Global Information Assurance Certification Paper

GIAC Firewall Practical

SANS Security DC 2000
Rick Thompson

Introduction

This is a practical assignment for GIAC certification from the SANS Security DC 200 conference, requiring a tutorial on implementation of several specified firewall filters. This is purely an academic and illustrative exercise, and must not be taken as a complete or practical ruleset for a real-world firewall.

Assignment

The assignment for this practical was as follows:

This track requires eleven short practical assignments. Please check your spelling and read through your wording, this is how the world will see you and we will not accept a second submission. You will be graded largely on the accuracy and educational value of your submission, but also on its appearance. Write a tutorial on how to implement each recommended action in the filtering policy below on your firewall or perimeter defense solution. Be explicit about the brand and version of perimeter defense. The policy is taken from the recommended perimeter defense actions in the "Top Ten" document. Screenshots, network traffic traces, firewall log information and URLs to find further information should all be used. Be certain to include the following:

1. These services have been the subject of published exploits, and are often targeted by attackers. They should generally be denied where there is no specific requirement for them.
2. Relevant information about the behavior of the protocol or service on the network
3. Syntax of the filter
4. Description of each of the parts of the filter
5. The filter is straightforwardly applied to both interfaces of Firewall 1, above.
6. If this filter is order dependent, what other rules should this filter precede and follow**
7. Explain how to test the filter

Be certain to point out any tips, tricks, or gotcha's.

** You may find it easier to create a section of your practical that describes the order you would apply all of the rules rather than trying to do it with each policy cluster. Be certain to explain your reasons for the order you choose, we cannot read your mind.

Security Policy

In this section, we list ports that are commonly probed and attacked. Blocking these ports is a minimum requirement for perimeter security, not a comprehensive firewall specification list. A far better rule is to block all unused ports. And even if you believe these ports are blocked, you should still actively monitor them to detect intrusion attempts. A warning is also in order. Blocking some of the ports in the following list may disable needed services. Please consider the potential effects of these recommendations before implementing them.

1. Block "spoofed" addresses-- packets coming from outside your company sourced from internal addresses or private (RFC1918 and network 127) addresses. Also block source routed packets.

2. Login services-- telnet (23/tcp), SSH (22/tcp), FTP (21/tcp), NetBIOS (139/tcp), rlogin et al (512/tcp through 514/tcp)
3. RPC and NFS-- Portmap/rpcbind (111/tcp and 111/udp), NFS (2049/tcp and 2049/udp), lockd (4045/tcp and 4045/udp)
4. NetBIOS in Windows NT -- 135 (tcp and udp), 137 (udp), 138 (udp), 139 (tcp). Windows 2000 - earlier ports plus 445(tcp and udp)
5. X Windows -- 6000/tcp through 6255/tcp
6. Naming services-- DNS (53/udp) to all machines which are not DNS servers, DNS zone transfers (53/tcp) except from external secondaries, LDAP (389/tcp and 389/udp)
7. Mail-- SMTP (25/tcp) to all machines, which are not external mail relays, POP (109/tcp and 110/tcp), IMAP (143/tcp)
8. Web-- HTTP (80/tcp) and SSL (443/tcp) except to external Web servers, may also want to block common high-order HTTP port choices (8000/tcp, 8080/tcp, 8888/tcp, etc.)
9. "Small Services"-- ports below 20/tcp and 20/udp, time (37/tcp and 37/udp)
10. Miscellaneous-- TFTP (69/udp), finger (79/tcp), NNTP (119/tcp), NTP (123/tcp), LPD (515/tcp), syslog (514/udp), SNMP (161/tcp and 161/udp, 162/tcp and 162/udp), BGP (179/tcp), SOCKS (1080/tcp)
11. ICMP-- block incoming echo request (ping and Windows traceroute), block outgoing echo replies, time exceeded, and unreachable messages

Topology Assumptions

The filter rules written in response to the assignment are obviously dependent upon the environment's addressing scheme and topology. For this tutorial, we will assume the network structure depicted in figure 1 below, for "somecompany.com," which we will (falsely) assume uses the 1.1.0.0 class B network, subnetted into three class C's.

