# Global Information Assurance Certification Paper

John B. Folkerts

# SANS Firewall & Perimeter Practical

*Implementing Perimeter Security with IPF*

# GIAC Firewall and Perimeter Practical

*Implementing Perimeter Security with IPF*

## Overview

Firewall have become so popular in the last five years that where they were once inaccessible to smaller organizations and businesses because of cost constraints, now they are giving them away on the internet for free. One such product is ip_filter (or IPF), a Unix based, stateful packet filter developed by Darren Reed, and found at http://coombs.anu.edu.au/~avalon/ip-filter.html.

This paper describes a simple implementation of ip_filter to accomplish eleven steps for securing your network from some of today's most hazardous security vulnerabilities. For more information on top security vulnerabilities, visit SANS at http://www.sans.org/topten.htm.

## Perimeter Description

For your IPF-based perimeter defense solution you will need a Unix box of some sort. Our example is based on Solaris 2.6, with all the currently recommended security patches. Be sure to secure your firewall system using Jean Chouanard's YASSP package (http://yassp.parc.xerox.com). Jean also maintains links to a number of useful Solaris security resources. The configuration described is built with IPF version 3.3.5.

The firewall has two interfaces, le0 (external) and le1 (internal), and a basic set of services are offered to internal users through our firewall. Services to the internet include dns (for our domain), mail, and one https server for access to email on the road.

### Policy

For our policy we like to go with the KISS principal ("Keep it simple, stupid"). Inbound connections require a valid reason and the tightest boundaries we can put on them. Outbound connections are allowed on nearly all ports, unless we have a good reason to deny an individual service. In addition, we will ensure that vulnerable services (such as the ones in the exercises) are blocked appropriately and logged for intrusion analysis.

### Network Topology

The network topology looks like this:

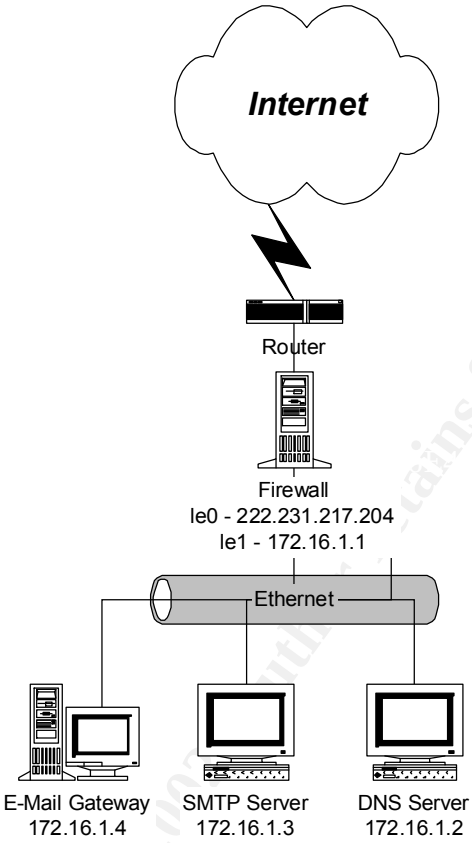


*Figure: Network Topology*

## Working with IPF

### Installation

IPF is easily installed once you've downloaded from the site referenced above. The documentation indicates that you should not use GNU `make` to compile the source. Once compiled, follow the directions in `INSTALL.Sol2` to make the package. This package can then be easily distributed to any system on which you have a need to install it.

In order to turn on firewall functionality, make sure you turn on IP forwarding (it has probably been turned off by YASSP at this point) by executing the command `ndd -set /dev/ip ip_forwarding 1` and ensuring that this setting is changed on your system for startup.

Note that IPF does use the native OS's IP stack to forward packets. The quality of a system's IP stack will influence your decision on which platform to implement IPF.

### Building Rules

IPF rules are kept in `/etc/opt/ipf/ipf.conf` and are similar in construction to other router and firewall rules—however (one big caveat) IPF does not implement a first match philosophy where the first rule that matches the packet is acted upon. Rather, the last rule that matches a packet is acted upon. This has its pros and cons, but the best thing is that if you are used to first match systems, you can essentially make first match rules using the key word `quick`. Another advantage is the ability to group rules and form a decision tree of a sort to enhance performance.

