Global Information Assurance Certification Paper

GIAC Certified Firewall Analyst (GCFW)
Practical Assignment Version 1.9

GIAC Enterprises: "Fortunes for Fundraisers"

Sam Wilson
3 April 2003

Table of Contents

Summary 5
1 Security Architecture 6
1.1 Introduction 6
1.2 Requirements 6
1.2.1 Customer Requirements 7
1.2.2 Supplier Requirements 7
1.2.3 Partner Requirements 8
1.2.4 On-Site Employee Requirements 8
1.2.5 Remote Employee Requirements 8
1.3 Architecture 9
1.3.1 Network Addressing 9
1.3.2 Border Router & VPN 11
1.3.3 Firewall 11
2 Security Policy and Tutorial 13
2.1 Router and VPN Configuration 13
2.1.1 Introduction 13
2.1.2 Global Configuration Rules 14
2.1.3 VPN & IPSec Configuration Rules 19
2.1.4 Interface Configuration Rules 20
2.1.5 Access Control Lists 22
2.2 Firewall Installation and Configuration Tutorial 24
2.2.1 Introduction 24
2.2.2 Installation Procedures 24
2.2.2.1 Download 24
2.2.2.2 Compilation 24
2.2.2.3 Installation 25
2.2.3 Configuring and Using IPFilter 26
2.2.3.1 Creating Rule Sets 26
2.2.3.2 Rule Syntax 26
2.2.3.3 Active and Inactive Rule Sets 29
2.2.3.4 Rule Set Behavior 30
2.2.4 The 'ipfstat' Utility 30
2.3 Firewall Configuration 32
2.3.1 Introduction 32
2.3.2 IPFilter Rule Set 32
2.3.2.1 `qfe0` Interface – Inbound from the Internet 32
2.3.2.2 `qfe0` Interface - Outbound to the Internet 36
2.3.2.3 `qfe1` Interface - Inbound from the Service Network 37
2.3.2.4 `qfe1` Interface – Outbound to the Service Network 39
2.3.2.5 `qfe2` Interface - Inbound from the Proxied Network 40
2.3.2.6 `qfe2` Interface – Outbound to the Proxied Network 41
2.3.2.7 `qfe3` Interface – Inbound from the Internal Network 42
2.3.2.8 `qfe3` Interface – Outbound to the Internal Network 43

2.3.2.9 `Additional Rules` 44
3 Verify the Firewall Policy 45
3.1 Audit Plan 45
3.1.1 Network Used for the Audit 45
3.1.2 Planned Tests 45
3.1.3 Estimated Costs 46
3.2 Audit Results and Analysis 46
3.2.1 ICMP Echo Requests 46
3.2.2 Telnet and rlogin Connections 48
3.2.3 Firewall Scans with Nmap 51
3.2.3.1 Scan of the `qfe0` Interface 51
3.2.3.2 Scans of the `qfe1`, `qfe2`, and `qfe3` Interfaces 54
3.2.4 Nmap Scans Across the Firewall 57
3.2.4.1 External Network to Service Network 58
3.2.4.2 Service Network to Proxied Network 59
3.2.4.3 Proxied Network to Internal Network 64
3.3 Audit Summary 67
4 Design under Fire 69
4.1 Introduction 69
4.2 Attack on the Firewall 71
4.3 DDoS Attack 72
4.4 Attack on an Internal System 74
4.5 Conclusion 76
A Appendix 1: Description of IPFilter 77
A.1 Introduction 77
A.2 IPFilter Documentation 77
A.3 IPFilter Support 78
A.4 Pre-compiled Binary 78
References 79

Table of Figures

Figure 1: GIAC Network Diagram 10
Figure 2: CISCO ConfigMaker Diagram 13
Figure 3: Network for Auditing the Firewall 45
Figure 4: Echo Requests from 10.1.1.1 47
Figure 5: TCP Scan of `qfe0` Interface 51
Figure 6: UDP Scan of `qfe0` Interface 54
Figure 7: TCP Scan of Firewall Interface `qfe2` 56
Figure 8: UDP Scan of Firewall Interface `qfe3` 57
Figure 9: Nmap Scan of UDP Port 123 on 10.1.4.2 from 10.1.3.2 67
Figure 10: GIAC Network Diagram from Nick Read's Practical 70

Summary

This paper is submitted to fulfill a part of the requirements for obtaining the GIAC Certified Firewall Analyst (GCFW) certification. The security architecture for a fictitious company, GIAC Enterprises, will be presented. GIAC is engaged in on-line commerce, namely, the sale of fortune cookie fortunes, so the security of their network is highly important. The main features of GIAC's perimeter security will be discussed, with an emphasis on the configurations of the border router, the virtual private network (VPN) between GIAC and a partner company, and the primary corporate firewall. A tutorial will be presented explaining the procedures that should be followed to implement the firewall. The results of a security audit of the firewall will be presented and analyzed. Finally, an alternate version of the GIAC security architecture, as presented by a previous certification candidate, will be analyzed for possible security flaws.