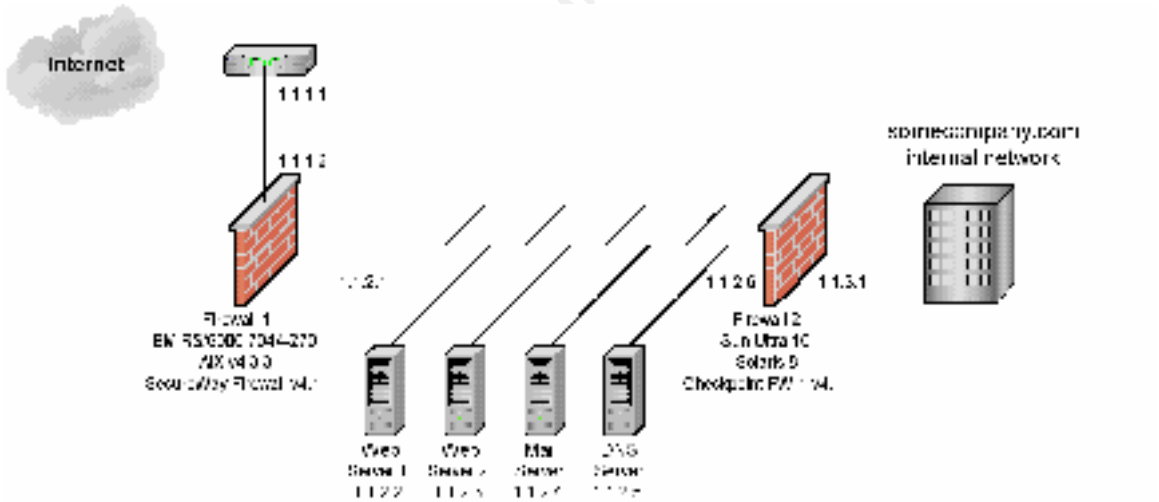

Figure 1
Topology Assumed for Tutorial

This is not an unusual setup.

We will assume that the router shown is some unspecified Cisco device (adequate for doing basic filtering on the traffic passing through it), running IOS 11.3 or higher. The anti-spoofing filters will be applied in this router (though duplicated on Firewall 1 for safety's sake.)

Firewall 1 protects the screened subnet from the "outside world." Web, mail, and external DNS servers reside on this 1.1.2.0 / 255.255.255.0 class C subnet.

Firewall 2 separates the screened subnet from somecompany's intranet. This machine runs a different firewall product on a different operating system from Firewall 1. Even if a gaping exploit is found in the products on Firewall 1, resulting in the utter compromise of the servers on the screened subnet, this still provides protection for somecompany's internal network. Both products are capable of ipsec tunneling.

Product Selection

The firewall product chosen for Firewall 1 is IBM's SecureWay Firewall v4.1.

The SecureWay product combines packet filtering with proxies, making it a relatively speedy product. Its drawbacks are that: (1) it is purely packet filtering and proxy, without state tables; and, (2) it is relatively difficult to administer (somewhat moreso than ipchains in command-line mode, somewhat moreso than Firewall-1 through the administrative GUI client). It was selected for this illustration because: (1) the ruleset it produces is clearly readable – virtually firewall pseudocode; and, (2) the speed of packet filtering is not a bad choice for these external servers, especially since the intranet – and presumably the backend database servers powering the webservers – is still protected through the stateful inspection of Firewall 2.

Although the product will also run on NT, the Unix operating system chosen is generally regarded as being more secure.

Rule Syntax

Syntax of the SecureWay filter ruleset is illustrated in the example below, allowing HTTP to Webserver 1 (in figure 1 above) from any address:

```
# Allow World HTTP to Webserver 1 and reply
permit 0.0.0.0 0.0.0.0 1.1.2.2 255.255.255.255 tcp gt 1023 eq 80 nonsecure route inbound
permit 0.0.0.0 0.0.0.0 1.1.2.2 255.255.255.255 tcp gt 1023 eq 80 secure route outbound
permit 1.1.2.2 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 80 gt 1023 secure route inbound
permit 1.1.2.2 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 80 gt 1023 nonsecure route
outbound
```

The first two rules are for HTTP inbound to Webserver 1. The latter two are for the reply from Webserver 1 to the client.

The first two octets in each rule represent the source IP address and netmask. In rule one, these are "0.0.0.0 0.0.0.0" representing any address whatsoever. The second two octets in each rule represent the destination IP address and netmask. In rule one, these are "1.1.2.2 255.255.255.255" for the single host Webserver 1. The next field is the protocol, in this case, TCP. The next two fields are the source port. In rule one this is "gt 1023" or any ephermal port above 1023. The next two fields are the destination port "eq 80", or "equal to port 80." The final three fields represent the traffic direction: this traffic is expected to arrive inbound on a nonsecure (exterior) interface, to be routed through the firewall. Basically, the first two rules allow the traffic to come in the front (internet) side, and go out the back (screened subnet) side. These two rules:

```
permit 0.0.0.0 0.0.0.0 1.1.2.2 255.255.255.255 tcp gt 1023 eq 80 nonsecure route inbound
permit 0.0.0.0 0.0.0.0 1.1.2.2 255.255.255.255 tcp gt 1023 eq 80 secure route outbound
```

could have been combined into this:

```
permit 0.0.0.0 0.0.0.0 1.1.2.2 255.255.255.255 tcp gt 1023 eq 80 both route both
```

but this method allows webs “surfing” out while the two rules do not.