Rules have the following general format:

```
action in/out [options] [on interface] [protocol(s)] [from source to dest]
         [ports] [protocol options] [head] [group]
```

Actions are defined as "block", "pass", and "log". "in" or "out" refer to the direction the packet is passing through the referenced interface. Rule options can be set, such as "log" a packet that matches this rule, and "quick" processing of the packet. Protocol choices are typically "tcp", "udp", "tcp/udp" (which is shorthand to match both protocols), and "icmp". Each protocol has ports and/or options associated with it. Head and group are used to group rules, which will be explained when it comes time to put all our rules together.

In addition, beware that IPF is distributed with a default behavior to pass packets. When no rule applies, it passes the packet. Thus it is important to make sure that you cover all the options in your ruleset.

In order to make your ruleset active, use the command: `/sbin/ipf -Fa` to flush the current ruleset. Then `/sbin/ipf -f /etc/opt/ipf/ipf.conf` will load your rules.

### IPNAT

IPF also supports Network Address Translation (NAT) with the `/sbin/ipnat` program. This evaluates rules that you set up in `/etc/opt/ipf/ipnat.rules`. For our example, we are using three NATed addresses which map back to ports on the external firewall interface:

```
# /etc/opt/ipf/ipnat.rules
#
map le0 172.16.1.0/24 -> 222.231.217.204/32
rdr le0 222.231.217.204 port 53 -> 172.16.1.2 port 53
rdr le0 222.231.217.204 port 53 -> 172.16.1.2 port 53 udp
rdr le0 222.231.217.204 port 25 -> 172.16.1.3 port 25
rdr le0 222.231.217.204 port 443 -> 172.16.1.4 port 443
```

The first line ensures that the firewall rewrites all outgoing addresses as the external interface IP address of the firewall. The next four lines are port redirects, sending dns, smtp, and https services to specific IP addresses inside the firewall. Use `/sbin/ipnat -CF -f /etc/opt/ipf/ipnat.rules` to load your nat ruleset.

### Other Resources

The documentation provided in the IPF distribution is fairly useful, but you will probably want to reference the *IP Filter Based Firewalls HOWTO* by Brendon Conoboy and Erik Fichtner, available at http://www.obfuscation.org/ipf/.

## Implementing GIAC Recommended Security Controls

### 1. Block Spoofed Addresses and Source-routed Packets

#### Rationale

Spoofed addressing is the technique of placing packets on the network with a source address other than the real source address of your system. This is very easy to do (e.g. with tools such as netcat) and very useful to an attacker since your firewall rules rely a great deal on source address for decision-making.

Source-routing is a feature included with IP which enables you to place a return route on a packet. This means an attacker can make sure that the responses to spoofed packets get returned back to them, no matter what the routing tables say.

### Applying the Rules

Source routing, while legal in the IP specification, is simply considered "bad form" on the network. We will disallow source routing (designated as IP options `lsrr` and `ssrr`) and packets which are too short to include a complete IP header with the following syntax which applies to all interfaces.

```
block in log quick all with short
block in log quick all with opt lsrr
block in log quick all with opt ssrr
```

It's impossible to absolutely know a spoofed packet when you see one, but some hints include a source address of localhost (127.*.*.*), or internal addresses, or private addresses coming in from an external interface.

The following lines prevent inbound traffic on our external interface from private or loopback addresses. Since we have private addresses inside (172.16.1.*), it also happens to protect us from attacks that pretend to be coming from internal addresses. Also note that these rules are applied to addresses on the way out of our network as well, to prevent routing accidents.

```
block in log  quick on le0 from 10.0.0.0/8 to any
block in log  quick on le0 from 192.168.0.0/16 to any
block in log  quick on le0 from 172.16.0.0/12 to any
block in log  quick on le0 from 127.0.0.0/8 to any

block out log  quick on le0 from 10.0.0.0/8 to any
block out log  quick on le0 from 192.168.0.0/16 to any
block out log  quick on le0 from 172.16.0.0/12 to any
block out log  quick on le0 from 127.0.0.0/8 to any
```

Notice that the log and quick options are enabled on these rules, and that the same rules apply to both inbound and outbound packets on the le0 (external) interface. Source and destination addresses are described in common network/mask notation, with the "any" keyword specifying any address (or 0.0.0.0/0). Also, note that the 172.16.0.0/12 entry any spoofed packets that pretend to be coming from our internal net.