1 Security Architecture

1.1 Introduction

GIAC Enterprises is a small company that has been recently established to compete in the business area of fund-raising for school organizations and other non-profit groups. GIAC contracts with its customers to deliver, for a fee, novelty fortune-cookie fortunes, which have been researched so as to relate to the customer's area of interest. For example, a high school French club would receive fortunes written in French, consisting of facts and trivia about France, as well as short sayings and proverbs that the French people might use in their daily lives. A municipal orchestra might receive a package of fortunes containing various quotes and tidbits of information about classical music and composers. On the reverse side of each printed fortune, the customer can put its name, logo, or web address. GIAC Enterprises does not provide the actual cookies. The customer arranges with a local bakery to produce the fortune cookies and load them with the fortunes. Finally, the customer sells the cookies to its supporters in the community.

1.2 Requirements

GIAC Enterprises' business is almost entirely Internet-based. In order to protect the company's investment, the security features of the company's network must be sufficiently strong to provide the three cornerstones of network security: integrity, confidentiality, and accessibility.

The security architecture must protect the integrity of the fortunes, which are GIAC's only product. If an attacker were to replace a fortune file with a file of his own creation, and that file were delivered to a customer, GIAC's reputation would be severely damaged and the company could be liable for damages. The integrity of the customers', suppliers', and partners' accounts with GIAC must also be protected, otherwise GIAC could lose revenue from not being able to bill customers properly or by overpaying suppliers and partners.

The confidentiality of information on the GIAC network must be protected to prevent third parties from accessing and abusing customers' confidential data such as credit card numbers. GIAC's proprietary data, in the form of fortune files, must also be protected to maintain GIAC's competitive advantage in its market. If a fortune file were stolen and published on the Internet, then GIAC would no longer be able to sell that file to a customer.

The GIAC network must be accessible at all times, since some of the suppliers and partners may be located overseas and will be making contact with the network at any hour of the day. In addition, if the network is unavailable when a customer wants to view the website or create an account, GIAC risks losing that customer's business. Therefore, the security architecture must prevent downtime as much as possible.

An additional consideration is that the security features must be provided at a cost low enough to allow GIAC to be profitable. GIAC is a very small company in a niche market where the potential earnings are marginal at best. Overhead expenses must be kept to a minimum.

All employees, suppliers, and partners are asked to read and agree to comply with GIAC's published security policy. This document provides rules for Internet usage and information about methods by which the network may be compromised. The document is intended to educate the network's users about secure network practices and inform them of their responsibilities to participate in keeping the network secure.

1.2.1 Customer Requirements

Customers must be able to use HTTP to access the GIAC web server to learn about the product offered by GIAC, to read basic information about the company, to find contact information, and to be instructed as to how to order a set of fortunes. The web server is located in the service network, which is the only part of the GIAC network that is available for public access. The customers may also browse sample fortunes by entering a topic of interest. If there are any fortunes on that topic already stored in the database, a small sample of them will be displayed.

The customers will use HTTPS for security while placing orders, tracking the status of those orders, and managing their user accounts. Each customer will set up an account with a username and password. After the account has been created, the customer can pay for orders with credit cards through a secure server. When their orders are ready, they will download their fortunes from the web server.

The customers also must be able to exchange e-mail messages with GIAC employees. They will use e-mail to specify the exact topic to which the fortunes should apply, ask questions and receive responses about the status of their order, and be notified when the order is ready to be downloaded.

1.2.2 Supplier Requirements

GIAC obtains its fortunes from a small group of individuals who contract to research topics for which fortunes are needed. These individuals correspond with GIAC employees through e-mail to receive assignments. They also have access through a secure server to submit invoices and to upload the fortunes that they have generated. The suppliers will use secure FTP over an SSL connection to transmit their completed assignments to the FTP server on the GIAC service network. GIAC will scan the files received for viruses, transfer the files to the internal network, and load the supplied fortunes into the database. The suppliers have usernames and passwords and will log into their accounts on the GIAC web server using HTTPS.

1.2.3 Partner Requirements

When a customer needs fortunes provided in a language other than English, GIAC calls on the services of its international business partner, a company that specializes in quick translations of documents. The partner company communicates with GIAC via a VPN network-to-network IPSec tunnel to retrieve the untranslated fortunes. Their access to the GIAC database server is read-only. They submit the translated fortunes back to GIAC using the same secure FTP connection as the suppliers.

1.2.4 On-Site Employee Requirements

GIAC's on-site staff must have access to a mail server. There are mail servers on both the service network and the internal network. E-mail to and from employees will be forwarded between the two mail servers. This policy reduces the number of connections made between the internal network and the Internet. The employees will use HTTP and HTTPS to access the web through a proxy server. They may initiate FTP file transfers to retrieve information from the Internet and may use FTP to transmit files to customers as needed. They will have access to the database server and other internal servers depending on their job descriptions. Employees must also have access to the service network to update the website and perform other maintenance.

GIAC security policy forbids the employees to use instant messaging and file sharing software due to the security problems posed by those applications.

1.2.5 Remote Employee Requirements

GIAC employees occasionally go on the road to make sales pitches to potential customers. These employees must have access to the GIAC internal network. They will use SSH software installed on their laptops to make an encrypted, dial-in connection to the router [1]. In order to reduce the risk that someone using a stolen laptop will penetrate the network, the remote employees will also use a smart key to provide two-token authentication. A smart key plugs into a USB port on the laptop [2]. When the employee logs in, he or she must provide a Personal Identification Number (PIN) in order to be authenticated. The smart key verifies the PIN and provides authentication credentials to the network. Using this technology, the remote employee is authenticated in two ways – by possession of the smart key and by knowledge of the correct PIN.