The second two rules allow the server reply to the client request. These reverse the source and destination (and the traffic flow). Additionally, these require that the ACK flag be set (similar to an “established” connection in a Cisco router ACL.)

Rules to simply deny traffic are simpler (since traffic flow need not be considered), for example, to deny all telnet:

```
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 23 both both both
```

A slight amount of detail has been left off of the end of each rule, related to logging and fragmentation control. These syntactic details are handled by the user interface, and have been omitted here to make the rules conceptually more clear.

All SecureWay rules follow the same pattern, and on the specified AIX host would be stored in /etc/security/fwfilters.cfg.

I have attempted to keep these rules sufficiently generic so that they can be applied to earlier versions of the product (sold as IBM eNetwork Firewall).

It should be noted that the product at level 3 and above does not anticipate that these rulesets will be edited manually. The fwfilters.cfg file is still produced, and is human-readable, but it is automatically generated through the administrative GUI client, and manual edits will likely be overwritten. This practical assignment is not a substitute for the product documentation.

Rule Order

The order in which firewall rules appear is important both for implementation of filtering according to the administrator’s intent (most firewall products follow an algorithm of “first match,” rather than “best fit”), and for firewall performance in terms of network throughput. A typical ordering scheme appears in figure 2 below:

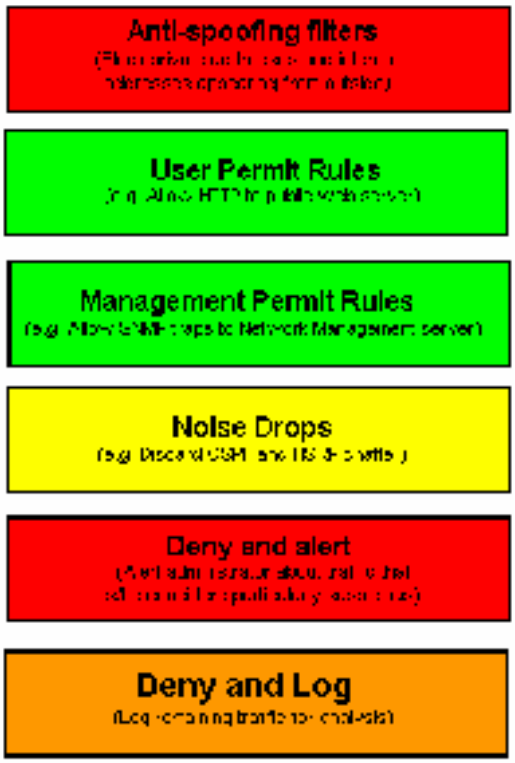

Figure 2
Typical ruleset overview

The rationale behind this arrangement (which assumes that the particular firewall uses a “first match” strategy, as most do) is fairly simple.

The initial anti-spoof filters are placed at the top because: (a) this traffic is never legitimate, even if it would otherwise be allowed from a legitimate address and thus must precede any “permits;” and (b) it can be disposed of with little overhead, using simple packet filtering.

Following the anti-spoofing are user permit rules. This includes rules such as allowing public access to port 80 of a public web server, or allowing your public mail server to send and receive mail. These are close to the top because logically they should be the most often hit rules. This arrangement should maximize performance by minimizing the amount of processing the firewall must do before determining that the traffic should be allowed.

Next are connections permitted for management and administrative purposes. These may include connections like firewall administration, or various network monitoring tools to and from trusted hosts.

If the firewall is still processing the packet(s) after this block of rules, then it is destined to be denied. Our only remaining choices are how to handle the denial.

First, normal “noise” is discarded. This is traffic, such as router broadcasts, which should not be passed along to the interior, but which represents expected chatter and is not a sign of suspicious activity.

Next, traffic which the administrator regards as particularly troublesome is discarded and an alert generated. Here we might find a "lockdown" rule, generating an alert if there is forbidden traffic coming at the firewall itself, or a rule which causes an alert if some server under our supposed control suddenly begins communication on port 31337.

Finally, all remaining traffic is discarded and logged in as much detail as possible. Note that in a real production environment, this rule would cover most of the "deny" requirements in the above GIAC assignment. If the firewall is set up with the philosophy that "anything not specifically permitted is denied and logged" there is no reason to have a rule to specifically block X Windows.

The GIAC assignment, then, must be regarded primarily as a requirement to demonstrate an ability to construct filters and explain their rationale. The resulting firewall ruleset is both dubious and incomplete, and in no way intended as a pattern for real world implementation.