The filter must be applied at the front of your ipf.conf file. The rules are written "quick" to ensure that we don't inadvertently let packets "fall through" our rule set and also prevent processing time from being wasted on garbage such as this. Also, we are taking a paranoid logging stance by logging all such traffic.

## 2. Block Login Services

### Rationale

Login services provide an avenue for attack against your users' passwords, usually the most vulnerable point of attack in an organization. Brute force attacks can be automated and quickly conducted through a login service such as telnet – either one account at a time, or more likely, across a large spectrum of accounts. The chances of finding one account out of a hundred or a thousand that has a poorly chosen password are pretty good, so we want to severely restrict the ability to try this method.

### Filter Syntax and Description

The common login services in an environment are telnet, the "r-commands", NetBIOS, ftp, and yes, SSH. We will block and log all attempts made on these services from the outside. If users want access to internal resources from the Internet, we'll require some better authentication than a password, like a certificate or one-time password.

```
block in log quick on le0 proto tcp from any to any port = 21
```

```
block in log quick on le0 proto tcp from any to any port = 22
block in log quick on le0 proto tcp from any to any port = 23
block in log quick on le0 proto tcp from any to any port = 139
block in log quick on le0 proto tcp from any to any port 511 >< 515
```

Notice that these examples require the "proto tcp" specification in order to define the appropriate port for each. Ports can be specified with =, <, >, <>, >< (see following table)

| Symbol | Meaning | Example |
|---|---|---|
| = | "equal to" | port = 143 Matches port 143 |
| < | "less than" | port < 20 Matches ports 0-19 |
| > | "greater than" | port > 1023 Matches ports 1024 and higher |
| <> | "exclusive range" | port 6000 <> 6010 Matches ports 0-5999 and 6011+ |
| >< | "inclusive range" | port 511 >< 515 Matches ports 512, 513, 514 |

*Table. Operator signs for specifying ports*

## 3. Block RPC and NFS

### Rationale

RPC and NFS have a bad history of compromises associated with vulnerabilities in the services and the protocols themselves. Buffer overflows abound on RPC services, sadmind being among the most popular. NFS suffers from similar programmatic vulnerabilities, combined with poor authentication of network file share service requests (default setup is no authentication).

### Filter Syntax and Description

Since many Linux and Sun systems come with these services enabled by default, the place to stop this threat is at the perimeter of your organization. In addition, there are plenty of safer alternatives for sharing data (e.g. SMTP, HTTP, HTTP-S, FTP) across the firewall, so we'll deny all RPC and NFS traffic inbound on the firewall and log it if it occurs.

```
block in log quick on le0 proto tcp/udp from any to any port = 111
block in log quick on le0 proto tcp/udp from any to any port = 2049
block in log quick on le0 proto tcp/udp from any to any port = 4045
```

## 4. Block NetBIOS Services

### Rationale

NetBIOS services are commonly used for sharing files internal to an organization, but when allowed through a firewall, provide an opportunity for an attacker to take advantage of poor file share security. By default, usernames, shares, domain names, and a lot of other configuration information are available to "null sessions" over tcp ports 137 and 139 on Windows NT. Windows 9x clients have poor file share security and open shares are common in these environments. In addition, authentication using the legacy LanManager protocol is weak and this opens another route for obtaining passwords.

### Filter Syntax and Description

Again, there are many better alternatives for transferring data across a firewall. We'll deny all NetBIOS traffic inbound on the firewall and log it if it occurs.

```
block in log quick on le0 proto tcp/udp from any to any port = 135
block in log quick on le0 proto tcp/udp from any to any port = 445
block in log quick on le0 proto udp from any to any port = 137
block in log quick on le0 proto udp from any to any port = 138
block in log quick on le0 proto tcp from any to any port = 139
```

## 5. Block X Windows

### Rationale

X Windows does not utilize much in the way of security mechanisms by default. Window traffic (and passwords) traverse in the clear, session hijacking is a risk, and it is far too easy for a user to type "xhost +" to allow all hosts to establish connections to their server via X. If users do need this functionality across unprotected networks, it is best provided by tunneling X over SSH.