Once the remote employees have established a connection and are authenticated, they will connect to a telnet proxy server on the service network. The proxy server will relay the telnet traffic between the employee and the internal network. The company-owned laptops will also have personal firewall software and anti-virus software installed and updated regularly.

1.3 Architecture

The GIAC network architecture is shown in Figure 1. The design includes an external network between the border router and the firewall. A service network contains the servers that must be accessible to the public from the Internet. For enhanced security, the FTP server and web server are not on this service network. Instead, a Squid proxy server receives the incoming HTTP, HTTPS, and FTP requests and communicates with the web server and FTP server, which are on a separate subnet. That subnet will be referred to as the proxied network. There is no traffic allowed directly from the Internet to the proxied network. The service network will also proxy telnet traffic and relay e-mail to and from the internal network, which is where GIAC's management and employees workstations are. There will also be no direct traffic from the Internet to the internal network. All servers, including the firewall, are hardened Solaris platforms with up-to-date patches applied.

1.3.1 Network Addressing

GIAC Enterprises has obtained from an Internet Service Provider only as many IP addresses as it needs to do business. It does not need so many addresses that it must purchase an entire class C address space. For the purposes of this paper, GIAC's primary address will be represented as 192.168.100.1, an address from the private address space, rather than a true Internet address. Internally, the GIAC network uses addresses within the range of private addresses 10.1.0.0/16. The router performs dynamic NAT to enable connectivity from the internal network to the Internet.

The network belonging to GIAC's partner company will be represented by the IP address 192.168.200.1. The external subnet from their router to their firewall will be represented by the addresses 10.2.1.1 and 10.2.1.2.

The external network comprising the router and the firewall uses the network 10.1.1.0/28. The inside interface of the router is at address 10.1.1.1, while the outward facing interface of the firewall is at 10.1.1.2. The syslog server is statically routed to 10.1.1.3. A pool of addresses from 10.1.1.4 to 10.1.1.7 is used to support access for the remote employees.

The service network with the firewall, proxy server, DNS server, mail relay server, and NTP Server uses the subnet 10.1.2.0/29. The proxied network with the firewall, web server, and the FTP server uses subnet 10.1.3.0/29. Finally, the internal network uses addresses in the subnet 10.1.4.0/24.

