#!/bin/bash
#
# Firewall Script - Version 0.9.0

# chkconfig: 2345 09 99
# description: firewall script for 2.2.x kernel

# Set for testing
# set -x

#
# NOTES:
#
#  This script is written for RedHat 6.0 or better.
#
#  This firewall script should work for most routers, dial-up or cable modem.
#  It was written for RedHat distributions.
#
#  Be careful about offering public services like web or ftp servers.
#
# INSTALLATION:
#  1. This file planned for a RedHat system.  It would work
#     on other distro's with perhaps no modification, but again...
#     Who knows?!!?  These instructions apply to RedHat systems.
#
#  2. place this file in /etc/rc.d/init.d  (you'll have to be root..)
#     call it something like "firewall"    :-)
#     make it root owned -->  "chown root.root <filename>"
#     make it executable -->  "chmod 755 <filename>"
#
#  3. set the values for your network, internal interface, and DNS servers
#     uncomment lines further down to enable optional in-bound services
#     make sure "eth0" is your internal NIC (or change the value below)
#     test it -->  "/etc/rc.d/init.d/<filename> start"
#     you can list the rules -->  "ipchains -L -n"
#     fix anything that broke...  :-)
#
#  4. add the firewall to the RH init structure --> "chkconfig --add <filename>"
#     next time the router boots, things should happen automagically!
#     sleep better at night knowing you are *LESS* vulnerable than before...
#
# RELEASE NOTES
#   20 July, 1999 - initial writing - Anthony Ball <tony@LinuxSIG.org>
#   11 Dec, 1999 - updated by Mark Grennan <mark@grennan.com>
#

################################################
#  Fill in the values below to match your
#  local network.

PRIVATENET=xxx.xxx.xxx.xxx/xx

PUBLIC=ppp0
PRIVATE=eth0

# your dns servers
```

```
DNS1=xxx.xxx.xxx.xxx
DNS2=xxx.xxx.xxx.xxx

################################################

# some handy generic values to use
ANY=0.0.0.0/0
ALLONES=255.255.255.255

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

# See how we are called
case "$1" in

  start)
        # Start providing access
        action "Starting firewall: " /bin/true

        ##
        ## Setup Envirement
        ##
        # Flush all lists
        /sbin/ipchains -F input
        /sbin/ipchains -F output
        /sbin/ipchains -F forward

        # Plug up everything
        /sbin/ipchains -I input 1 -j DENY

        # set policy to deny (Default is ACCEPT)
        /sbin/ipchains -P input DENY
        /sbin/ipchains -P output ACCEPT
        /sbin/ipchains -P forward ACCEPT

        # Turn on packet forwarding
        echo 1 > /proc/sys/net/ipv4/ip_forward

        ##
        ## Install Modules
        ##
        # Insert the active ftp module.  This will allow non-passive ftp to machines
        # on the local network (but not to the router since it is not masq'd)
        if ! ( /sbin/lsmod | /bin/grep masq_ftp > /dev/null ); then
            /sbin/insmod ip_masq_ftp
        fi

        ##
        ## Some Security Stuff
        ##
        # turn on Source Address Verification and get spoof protection
        # on all current and future interfaces.
        if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
            for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
                echo 1 > $f
            done
```

```
else
    echo
    echo "PROBLEMS SETTING UP IP SPOOFING PROTECTION.  BE WORRIED."
    echo
fi

# deny bcasts on remaining interfaces
/sbin/ipchains -A input -d 0.0.0.0 -j DENY
/sbin/ipchains -A input -d 255.255.255.255 -j DENY

# deny these without logging 'cause there tend to be a lot...
/sbin/ipchains -A input -p udp -d $ANY 137 -j DENY   # NetBIOS over IP
/sbin/ipchains -A input -p tcp -d $ANY 137 -j DENY   #   ""
/sbin/ipchains -A input -p udp -d $ANY 138 -j DENY   #   ""
/sbin/ipchains -A input -p tcp -d $ANY 138 -j DENY   #   ""
/sbin/ipchains -A input -p udp -d $ANY 67 -j DENY    # bootp
/sbin/ipchains -A input -p udp -d $ANY 68 -j DENY    #   ""
/sbin/ipchains -A input -s 224.0.0.0/8 -j DENY       # Multicast addresses

##
## Allow private network out
##
# allow all packets on the loopback interface
/sbin/ipchains -A input -i lo -j ACCEPT

# allow all packets from the internal "trusted" interface
/sbin/ipchains -A input -i $PRIVATE -s $PRIVATENET -d $ANY -j ACCEPT
/sbin/ipchains -A input -i $PRIVATE -d $ALLONES -j ACCEPT