### Filter Syntax and Description

The easiest way to block and log X is with the following one line that encompasses the entire X Windows range.

```
block in log quick on le0 proto tcp from any to any port 5999 >< 6256
```

## 6. Block Naming Services

### Rationale

Naming services suffer from the same vulnerability problems as other network services. For example, BIND vulnerabilities are so prevalent on the Internet, that hackers have produced automated cracking scripts that travel from host to host using the same vulnerability. However, naming services are also dangerous from the standpoint that they reveal information about your network which could be useful in mounting an attack.

### Filter Syntax and Description

DNS is one of those services which the Internet can not live without. In our particular setup, we have an internal DNS server which must communicate with an external secondary to do zone transfers. It also provides service for our domain name, so udp/53 needs to be enabled to this server. The first rule allows a secondary initiated zone transfer of our dns server. Keep state ensures that this connection is tracked in the state table (and also allows us to ignore the effects of further applications of the rule set once the connection is established). The second pass rule allows dns queries against our domain's primary dns server (which is mapped through IPF Network Address Translation from the external firewall interface to an internal host). Keep state allows our dns server to talk back. The third rule blocks everything else inbound on that port.

```
pass  in log  quick on le0 proto tcp from 222.231.216.1/32 to 222.231.217.204/32
        port = 53 flags S/SA keep state
pass  in log  quick on le0 proto udp from any to 222.231.217.204/32 port = 53 keep
        state
block in log  quick on le0 proto tcp/udp from any to any port = 53
```

## 7. Block Mail Services

### Rationale

E-mail is the "killer app" in more than one way. Sendmail has a long and sordid history of security vulnerabilities. Likewise, many IMAP and POP3 servers have had problems with this sort of thing. Naturally, mail transfer to your domain requires an inbound connection, but we would like to restrict that connection with some firewall rules.

### Filter Syntax and Description

Our SMTP mail relay is 222.231.212.1, so we've allowed connections on tcp/25 to our internal mail host (NAT'ed through the external firewall interface). Since all the rules are

"quick", the first takes precedence. The last three block rules simply disallow all other connections on SMTP, POP, and IMAP protocols.

```
pass in log   quick on le0 proto tcp from 222.231.212.1/32 to 222.231.217.204/32
        port = 25 flags S/SA keep state
block in log  quick on le0 proto tcp from any to any port = 25
block in log  quick on le0 proto tcp from any to any port 108 >< 111
block in log  quick on le0 proto tcp from any to any port = 143
```

### 8. Block Web Services Inbound

#### Rationale

Often organizations have confidential information on their organizational web servers, so it's a good idea to have a few network controls protecting them. Aside from that, there are a multitude of attacks to be launched against a mail server. Among them: CGI exploits, buffer overflows, brute force attacks against user passwords, and vulnerable application extensions to name a few.

#### Filter Syntax and Description

We want to make sure that external users can get to their email via an SSL enabled web server NATed to the external interface. The first pass rule accomplishes that, keeping state so that IPF will keep track of the connection. We've also constructed the next four blocking rules to block connections to all the common web services ports in our network: 80, 443, 8000-8010, 8080, 8888.

```
pass in log   quick on le0 proto tcp from any to 222.231.217.204/32
        port = 443 flags S/SA keep state
block in log  quick on le0 proto tcp from any to any port = 80
block in log  quick on le0 proto tcp from any to any port = 443
block in log  quick on le0 proto tcp from any to any port 7999 >< 8011
block in log  quick on le0 proto tcp from any to any port = 8080
block in log  quick on le0 proto tcp from any to any port = 8888
```

### 9. Block Small Services

#### Rationale

Small services are a good way to set yourself up for a denial of service (DoS) attack. One the of small services (chargen, tcp/19) will generate character responses to your network request. Echo (tcp/7) will echo back the characters received on a connection to it. This means that if you spoof an chargen service connection to an echo service, you can get the two in a never ending conversation which can slow or crash a machine. The bottom line with small services is that no one ever really uses them, and they simply give an attacker some more information or an additional exploit opportunity.