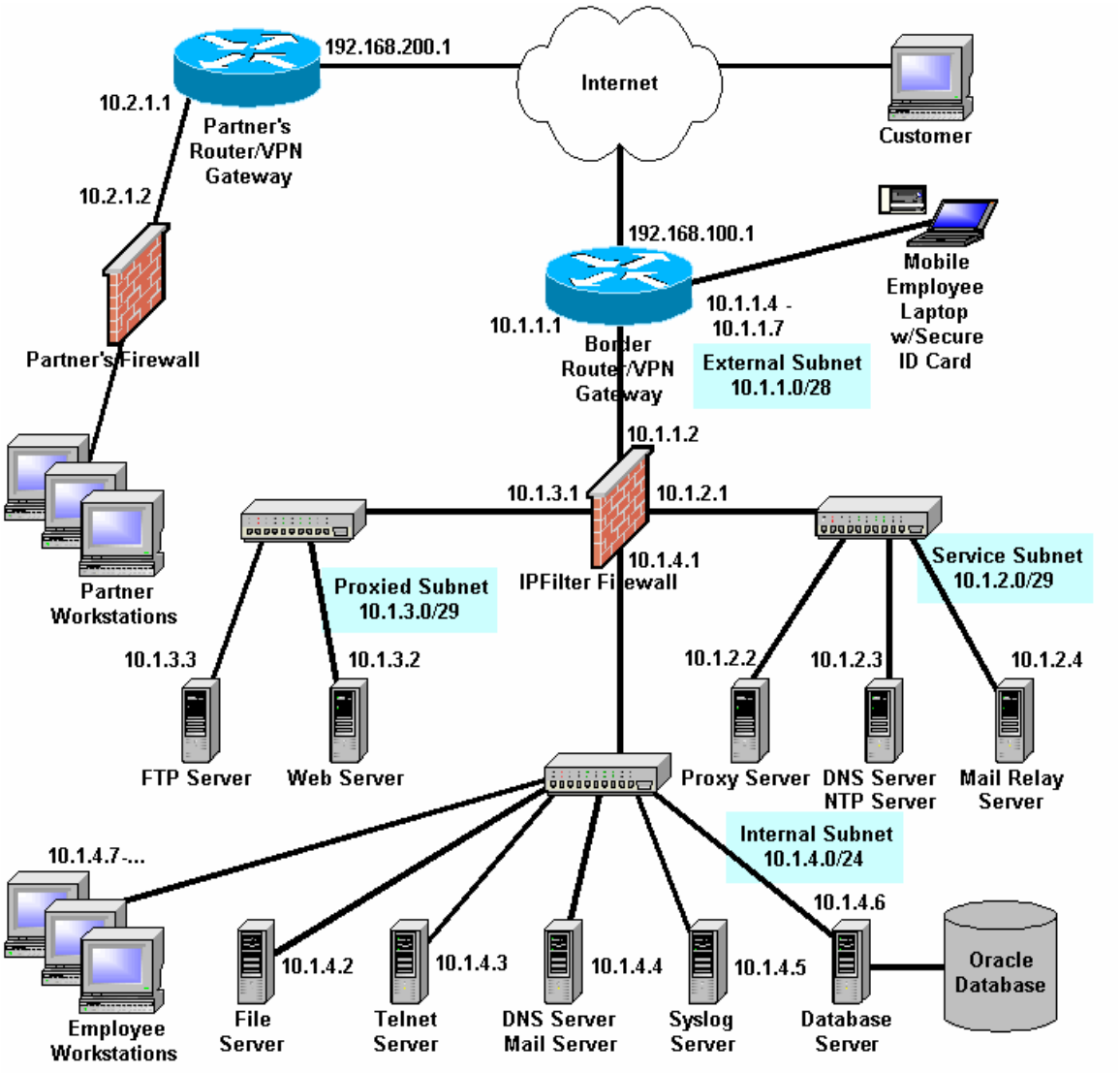

Figure 1: GIAC Network Diagram

1.3.2 Border Router & VPN

The GIAC network uses a CISCO 2691 as its border router [3]. The 2691 is a mid-range router that runs IOS release 12.2(8)T. VPN is also bundled with the router [4]. The bundled VPN eliminates the need for GIAC to purchase and integrate separate hardware or software. In addition, the VPN card offloads the encryption from the main CPU on the router. This choice of router and VPN, although somewhat expensive in its initial cost, should pay off in reliability and reduced maintenance costs as time passes. The router is configured with two fast ethernet interfaces and a 4-port Async/Sync module, which will be used by remote employees to dial in through their modems [5].

The router will be hardened as much as possible by removing unneeded services and by employing warning banners. It will transmit all of its log messages to the syslog server on the internal network.

1.3.3 Firewall

GIAC has chosen to install the IPFilter firewall software, version 3.4.31. This choice was made in part due to the savings that will result from using a free product, and in part due to previous familiarity with the software on the part of the IT staff. IPFilter, by Darren Reed, is a software package that provides a stateful, packet-filtering firewall as well as network address translation. The source code for IPFilter may be downloaded from http://coombs.anu.edu.au/~avalon/ [6]. The firewall is supported by a large user community and has an active mailing list. Appendix A contains additional background information on IPFilter.

A possible drawback to the use of IPFilter is that it is a packet filter only, i.e., it does not supply any proxying capability. A proxying firewall can provide a stronger level of security than a packet filter, in two ways. First, the firewall can inspect the contents of each packet and ensure that those contents are appropriate to the service being requested. Second, the proxy takes over the task of forwarding packets from the incoming interface to the outgoing interface on the firewall. IP forwarding at the kernel level can be turned off. This is not possible for a packet filter. If the packet filter firewall fails, the operating system may continue to forward packets, that is, it may "fail open," which leaves the network exposed. The proxy firewall is more likely to "fail closed," since IP forwarding has been turned off. A disadvantage of the proxying firewalls is that they perform more slowly and consume more system resources, due to the amount of work that has to be done to check the contents of each packet.

Although IPFilter is a packet-filtering firewall, it does provide transparent proxy support, through the 'ipnat' program, and it can be combined with a separate proxy server [7]. The GIAC network will use a Squid proxy server and a telnet proxy server on the service network to provide additional protection to the web server, FTP server, and hosts on the internal network.

The firewall host is a Sun Ultra 60 running Solaris 8, with all patches applied as recommended by Sun [8]. Before IPFilter was installed, the host was hardened using YASSP, to disable unnecessary services [9]. The host is also protected by Tripwire to provide an additional layer of security [10].

2 Security Policy and Tutorial

2.1 Router and VPN Configuration

2.1.1 Introduction

The first line of defense for the GIAC network is the border router. The router acts as a basic packet filter, so it provides a minimum level of protection to the network behind it. Simple packet filtering alone is not sufficient to provide complete protection to the network, however, so a stateful packet-filtering firewall, coupled with a separate proxy server, is also used behind the router. The router configuration was performed with inputs from several sources. The basic configuration was created using Cisco's free tool, Cisco ConfigMaker v2.6 [11]. A diagram of this configuration is shown as Figure 2. No attempt was made to reproduce the entire GIAC network, only the parts of it that were needed to set up the router configuration. The VPN link to the partner is shown, the dial-in connection for the remote employees, and the link to the firewall. These pieces were configured to match the GIAC network and the IOS configuration was then stored as a text file.