##
## Allow Outside Services into the firewall (if you dare)
##
# allow ICMP
/sbin/ipchains -A input -p icmp -j ACCEPT
# allow TCP
/sbin/ipchains -A input -p tcp ! -y -j ACCEPT

# allow lookups to DNS (on firewall)
/sbin/ipchains -A input -p udp -s $DNS1 domain -d $ANY 1023: -j ACCEPT
/sbin/ipchains -A input -p udp -s $DNS2 domain -d $ANY 1023: -j ACCEPT
# or (BETTER IDEA) run a caching DNS server on the router and use the
# following two lines instead...
# /sbin/ipchains -A input -p udp -s $DNS1 domain -d $ANY domain -j ACCEPT
# /sbin/ipchains -A input -p udp -s $DNS2 domain -d $ANY domain -j ACCEPT

# uncomment the following to allow ssh in
/sbin/ipchains -A input -p tcp -d $ANY 22 -j ACCEPT

# uncomment the following to allow telnet in (BAD IDEA!!)
/sbin/ipchains -A input -p tcp -d $ANY telnet -j ACCEPT

# uncomment to allow NTP (network time protocol) to router
# /sbin/ipchains -A input -p udp -d $ANY ntp -j ACCEPT

# uncomment to allow SMTP in (not for mail clients - only a server)
/sbin/ipchains -A input -p tcp -d $ANY smtp -j ACCEPT

# uncomment to allow POP3 in (for mail clients)
/sbin/ipchains -A input -p tcp -d $ANY 110 -j ACCEPT

# allow auth in for sending mail or doing ftp
/sbin/ipchains -A input -p tcp -d $ANY auth -j ACCEPT
```

```
        # uncomment to allow HTTP in (only if you run a web server on the router)
        /sbin/ipchains -A input -p tcp -d $ANY http -j ACCEPT

        # uncomment to allow FTP in
        /sbin/ipchains -A input -p tcp -d $ANY ftp -j ACCEPT

        ##
        ## Masquerading stuff
        ##
        # masquerade packets forwarded from internal network
        /sbin/ipchains -A forward -s $PRIVATENET -d $ANY -j MASQ

        ##
        ## deny EVERYthing else and log them to /var/log/messages
        ##
        /sbin/ipchains -A input -l -j DENY

        # Remove the Plug
        /sbin/ipchains -D input 1

        ;;

  stop)
        action "Stoping firewall: " /bin/true
        echo 0 > /proc/sys/net/ipv4/ip_forward
        /sbin/ipchains -F input
        /sbin/ipchains -F output
        /sbin/ipchains -F forward

        echo
        ;;

  restart)
        action "Restarting firewall: " /bin/true
        $0 stop
        $0 start

        echo
        ;;

  status)
        # List out settings
        /sbin/ipchains -L
        ;;

  test)
        ##
        ## This is about as simple as it gets
        ##    (This is not secure AT ALL)
        action "WARNING Test Firewall: " /bin/true
        /sbin/ipchains -F input
        /sbin/ipchains -F output
        /sbin/ipchains -F forward
        echo 1 > /proc/sys/net/ipv4/ip_forward
        /sbin/ipchains -A input -j ACCEPT
        /sbin/ipchains -A output -j ACCEPT
        /sbin/ipchains -P forward DENY
        /sbin/ipchains -A forward -i $PUBLIC -j MASQ

        echo
        ;;
```

```
  *)
        echo "Usage: $0 {start|stop|restart|status|test}"
        exit 1

esac
```

16.APPENDEX B – An VPN RC Script for RedHat

```
#!/bin/sh
#
# vpnd            This shell script takes care of starting and stopping
#                 vpnd (Vertual Privage Network connections).
#
# chkconfig: - 96 96
# description: vpnd
#

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -f /usr/sbin/vpnd ] || exit 0

[ -f /etc/vpnd.conf ] || exit 0

RETVAL=0

# See how we were called.
case "$1" in
  start)
        # Start daemons.
        echo -n "Starting vpnd: "
        daemon vpnd
        RETVAL=$?
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/vpnd
        echo
        ;;
  stop)
        # Stop daemons.
        echo -n "Shutting down vpnd: "
        killproc vpnd
        RETVAL=$?
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/vpnd
        echo
        ;;
  restart)
        $0 stop
        $0 start
        ;;
```

```
  *)
        echo "Usage: vpnd {start|stop|restart}"
        exit 1
esac

exit $RETVAL
```