#### Filter Syntax and Description

Following the convention already established, the ports are blocked as shown below.

```
block in log  quick on le0 proto tcp/udp from any to any port < 20
block in log  quick on le0 proto tcp/udp from any to any port = 37
```

### 10. Block Other Miscellaneous Ports

#### Rationale

Aside from the programmatic vulnerabilities that have been discovered on any of these miscellaneous ports, here is a rundown of reasons to block these services:

| Port | Vulnerable to |
|---|---|
| TFTP (69/udp) | TFTP often contains your router and other network device configurations. Also, requires no authentication. |
| Finger (79/tcp) | Anyone on the internet can use this to compile a list of valid usernames, even phone numbers to use in a social engineering attack. |
| NNTP (119/tcp) | NNTP is great for storing contraband software and information. Block this so an attacker doesn't set up a warez site on your network. |
| NTP (123/tcp) | Your hosts rely on NTP for accurate time. Someone messing with this can trick your host into running cron jobs at the wrong time and really mess up your logs. |
| Syslog (514/udp) | Syslog is used for log messaging. Someone could use it to DoS your loghost in an attempt to prevent you from finding out what is really going on. |
| LDP (515/tcp) | LPD controls your print queues, and generally no authentication is required. Who needs someone on the internet emptying your print trays? |
| SNMP (161/tcp/udp & 162/tcp/udp) | SNMP runs our networks, but the authentication is terrible. And this means that anyone with access can gather routing and system information – and even change it possibly. |
| BGP (179/tcp) | BGP controls exterior routing. If someone can adjust your routing, they can spoof addresses, and this opens up a wide array of possible attacks. |
| SOCKS (1080/tcp) | SOCKS is generally used for proxying network traffic in through a firewall, but also can be used to bounce attacks through to other sites. Restrict access to people you know. |

*Table. Miscellaneous Port Vulnerabilities*

### Filter Syntax and Description

Following the convention already established, the ports are blocked as shown below.

```
block in log  quick on le0 proto udp from any to any port = 69
block in log  quick on le0 proto udp from any to any port = 514
block in log  quick on le0 proto tcp from any to any port = 79
block in log  quick on le0 proto tcp from any to any port = 119
block in log  quick on le0 proto tcp from any to any port = 123
block in log  quick on le0 proto tcp from any to any port = 515
block in log  quick on le0 proto tcp from any to any port = 179
block in log  quick on le0 proto tcp from any to any port = 1080
block in log  quick on le0 proto tcp/udp from any to any port 160 >< 163
```

## 11. Block ICMP

### Rationale

ICMP is the protocol that keeps the Internet running, because it handles error messages for broken tcp and udp communications. It can also be used to map a network, the ICMP echo request to find out which hosts are alive. Since it has no state and is normally necessary for many network functions, ICMP is used for nasty activity as well as nice. For example, attackers use if for scanning and reconnaisance, they can set up covert channels through a firewall with it, and they use it as the basis for new, unusual DoS attacks.

### Filter Syntax and Description

The policy is to block ICMP echo requests inbound and outgoing echo replies, time exceeded in transit, and unreachable messages. The first two inbound blocks take care of echo requests, Windows traceroute (which uses ICMP echo and echo replies), and Unix traceroute (which uses a high numbered UDP connection request). The last three lines block the echo replies, TIMEX's, and unreachable messages from leaving our firewalled environment.

```
block in log quick on le0 proto icmp from any to any icmp-type echo
block in log quick on le0 proto udp from any to any port 32999 >< 34001

block out log quick on le0 proto icmp from any to any icmp-type echorep
block out log quick on le0 proto icmp from any to any icmp-type timex
block out log quick on le0 proto icmp from any to any icmp-type unreach
```

## Ordering the rules

Now that we have a lot of rules, it's time to put them in a rational order. For our ordering criteria we chose the following principles:

- Group rules by common interface
- Group rules by common protocol
- Specifically designate quick on blocking rules to circumvent the "fall through" rule processing
- Use "block in log" to log attempts to connect to ports defined in this paper
- Allow outgoing connections, pings, etc… for user convenience (though not the most secure policy)

```
#!/sbin/ipf -f -
# ip filter 3.3
#
# Filter rules for GIAC FW Practicum
#
# John B. Folkerts
#
# le0 - external interface to 222.231.217.204 mask 255.255.255.255
# le1 - internal interface to 172.16.1.0 mask 255.255.255.0
#
# Pass lo0 (loopback) freely
pass out quick on lo0
pass in quick on lo0

# Section 1: Block all nasty packets

# Block packets which are too short to have a full IP header
#
block in log quick all with short

# Block packets which have source routing options
block in log quick all with opt lsrr
block in log quick all with opt ssrr

# Set up interface groups
#
```

```
block in  on le0 all head 100
block out on le0 all head 150
block in  on le1 all head 200
block out on le1 all head 250

# Block private addresses, broadcasts, and spoofed addresses
#
block in on le0 all head 100
  block in      quick from 10.0.0.0/8 to any        group 100
  block in      quick from 192.168.0.0/16 to any    group 100
  block in      quick from 172.16.0.0/12 to any     group 100
  block in log  quick from 127.0.0.0/8 to any       group 100
  block in log  quick from 172.16.1.0/24 to any     group 100

# Block/log all disallowed and suspicious tcp/udp ports
#
block in on le0 proto tcp/udp all head 110
  block in log  quick proto tcp/udp from any to any port = 111 group 110
  block in log  quick proto tcp/udp from any to any port = 2049 group 110
  block in log  quick proto tcp/udp from any to any port = 4045 group 110
  block in log  quick proto tcp/udp from any to any port = 135 group 110
  block in log  quick proto tcp/udp from any to any port = 445 group 110
  block in log  quick proto tcp/udp from any to any port < 20 group 110
  block in log  quick proto tcp/udp from any to any port = 37 group 110
  block in log  quick proto tcp/udp from any to any port 160 >< 163 group 110

# Block/log all disallowed and suspicious tcp traffic
#
block in on le0 proto tcp all head 115 group 110
  pass in log   quick proto tcp from any to 222.231.217.204/32 port = 22 keep state group
115
  pass in log   quick proto tcp from any to 222.231.217.204/32 port = 443 keep state group

115
  pass  in log  quick proto tcp from 222.231.216.1/32 to 222.231.217.204/32 port = 53
flags S/SA keep state group 115
  pass in log   quick proto tcp from 222.231.212.1/32 to 222.231.217.204/32 port = 25
flags S/SA keep state group 115
  block in log  quick proto tcp from any to any port 20 >< 24 group 115
  block in log  quick proto tcp from any to any port = 139 group 115
  block in log  quick proto tcp from any to any port 511 >< 515 group 115
  block in log  quick proto tcp from any to any port 5999 >< 6256 group 115
  block in log  quick proto tcp from any to any port = 53 group 115
  block in log  quick proto tcp from any to any port = 25 group 115
  block in log  quick proto tcp from any to any port 108 >< 111 group 115
  block in log  quick proto tcp from any to any port = 143 group 115
  block in log  quick proto tcp from any to any port = 80 group 115
  block in log  quick proto tcp from any to any port = 443 group 115
  block in log  quick proto tcp from any to any port 7999 >< 8011 group 115
  block in log  quick proto tcp from any to any port = 8080 group 115
  block in log  quick proto tcp from any to any port = 8888 group 115
  block in log  quick proto tcp from any to any port = 79 group 115
  block in log  quick proto tcp from any to any port = 119 group 115
  block in log  quick proto tcp from any to any port = 123 group 115
  block in log  quick proto tcp from any to any port = 515 group 115
  block in log  quick proto tcp from any to any port = 179 group 115
  block in log  quick proto tcp from any to any port = 1080 group 115


block in on le0 proto udp all head 120 group 110
  pass in log  quick proto udp from any to 222.231.217.204/32 port = 53 keep state group
120
  block in log  quick proto udp from any to any port = 137 group 120
```

```
  block in log  quick proto udp from any to any port = 138 group 120
  block in log  quick proto udp from any to any port = 69 group 120
  block in log  quick proto udp from any to any port = 514 group 120
  block in log quick on le0 proto udp from any to any port 32999 >< 34001

# Block/log all disallowed and suspicious icmp traffic
#
block in on le0 proto icmp all head 130
  block in log quick proto icmp from any to any icmp-type echo group 130

# Default block rule
#
block in on le0 all group 100

# Outbound external interface rules
#
block out on le0 all head 150
  block out log quick from any to 10.0.0.0/8        group 150
  block out log quick from any to 192.168.0.0/16    group 150
  block out log quick from any to 172.16.0.0/12     group 150
  block out log quick from any to 127.0.0.0/8       group 150
  block out log quick from 127.0.0.0/8 to any       group 150
  block out log quick proto icmp from any to any icmp-type echorep group 150
  block out log quick proto icmp from any to any icmp-type timex group 150
  block out log quick proto icmp from any to any icmp-type unreach group 150
  pass out on le0 all group 150

# Pass (for now) packets freely on le1
# Allow outbound stateful connections and UDP/ICMP request/replies
#
pass out on le1 all group 250
pass in quick on le1 proto tcp/udp from 172.16.1.0/24 to any keep state group 200
pass in quick on le1 proto icmp from 172.16.1.0/24 to any keep state group 200
```