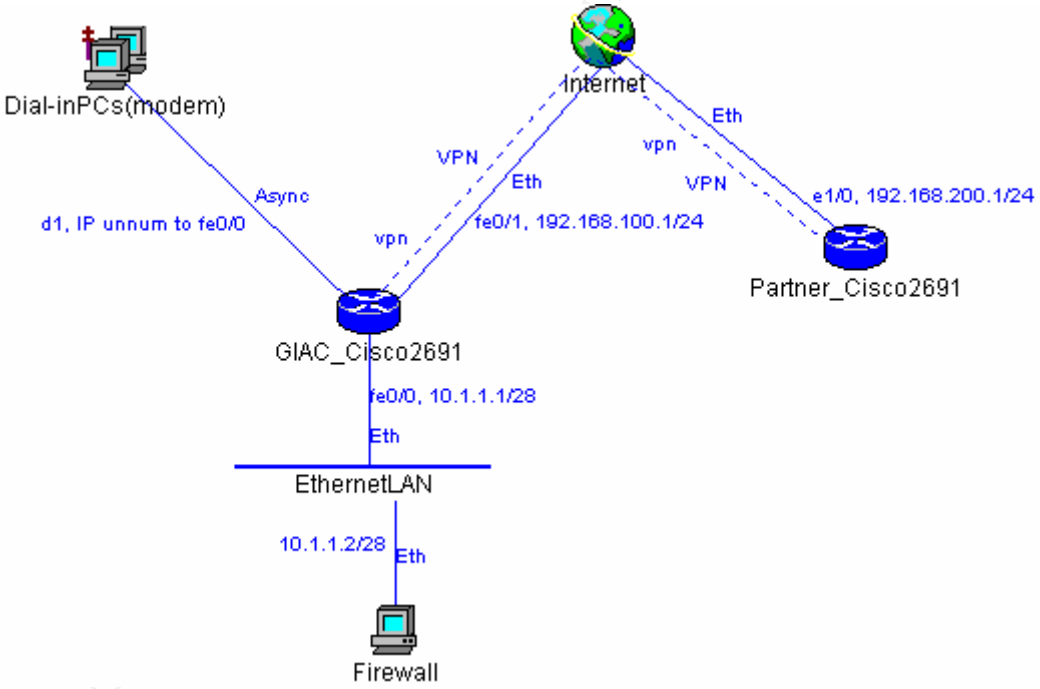

Figure 2: CISCO ConfigMaker Diagram

The IOS configuration created by the ConfigMaker tool does not contain many of the necessary commands to keep the router properly secured, so the configuration was edited. The additions to the basic configuration were based heavily on the O'Reilly book "Hardening Cisco Routers" by Thomas Akin [12], and on Rob Thomas's "Secure IOS Template" [13].

The Cisco 2691 integrates the router and the VPN gateway into one unit in the GIAC network design. The router configuration commands below include the commands to enable the VPN connection to GIAC's partner company.

2.1.2 Global Configuration Rules

The rules in this section apply to the router as a whole. The first section is a header generated by the Cisco ConfigMaker program. (Cisco ConfigMaker v. 2.6 does not include the 2691 router, which is relatively new. To generate the diagram, a Cisco 2621 was used in place of the 2691, and the configuration file was edited so the correct router type appears.)

```
! ************************************************************
! GIAC_Cisco2691.cfg - Cisco router configuration file
! Automatically created by Cisco ConfigMaker v2.6 Build 6
!   Tuesday, March 25, 2003, 07:51:46 PM
!
! Hostname: GIAC_Cisco2691
! Model: 2621
! ************************************************************
! Modified VPN and IPSec settings 3/25/03  8:31 pm
```

Enable encrypted passwords.

```
service password-encryption
```

Make sure the password is protected with an MD5 hash and that the older weakly encrypted passwords are not used.

```
enable secret 5 <PASSWORD>
no enable password
```

Harden the router by disabling services that are not needed, including the TCP and UDP "small services" (so-called because they use small port numbers) e.g., echo, chargen, daytime, and discard. Also, disable finger, bootp, and snmp.

```
no service tcp-small-servers
no service udp-small-servers
no service finger
no ip bootp server
no snmp-server
```

Other services should also be turned off. There is no need for remote configuration. The GIAC network does not use Dynamic Host Configuration Protocol so that service is disabled. PAD is an acronym for Packet Assembler/Disassembler, and is used for communication with remote devices, so it is turned off here as well [14]. Finally, disable the web browser user interface on the router.

```
no service config
no boot network
```

```
no service dhcp
no service pad
no ip http server
```

Disable Cisco Discovery Protocol (CDP) globally, since it can give away a lot of information to an attacker.

```
no cdp run
```

(If it is decided later to use CDP internally, the '`cdp run`' command would be used place of '`no cdp run`', and CDP would be then disabled on the Internet-facing interface, with '`no cdp enable`'.)

Turn off classless routing and source routing.

```
no ip classless
no ip source-route
```

Since Domain Name Service does not need to be offered by the router, the following commands are used.

```
no ip name-server
no ip domain-lookup
```

Enable services that will be used.