With that in mind, there is still a requirement to put the rules in a logical and internally consistent order. We can stay fairly close to the more common model, by making two assumptions: (1) the rules required do not represent the complete ruleset which would be in use, and (2) the administrator has some special reason to be specifically denying this traffic – perhaps s/he is being forced to have "allow and log" as their final rule, or perhaps it is important to them to associate the denials with a specific rule for purposes of log parsing. Those assumptions allow us to order the rules as shown in figure 3:

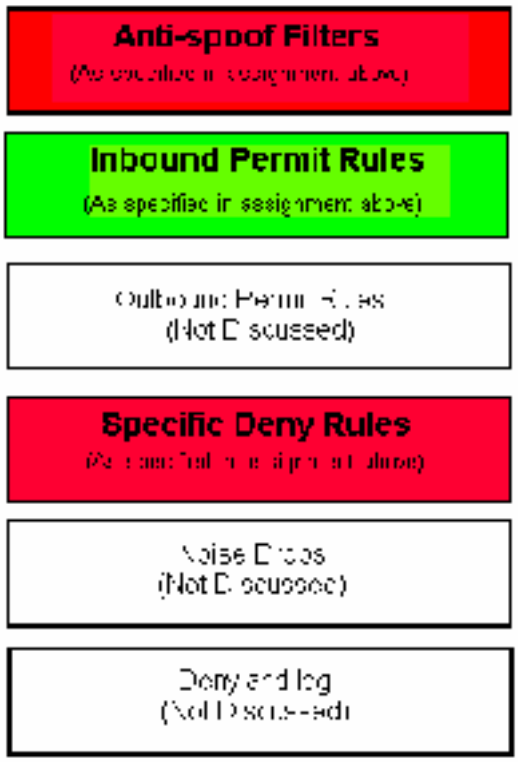

Figure 3
Ruleset Assumed for Tutorial

Note here that this ordering will result in some of the required rules being separated into different blocks. The requirement to block "HTTP (80/tcp) and SSL (443/tcp) *except to external Web servers*," for instance, will demand both "permit" rules in the "Inbound Permit Rules" block and

"deny" rules in the "Specific Deny Rules" block. These rules will be specifically noted below, as the required filters are explained in order.

Block Spoofed Addresses

"Block "spoofed" addresses-- packets coming from outside your company sourced from internal addresses or private (RFC1918 and network 127) addresses. Also block source routed packets."

Rationale and Behavior: Packets arriving on the external interface from outside should never claim to have originated inside your secure network(s). This is generally either the result of serious routing misconfiguration, or a deliberate attempt to bypass your security, and in either case should not be forwarded. Private addresses as specified in RFC1918 (10.n.n.n, etc. – see http://www.cis.ohio-state.edu/htbin/rfc/rfc1918.html) and 127. (loopback) addresses cannot be legitimately routed through the internet, and as such are guaranteed to be discardable at your gateway. Source routed packets attempt to specify a return route that the traffic should or must take. This is seldom used in normal traffic, and is more likely an attempt to force your traffic through some hostile or comprised node.

Topology and Application: Ideally, these should be applied at the router. In the case of a Cisco router, this could be accomplished by adding an ACL something like this:

```
Interface Serial 0
       ip address <router serial interface> <router serial netmask>
       ip access-group 11 in

access-list 11 deny 192.168.0.0 0.0.255.255
access-list 11 deny 172.16.0.0 0.15.255.255
access-list 11 deny 10.0.0.0 0.255.255.255
access-list 11 deny 127.0.0.0 0.255.255.255
access-list 11 deny 1.1.0.0 0.0.255.255
access-list 11 permit any
```

and then issuing the

```
no ip source-route
```

command.

However, we will not rely solely on the router, and will implement these rules in the firewall as well.

Description: On the external interface, we will deny all packets with source addresses matching the specified list. Note that unlike the router, which would require a second ACL for egress filtering (an excellent idea, if one wishes to be a "good neighbor," but not specified in the assignment) this will also prevent packets with these source addresses from being transmitted by the interface.

The SecureWay firewall will not allow the blocking of source-routed packets as a firewall rule, but this can easily be done at the operating system level by adding the commands

```
no –o ipsrcrouterecv=0
no –o ipsrcrouteforward=0
```

into the /etc/rc.net startup file

Syntax:

```
# anti-spoofing
deny 1.1.3.0 255.255.255.0 0.0.0.0 0.0.0.0 all any 0 any 0 nonsecure both both
deny 10.0.0.0 255.0.0.0 0.0.0.0 0.0.0.0 all any 0 any 0 nonsecure both both
deny 127.0.0.0 255.0.0.0 0.0.0.0 0.0.0.0 all any 0 any 0 nonsecure both both
deny 172.16.0.0 255.240.0.0 0.0.0.0 0.0.0.0 all any 0 any 0 nonsecure both both
deny 192.168.0.0 255.255.0.0 0.0.0.0 0.0.0.0 all any 0 any 0 nonsecure both both
```

Dependencies: These should appear in the "anti-spoofing" block in figure 3, above.