## Testing

Use the following functionality test cases to ensure operability:

1. Ping and nslookup from internal addresses to external hosts
2. Connect to external web sites from internal addresses.
3. Ensure SMTP and DNS traffic move as expected.
4. Ensure that the HTTPS server is available to external addresses.

The following test cases should be non-functional and logged by the firewall:

1. Attempt connections to SMTP, and DNS from other than the external mail relay and dns secondary.
2. Attempt ping from external address
3. Attempt variety of vulnerable services

Assuming these work, test your firewall configuration with nmap. The following test cases should outline a pretty good workout for your setup.

```
# Full, normal TCP scan of firewall
nmap -v -P0 -sS '222.231.217.204'

# Full, normal UDP scan of firewall
nmap -v -P0 -sS -sU  -PI '222.231.217.204'

# Stealth FIN scanning
nmap -v -P0 -sF '222.231.217.204'

# Stealth FIN scanning with fragmentation
nmap -v -P0 -f -sF '222.231.217.204'

# Stealth Xmas scan with tcp F,U,P flags set
nmap -v -P0 -sX '222.231.217.204'
```

## Improving the Design

Note that a screened subnet would be a significant advantage to our architecture. By using a screened subnet to place our DNS server and other services on, we can provide an extra layer of security from BIND, sendmail, or web server attacks. It will require a bit more cost to set up, but the results are much more satisfactory. Assume ipnat redirects tcp/53, udp/53, tcp/25, and tcp/443 to a host on the screened subnet (172.16.2.2) and the screened subnet interface is le2:

```
pass  in log  quick on le0 proto tcp from 222.231.216.1/32 to 222.231.217.204/32
        port = 53 flags S/SA keep state
pass  in log  quick on le0 proto udp from any to 222.231.217.204/32 port = 53
        keep state
pass  in log  quick on le0 proto tcp from 222.231.212.1/32 to 222.231.217.204/32
        port = 25 flags S/SA keep state
pass  in log  quick on le0 proto tcp from any to 222.231.217.204/32
        port = 443 flags S/SA keep state
pass out on le0 all

pass out log  quick on le2 proto tcp from 222.231.216.1/32 to 172.16.2.2/32
        port = 53 flags S/SA keep state
pass out log  quick on le2 proto udp from any to 172.16.2.2/32 port = 53
        keep state
pass out log  quick on le2 proto tcp from 222.231.212.1/32 to 172.16.2.3/32
        port = 25 flags S/SA keep state
pass out log  quick on le2 proto tcp from any to 172.16.2.4/32
        port = 443 flags S/SA keep state
pass in log   quick on le2 proto udp from 172.16.2.0/24 to any keep state

block out on le1 all
```

## Summary

The goal in all of this was to show how common vulnerabilities can be readily kept at bay, despite a small budget and, in this case, only one external IP address. IP Filter provides a tremendous amount of flexibility at basically no cost. However, a firewall only exercises as much good sense as its owner is willing to put into it, and a solid understanding of protocol vulnerabilities and controls is essential to success.