Cisco Express Forwarding (CEF) checks the routing information on packets that enter an interface to see if they should have come in on that interface. This service provides more efficient anti-spoofing than an Access Control List (ACL) [12].

```
ip cef
```

Turn on the Nagle congestion control algorithm, since remote employees will be using Telnet. "Without `service nagle` on a Cisco router, each character in a Telnet session is a separate CPU interrupt ... the Nagle service not only helps to optimize the Telnet session but also lessens the load on the router [15].”

```
service nagle
```

Turn on “keepalives” to keep telnet connections from becoming hung [16].

```
service tcp-keepalives-in
service tcp-keepalives-out
```

Turn on IP routing so the router will actually do its job. Allow the use of the "zero subnet.”

```
ip routing
ip subnet-zero
```

Set the static routes.

```
ip route 0.0.0.0 0.0.0.0 FastEthernet 0/1
ip route 10.1.4.0 255.255.255.248 FastEthernet 0/0 1 permanent
```

Define the address pool for the remote employees.

```
ip local pool GIAC_Cisco2691-Group-1 10.1.1.4 10.1.1.7
```

Turn debug logging on for the local machine and buffer 16 kilobytes of logs. "16k buffering is resonable [sic] for most modern routers/switches unless you have gigantic configs. It is usually best to have debugging logs on the local device. It helps when troubleshooting, but it's possible that you'd want a lower setting [17]."

```
logging buffered 16384 debugging
no logging console
```

Put timestamps in the log files.

```
service timestamps debug datetime msec show-timezone localtime
service timestamps log datetime msec show-timezone localtime
```

Use TACACS+ for Authentication, Authorization, and Accounting (AAA). By making the local account case-sensitive, brute-force attacks are less effective.

```
aaa new-model
aaa authentication login default group tacacs+ local-case
aaa authentication enable default group tacacs+ enable
aaa authorization commands 15 default group tacacs+ local
aaa accounting exec default stop-only group tacacs+
aaa accounting commands 15 default stop-only group tacacs+
aaa accounting network default stop-only group tacacs+
tacacs-server host 10.1.1.3
tacacs-server key <KEY>
```

If TACACS+ fails, the fallback is to use case-sensitive local authentication instead.

```
username <USERNAME> password <PASSWORD>
```

Configure TCP Intercept as a protection against TCP SYN flood attacks. The TCP Intercept software intercepts TCP connection requests and establishes a connection with the client, as if it were the requested server. If the connection is successful, the software establishes another connection with the server, as if it were the client. It then forwards packets from client to server and server to client throughout the lifetime of the connection. If the connection is not legitimate, the software aggressively times out half-open connections [18].

```
ip tcp intercept list 200
```

TCP intercept is configured to drop inactive connections after 60 seconds and to keep half-open connections for 10 seconds.

```
ip tcp intercept connection-timeout 60
ip tcp intercept watch-timeout 10
```

The “high-water mark” is the point at which TCP intercept will start aggressively timing out connections. The “low-water mark” is the level to which activity must fall before TCP intercept will drop out of aggressive mode. These values are set to 6000 and 1500 connection requests per minute, respectively.

```
ip tcp intercept one-minute high 6000
ip tcp intercept one-minute low 1500
```

Catch crash dumps and send them via FTP to the syslog server.

```
ip ftp username <USERNAME>
ip ftp password <PASSWORD>
exception core-file GIAC_Cisco2691-core
exception protocol ftp
exception dump 10.1.1.3
```

Synchronize the router's clocks with the NTP server on the service network. The key must match the NTP server.

```
ntp authentication-key 6767 md5 <SECRETKEY>
ntp authenticate
ntp update-calendar
ntp server 10.1.1.3
```

Standardize the timezone.

```
clock summer-time CDT recurring
```

Set up dynamic NAT

```
ip nat translation timeout 86400
ip nat translation tcp-timeout 86400
ip nat translation udp-timeout 300
ip nat translation dns-timeout 60
ip nat translation finrst-timeout 60
ip nat inside source list 1 interface FastEthernet 0/1 overload
```

Configure the Router Information Protocol [19].

```
router rip
 version 2
 network 10.1.1.1
```

```
 passive-interface FastEthernet 0/1
 no auto-summary
```

Set a banner to warn users that they are subject to monitoring.

```
banner motd #
Warning!!! This system is solely for the use of authorized users and
only for official purposes. Users must have express written
permission to access this system. You have no expectation of privacy
in its use. To ensure that the system is functioning properly,
individuals using this system are subject to having their activities
monitored and recorded at all times. Use of this system evidences an
express consent to such monitoring and agreement that if such
monitoring reveals evidence of possible abuse or criminal activity,
the results of such monitoring will be supplied to appropriate
officials.
#
```

To enable SSH on the router, set the router's host name and the GIAC domain name.

```
hostname GIAC_Cisco2691
ip domain-name giac.com
crypto key generate rsa
username <ADMIN> password <PASSWORD>
username <REMOTE1> password <PASSWORD>
username <REMOTE2> password <PASSWORD>
username <REMOTE3> password <PASSWORD>
ip ssh timeout 60
ip ssh authentication-retries 2
```

Configure the console to use the TACACS+ authentication set up earlier.

```
line console 0
 login authentication default
 transport input none
```

Disable the AUX port, which is not being used.

```
line aux 0
 login
 no password
 transport input none
 no exec
 exec-timeout 0 1
```

Provide terminal access to the router to an administrator, through Virtual TTY (VTY). The administrator's IP address is set in access list 500.

```
line vty 0 4
 login authentication default
 transport input ssh
 exec-timeout 5 0
```

```
 access-sclass 500 in
```

This configuration for the dial-in lines was generated by Cisco ConfigMaker [11]. The transport method was changed from “all” to “ssh” for greater security.

```
line 33 36
 autoselect ppp
 modem InOut
 transport input ssh
 stopbits 1
 speed 38400
 flowcontrol hardware
```

2.1.3 VPN & IPSec Configuration Rules

The following part of the router configuration file handles the setup for the VPN. The commands in this section were automatically generated by the Cisco ConfigMaker tool, and are part of the global router configuration.