Test: Testing can be accomplished by obtaining some tool for the creation of spoofed packets, and directing this traffic through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Comments: Disabling source routing may inconvenience network support personnel who are accustomed to use it as a troubleshooting technique.

Block Login Services

"Login services-- telnet (23/tcp), SSH (22/tcp), FTP (21/tcp), NetBIOS (139/tcp), rlogin et al (512/tcp through 514/tcp) "

Rationale and Behavior: These are the most straightforward and "normal" methods to take control of a machine. There is no reason to allow these from the Internet.

Topology and Application: Note that this ruleset will be applied on Firewall 1 above. This makes it quite simple. It can simply be blocked. The filters on Firewall 2 might well require some permits as well, for management of the servers within the screened subnet.

Description: On both interfaces, we will deny all packets destined for the specified port and protocol combinations.

Syntax:

```
# Block ftp
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 21 both both both

# Block ssh
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 22 both both both

# Block telnet
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 23 both both both

# Block nb-sess
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 139 both both both

# Block rlogin, etc
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 512 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 513 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 514 both both both
```

Dependencies: No strict order dependencies. This would fit in the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Comments: Note that this will prevent the use of Microsoft File and Print Sharing across the internet to any somecompany server. If this is the goal, it might be well to disable the remaining

NetBIOS ports (135-139) as well. Further note that this will block ssh management to the firewall from the secure network.

Block RPC and NFS

"RPC and NFS-- Portmap/rpcbind (111/tcp and 111/udp), NFS (2049/tcp and 2049/udp), lockd (4045/tcp and 4045/udp) "

Rationale and Behavior: Theses services allow people outside your organization to execute code on your machines, and have been the targets of several published exploits.

Topology and Application: The filter is straightforwardly applied to both interfaces of Firewall 1, above.

Description: On both interfaces, we will deny all packets destined for the specified port and protocol combinations.

Syntax:

```
# Block portmap/rpcbind
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 111 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 111 both both both

# Block NFS
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 2049 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 2049 both both both

# Block lockd
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 4045 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 4045 both both both
```

Dependencies: No strict order dependencies. This would fit in the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Block NetBIOS

"NetBIOS in Windows NT -- 135 (tcp and udp), 137 (udp), 138 (udp), 139 (tcp). Windows 2000 - earlier ports plus 445(tcp and udp) "

Rationale and Behavior: These Microsoft services are often abused and have been the subjects of published exploits. The network is particularly at risk if all Microsoft fixes have not been applied.

Topology and Application: The filter is straightforwardly applied to both interfaces of Firewall 1, above.

Description: On both interfaces, we will deny all packets destined for the specified port and protocol combinations.

Syntax:

```
# Block NetBIOS
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 135 both both both
```

```
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 135 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 136 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 137 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 138 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 139 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 445 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 445 both both both
```

Dependencies: No strict order dependencies. This would fit in the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Comments: Note that this will prevent the use of Microsoft File and Print Sharing across the internet to any somecompany server.

Block X Windows

"X Windows -- 6000/tcp through 6255/tcp "

Rationale and Behavior: X windows, and various associated tools, have been the subject of published exploits. Further, it can be used for purposes of a remote terminal, allowing login access. It should be disabled unless specifically required.

Topology and Application: This is an excellent example of the reason for adopting a strategy of "deny all" at the bottom of the rules list. On the SecureWay firewall, and many others, blocking a large group of ports is not directly supported. Very few firewall administrators would ever approach this by coding the 255 rules required to do it port by port.

Description: On both interfaces, we can fulfill the assignment by denying all packets destined for the specified port and protocol combinations. As noted above, this is not a real-world approach. since it generates a huge list, as seen below.

Syntax:

```
# Block X-windows
# This would be better done in a deny all rule
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 6000 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 6001 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 6002 both both both
[. . .]
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 6253 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 6254 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 6255 both both both
```

Dependencies: No *strict* order dependencies. This would fit in the "specific deny" block in figure 3, above, however, parsing through this list of rules will be time-consuming, and it should be placed as far down in the ruleset as possible.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Block Inappropriate DNS

"Naming services-- DNS (53/udp) to all machines which are not DNS servers, DNS zone transfers (53/tcp) except from external secondaries, LDAP (389/tcp and 389/udp) "

Rationale and Behavior: BIND has been the subject of many published exploits, and is often targeted by attackers. All of these services give out information about your network, and need to be strictly controlled.

Topology and Application: The filter is straightforwardly applied to both interfaces of Firewall 1, above.

Description: On both interfaces, we will first allow traffic to the designated nameserver, then deny all packets destined for the specified port and protocol combinations. For illustrative purposes we will consider that we have an external secondary nameserver at address 4.3.2.1, owned by our ISP.

Syntax:

```
# Permit UDP queries to Nameserver
permit 0.0.0.0 0.0.0.0 1.1.2.5 255.255.255.255 udp any 0 eq 53 nonsecure route inbound
permit 0.0.0.0 0.0.0.0 1.1.2.5 255.255.255.255 udp any 0 eq 53 secure route outbound
permit 1.1.2.5 255.255.255.255 0.0.0.0 0.0.0.0 udp eq 53 any 0 secure route inbound
permit 1.1.2.5 255.255.255.255 0.0.0.0 0.0.0.0 udp eq 53 any 0 nonsecure route outbound

# Permit zone transfers to ISP secondary
permit 4.3.2.1 255.255.255.255 1.1.2.5 255.255.255.255 tcp any 0 eq 53 nonsecure route
inbound
permit 4.3.2.1 255.255.255.255 1.1.2.5 255.255.255.255 tcp any 0 eq 53 secure route
outbound
permit 1.1.2.5 255.255.255.255 4.3.2.1 255.255.255.255 tcp/ack eq 53 any 0 secure route
inbound
permit 1.1.2.5 255.255.255.255 4.3.2.1 255.255.255.255 tcp/ack eq 53 any 0 nonsecure
route outbound

# Deny remaining DNS and LDAP
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 upd any 0 eq 53 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 53 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 389 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 389 both both both
```

Dependencies: Notice that the "permit" rules *must* precede the "deny" rules. The permit rules would fit into the "specific permit" block, and the denies into the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Comments: Note that the syntax shown assumes that we cannot be certain what version of BIND the ISP may be running. Further, the filters used will deny "long" DNS queries as well as zone transfers. Finally, a split-DNS scheme should be used to limit the information that the nameserver will give out to the public. Note that the SecureWay firewall comes with a component that allows the firewall itself to act as a more split brain DNS server. In a production environment, this would likely be used to replace the dedicated external nameserver, but is outside the scope of this assignment.

Block Inappropriate Mail Services

"Mail-- SMTP (25/tcp) to all machines, which are not external mail relays, POP (109/tcp and 110/tcp), IMAP (143/tcp) "

Rationale and Behavior: Sendmail has been the subject of many, many, published exploits, and is often targeted by attackers.

Topology and Application: The filter is straightforwardly applied to both interfaces of Firewall 1, above. Note that since we are only concerned with Firewall 1, we can summarily drop services such as POP3, which depending on network design might be required through Firewall 2.

Description: On both interfaces, we will first allow traffic to the designated mail server, then deny all packets destined for the specified port and protocol combinations. Note that for SMTP, we will deny any traffic outbound as well.

Syntax:

```
# Permit SMTP to Mail Server
permit 0.0.0.0 0.0.0.0 1.1.2.4 255.255.255.255 tcp any 0 eq 25 nonsecure route inbound
permit 0.0.0.0 0.0.0.0 1.1.2.4 255.255.255.255 tcp any 0 eq 25 secure route outbound
permit 1.1.2.4 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 25 any 0 secure route inbound
permit 1.1.2.4 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 25 any 0 nonsecure route
outbound

# Permit SMTP from Mail Server
permit 1.1.2.4 255.255.255.255 0.0.0.0 0.0.0.0 tcp any 0 eq 25 secure route inbound
permit 1.1.2.4 255.255.255.255 0.0.0.0 0.0.0.0 tcp any 0 eq 25 nonsecure route outbound
permit 0.0.0.0 0.0.0.0 1.1.2.4 255.255.255.255 tcp/ack eq 25 any 0 nonsecure route
inbound
permit 0.0.0.0 0.0.0.0 1.1.2.4 255.255.255.255 tcp/ack eq 25 any 0 secure route outbound

# Deny remaining Mail
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 25 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp eq 25 any 0 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 109 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 110 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 143 both both both
```

Dependencies: Notice that the "permit" rules *must* precede the "deny" rules. The permit rules would fit into the "specific permit" block, and the denies into the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Comments: Note that the SecureWay firewall comes with a "securemail" component that allows the firewall itself to act as a more secure mail relay. In a production environment, this would likely be used to replace the external mail server machine, but is outside the scope of this assignment. Note also that these firewall filters provide no protection against having your external mail server used as a spam relay. Appropriate rules to prevent this should be placed into /etc/sendmail.cf.