The first two commands enable Internet Key Exchange [20] and define its identity for sharing keys.

```
crypto isakmp enable
crypto isakmp identity address
```

The next command block sets the policy for IKE. The default group number is 1, to designate a 768-bit encryption key. Here, the group number has been changed to 2, to enable a 1024-bit encryption key. This is made possible because the VPN module on the router handles the encryption work, freeing the unit's CPU to handle normal traffic. Otherwise, using the longer key might cause a performance decline on the part of the router. In addition, the default lifetime was 86400 seconds, or 1 day. This lifetime defines how often the security association must be updated. It has been set to 3600 seconds (1 hour) here for greater security. These commands must match the IKE policy that is set on the partner's router at the other end of the VPN tunnel.

```
crypto isakmp policy 1
  encryption des
  hash md5
  authentication pre-share
  group 2
  lifetime 3600
```

This statement sets the Internet address of GIAC's partner. It also sets the shared secret key that the two routers at either end of the VPN tunnel use to set up communications.

```
crypto isakmp key <KEY> address 192.168.200.1
```

Now that the characteristics of the VPN connection have been established, IPSec must be configured. Except for the '`set peer`' command, the next block of commands must be identical on the two routers. The first statement enables authentication headers (AH) and Encapsulated Security Payload (ESP). The next set of commands defines the rules for generating secret keys. A new key must be generated every 5 minutes (300 seconds). This ensures that if an attacker does crack the key, they can access only a minimal amount of data. The final command associates this IPSec configuration with the proper router interface.

```
crypto ipsec transform-set GCFWruleset ah-md5-hmac esp-des \
    esp-md5-hmac
crypto map GCFWmap 300 ipsec-isakmp
 match address 100
 set peer 192.168.200.1
 set transform-set GCFWruleset
 set security-association lifetime seconds 3600
 set security-association lifetime kilobytes 4608000
crypto map GCFWmap local-address FastEthernet 0/1
```

Access list 100 will be defined in paragraph 2.1.5.

2.1.4 Interface Configuration Rules

The following configuration commands should be issued for all interfaces of the router. These commands not only help secure the router, by disabling unused or dangerous services, but they also help GIAC to be a good network neighbor. By using these commands, the router won't forward redirected packets, won't send packets to unreachable addresses, won't propagate “smurf” attacks, won't proxy ARP requests, and won't give away the netmask. Finally, disabling source-routed packets helps to prevent address spoofing.

```
no ip redirects
no ip unreachables
no ip directed-broadcast
no ip proxy-arp
no ip mask-reply
no cdp enable
```

The following commands to configure the interfaces for the dial-in lines are copied almost verbatim from the configuration generated by Cisco ConfigMaker [11]. There are four serial lines, so GIAC can have up to four employees using remote access at one time, which should be enough for the immediate future. As the company grows, the Async/Sync module in the router could be replaced with one having more capacity.

```
interface Dialer 1
 description connected to Dial-inPCs(modem)
 ip unnumbered FastEthernet 0/0
 ip nat inside
 ip tcp header-compression passive
```

```
 encapsulation ppp
 dialer in-band
 dialer-group 1
 ppp authentication chap
 peer default ip address pool GIAC_Cisco2691-Group-1

interface Serial 1/0
 physical-layer async
 no shutdown
 description connected to Dial-inPCs(modem)
 ip unnumbered FastEthernet 0/0
 async mode dedicated
 dialer rotary-group 1
```

There are three more serial interfaces, configured exactly like the one above.

```
interface Serial 1/1
    ...
interface Serial 1/2
    ...
interface Serial 1/3
    ...
```

Configure the interface facing the Internet. The line beginning with '`crypto map`' refers back to the configuration for the VPN and IPSec. Access group 600 primarily contains the list of "bogon" addresses from which no traffic should be received.

```
interface FastEthernet 0/1
 no shutdown
 description Connected to Internet
 crypto map GCFWmap
 ip address 192.168.100.1 255.255.255.0
 ip nat outside
 no ip route-cache
 ip verify unicast reverse-path
 ip access-group 600 in
 keepalive 10
```

Rate limits are a defense against some types of flood attacks. The first statement rate-limits UDP traffic and the second one rate-limits ICMP traffic. When the bandwidth limit is exceeded by the given type of traffic, the excess traffic is dropped.

```
 rate-limit input access-group 300 2010000 250000 250000 \
     conform-action transmit exceed-action drop
 rate-limit input access-group 400 500000 62500 62500 \
     conform-action transmit exceed-action drop
```

Configure the inward facing interface.

```
interface FastEthernet 0/0
 no shutdown
```

```
description Connected to IPFilter firewall
ip address 10.1.1.1 255.255.255.0
ip nat inside
no keepalive
```

The Secure IOS Template [13] at this point sets commands which route packets from private network space to the null interface, i.e., the packets are "black holed." For example:

```
! ip route 192.0.2.0 255.255.255.0 null0
```

However, it is advised that the black hole routes not be used if TCP intercept is being used. "If you create black hole routes for all bogon ranges and point them to the null device, and if someone launches a SYN flood from a bogon range, then the router will send the SYN|ACK to the null device. The ... TCP Intercept queue begins to build quite quickly. [21]."