Block Inappropriate Web Services

"Web-- HTTP (80/tcp) and SSL (443/tcp) except to external Web servers, may also want to block common high-order HTTP port choices (8000/tcp, 8080/tcp, 8888/tcp, etc.) "

Rationale and Behavior: These services – particularly Microsoft IIS -- have been the subject of published exploits, and are often targeted by attackers. They should generally be denied where there is no specific requirement for them. Although an administrator may think that this can be omitted, since machines other than webservers will not respond to HTTP anyway, the firewall needs to protect against traffic to "unintentional" webservers: machines where IIS has been installed, and a webserver accidentally started, for example, or malicious code such as Back Orifice mini-http-servers, or netcat listeners placed to take advantage of port 80 being allowed through the firewall.

Topology and Application: It is assumed that Webserver 1 is the primary HTTP server, while Webserver 2 has additional hardening appropriate to an SSL (HTTPS) server. The permit filters are then straightforwardly applied to both interfaces of Firewall 1, above. Notice that the deny filters are applied only to inbound traffic on the nonsecure interface, to allow web browsing out.

Description: On both interfaces, we will first allow traffic to the designated webservers, then deny all packets destined for the specified port and protocol combinations.

Syntax:

```
# Permit HTTP to Web Server 1
permit 0.0.0.0 0.0.0.0 1.1.2.2 255.255.255.255 tcp any 0 eq 80 nonsecure route inbound
permit 0.0.0.0 0.0.0.0 1.1.2.2 255.255.255.255 tcp any 0 eq 80 secure route outbound
permit 1.1.2.2 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 80 any 0 secure route inbound
permit 1.1.2.2 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 80 any 0 nonsecure route
outbound

# Permit SSL to Web Server 2
permit 0.0.0.0 0.0.0.0 1.1.2.3 255.255.255.255 tcp any 0 eq 443 nonsecure route inbound
permit 0.0.0.0 0.0.0.0 1.1.2.3 255.255.255.255 tcp any 0 eq 443 secure route outbound
permit 1.1.2.3 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 443 any 0 secure route inbound
permit 1.1.2.3 255.255.255.255 0.0.0.0 0.0.0.0 tcp/ack eq 443 any 0 nonsecure route
outbound

# Deny remaining inbound web traffic
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 80 nonsecure both inbound
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 443 nonsecure both inbound
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 8000 nonsecure both inbound
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 8008 nonsecure both inbound
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 8080 nonsecure both inbound
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 8888 nonsecure both inbound
```

Dependencies: Notice that the "permit" rules *must* precede the "deny" rules. The permit rules would fit into the "specific permit" block, and the denies into the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Block "Small Services"

'"Small Services"-- ports below 20/tcp and 20/udp, time (37/tcp and 37/udp) '

Rationale and Behavior: These services, as governed by the IANA, are relatively trivial, with few associated exploits, but many of them are often turned on by default after an operating system install, and serve only to provide attackers with a toehold into your network. There is usually no reason to allow this traffic from the Internet. These services include:

```
tcpmux            1/tcp    TCP Port Service Multiplexer
tcpmux            1/udp    TCP Port Service Multiplexer
compressnet       2/tcp    Management Utility
compressnet       2/udp    Management Utility
compressnet       3/tcp    Compression Process
compressnet       3/udp    Compression Process
#                          Bernie Volz <VOLZ@PROCESS.COM>
#                 4/tcp    Unassigned
#                 4/udp    Unassigned
rje               5/tcp    Remote Job Entry
rje               5/udp    Remote Job Entry
#                 6/tcp    Unassigned
#                 6/udp    Unassigned
echo              7/tcp    Echo
echo              7/udp    Echo
#                 8/tcp    Unassigned
#                 8/udp    Unassigned
discard           9/tcp    Discard
discard           9/udp    Discard
#                10/tcp    Unassigned
#                10/udp    Unassigned
systat           11/tcp    Active Users
systat           11/udp    Active Users
#                12/tcp    Unassigned
#                12/udp    Unassigned
daytime          13/tcp    Daytime
daytime          13/udp    Daytime
#                14/tcp    Unassigned
#                14/udp    Unassigned
#                15/tcp    Unassigned [was netstat]
#                15/udp    Unassigned
#                16/tcp    Unassigned
#                16/udp    Unassigned
qotd             17/tcp    Quote of the Day
qotd             17/udp    Quote of the Day
msp              18/tcp    Message Send Protocol
msp              18/udp    Message Send Protocol
chargen          19/tcp    Character Generator
chargen          19/udp    Character Generator

time             37/tcp    Time
time             37/udp    Time
```

A complete list may be found at http://www.sockets.com/services.htm

Topology and Application: The filter is straightforwardly applied to both interfaces of Firewall 1, above. Note that simply applying a “small services” block on the router will not fulfill the assignment. It should probably be done, but it will not block all of the ports below 20. In a production environment, these would probably be caught on the firewall by means of a final “deny all” rule.

Description: On both interfaces, we will deny all packets destined for the specified ports. (There should similarly be little reasons for any protocol other than TCP or UDP to be using any of these ports.)

Syntax: Syntax of the filter

```
# Deny small services
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 1 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 2 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 3 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 4 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 5 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 6 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 7 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 8 both both both
```

```
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 9 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 10 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 11 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 12 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 13 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 14 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 15 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 16 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 17 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 18 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 19 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0 eq 37 both both both
```

Dependencies: No strict order dependencies. This would fit in the "specific deny" block in figure 3, above.

Test: Testing is straightforward. For registered ports, simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial. For the unassigned ports, some other tool would be required. Telnet would suffice for the TCP ports, but another option would be needed for testing the UDP denials.

Miscellaneous Blocks

"Miscellaneous-- TFTP (69/udp), finger (79/tcp), NNTP (119/tcp), NTP (123/tcp), LPD (515/tcp), syslog (514/udp), SNMP (161/tcp and 161/udp, 162/tcp and 162/udp), BGP (179/tcp), SOCKS (1080/tcp) "

Rationale and Behavior: Although most of these services have few associated exploits, many of them are often turned on by default after an operating system install, and serve only to provide attackers with a toehold into your network. There is usually no reason to allow this traffic from the Internet. Three are worthy of special attention:

tftp, depending on the configuration of the operating system, can allow transfer of files to and from your systems without authentication. It should always be disabled from the internet.

finger, depending on the configuration of the operating system, can return information about users and processes on your system without authentication. It should always be disabled from the internet.

snmp, if left with the default community strings, can provide attackers with highly detailed information, and if configured poorly enough, can allow them to change system values with minimal, easily-guessed authentication.

Topology and Application: These filters are straightforwardly applied to both interfaces of Firewall 1, above.

Description: On both interfaces, we will deny all packets destined for the specified port and protocol combinations.

Syntax:

```
# Deny miscellaneous
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 69 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 79 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 119 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 123 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 161 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 161 both both both
```

```
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 162 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 162 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 udp any 0 eq 514 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 515 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 179 both both both
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 tcp any 0 eq 1080 both both both
```

Dependencies: No strict order dependencies. This would fit in the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Comments: Note that the specified filters block this traffic outbound as well

Block Selected ICMP

"ICMP-- block incoming echo request (ping and Windows traceroute), block outgoing echo replies, time exceeded, and unreachable messages "

Rationale and Behavior: These services allow an attacker to map your network, and can even be used for covert communications and control. They are defined in RFC 792. (see http://www.cis.ohio-state.edu/htbin/rfc/rfc792.html).

Topology and Application: The filter is straightforwardly applied to both interfaces of Firewall 1, above.

Description: On both interfaces, we will deny all packets destined for the specified port and protocol combinations.

Syntax: Syntax of the filter

```
# Deny selected ICMP unreachables
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 icmp eq 3 any 0 both both both

# Deny echo request
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 icmp eq 8 any 0 both both both

# Deny echo reply
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 icmp eq 0 any 0 both both both

# Deny ICMP time exceeded
deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 icmp eq 11 any 0 both both both
```

Dependencies: No strict order dependencies. This would fit in the "specific deny" block in figure 3, above.

Test: Testing is quite straightforward. Simply use the ordinary and expected client(s), and attempt to connect through the firewall, observing that it fails, then check the firewall logs for a record of the denial.

Comments: Application of these filters is an excellent idea from a security perspective, but may conflict with policy or business needs. Management may, for example, demand that people should be able to ping the webservers from the internet. Network support may require the ability to traceroute. Various products may refuse to communicate if they cannot use "unreachable" messages to determine MTU's or the like.

A Note on SecureWay log entries

The firewall log entries for the SecureWay firewall, follow the general format of:

ICA1036i;#:;*rule_number*;R:*permit/deny*;
inbound/outbound:;*interface_address*;s:;*source_address*;d:;*dest_address*;p
:;*protocol*;sp:;*source_port*;dp:;*dest_port*;r:;*routed/local*;a:;*secure/nons
ecure*;f:;*yes/no*;T:;*tunnel_id*;e:;*C/D/n*;l:;*packet_len*;

To take the following example:

ICA1036i;#:;263;R:d; i:;1.1.1.2;s:;4.3.2.1;d:;1.1.2.3;p:;tcp;sp:;2048;dp:;31337;r:;r;a:;n;f:;n;T:;0;e:;n
;l:;40;

This packet matched rule number 263. It was denied.
It arrived on interface 1.1.1.2
It came from source 4.3.2.1 destined for 1.1.2.3
It was using tcp from source port 2048 to destination port 31337
It was requested to be routed through the firewall, rather than destined for the firewall itself
It arrived on a non-secure interface, was neither fragmented nor tunneled, and was 40 bytes inn
length