2.1.5 Access Control Lists

The first access control list corresponds to the dynamic NAT set up earlier.

```
no access-list 1
access-list 1 permit 10.1.1.0 0.0.0.240
```

An access control list must be defined to tell the router to encrypt traffic going out to the partner VPN network. The addresses used in this command are those of the internal networks on the inside of each company's router.

```
no access-list 100
access-list 100 permit 10.1.1.1 0.0.0.240 10.2.1.1 0.0.0.255
```

This ACL corresponds to TCP Intercept, which was set up earlier.

```
no access-list 200
access-list 200 permit tcp any 10.1.1.0 0.0.0.240
```

The following ACL's correspond to the rate limits previously set on UDP and ICMP.

```
no access-list 300
access-list 300 permit udp any any

no access-list 400
access-list 400 permit icmp any any
```

Access list 500 enables an administrator's IP address for VTY access to the router from the internal network.

```
no access-list 500
access-list 500 permit 10.1.4.10
```

Access list 600 is used to block packets from commonly spoofed IP addresses. From Rob Thomas's Secure IOS Template: "Deny any packets from the RFC 1918, IANA reserved, test, multicast as a source, and loopback netblocks [13]."

```
no access-list 600
access-list 600
access-list 600 deny ip 0.0.0.0 0.255.255.255 any log-input
access-list 600 deny ip 1.0.0.0 0.255.255.255 any log-input
access-list 600 deny ip 2.0.0.0 0.255.255.255 any log-input
...
...
...
access-list 600 deny ip 197.0.0.0 0.255.255.255 any log-input
access-list 600 deny ip 201.0.0.0 0.255.255.255 any log-input
access-list 600 deny ip 224.0.0.0 31.255.255.255 any log-input
```

Drop ICMP fragments.

```
access-list 600 deny icmp any any fragments log-input
```

Allow IP access to GIAC.

```
access-list 600 permit ip any 10.1.1.0 0.0.0.240
```

Drop everything else

```
access-list 600 deny ip any any log-input
end
```

2.2 Firewall Installation and Configuration Tutorial

2.2.1 Introduction

The focus of this tutorial is on the steps that must be followed to implement the firewall policy. However, unlike commercial firewalls or firewall appliances, the IPFilter firewall must be downloaded as source code, compiled, and installed by the user. Therefore, it seems appropriate here to include some information as to the procedures to be followed to get IPFilter working. Additional background information on IPFilter may be found in Appendix A.

2.2.2 Installation Procedures

2.2.2.1 Download

The source code for IPFilter is available to be downloaded via FTP from ftp://coombs.anu.edu.au/pub/net/ip-filter/ip-fil3.4.31.tar.gz or via HTTP from http://coombs.anu.edu.au/~avalon/ip-fil3.4.31.tar.gz. There are also mirror sites that can be used if the main sites are busy or offline. The list of mirror sites can be found at http://coombs.anu.edu.au/~avalon/ip-filter.html#Mirrors.

Create a directory for the download.

```
# mkdir /ipfilter
# cd /ipfilter
```

Download the source files via FTP from one of the above locations. There are unfortunately no safeguards to ensure the integrity of the source files, such as separately published MD5 hashes or PGP signatures, so it would be wise to inspect the source files after downloading them for any evidence that backdoor or Trojan code has been substituted for the original code.

After downloading the file, uncompress the archive and extract the source files from it.

```
# gunzip ip-fil3.4.31.tar.gz
# tar xvf ip-fil3.4.31.tar
```

The output of the 'tar' command is not shown here. It will create a subdirectory named `ip-fil3.4.31` and will place all of the archived source files in that subdirectory.

2.2.2.2 Compilation

Several files should be read before continuing to build and install IPFilter:

- `README`
- `IMPORTANT`
- `HISTORY`

- `COMPILE.Solaris2`
- `COMPILE.2.5`
- `INSTALL.Sol2`

In particular, the file `INSTALL.Sol2` gives instructions to be followed to build the IPFilter executable for Solaris 2.5 and up. The command to compile and link IPFilter is:

```
# make solaris
```

The make will fail with the following error message unless gcc version 3.0 or later is being used and is built to allow 64-bit objects:

```
Testing compiler gcc for 64 bit object file generation.
No 64 bit capable compiler was found
*** Error code 1
make: Fatal error: Command failed for target 'solaris'
```

If needed, a package file with gcc version 3.2.2 pre-compiled for Solaris 8 can be downloaded from ftp://ftp.sunfreeware.com/pub/freeware/sparc/8/gcc-3.2.2-sol8-sparc-local.gz [22].

2.2.2.3 Installation

If the compilation of IPFilter from source files is successful, the next step is to build a package file.

```
# cd SunOS5
# make package
```

The documentation with IPFilter implies that the 'make package' command will both make the package and do the 'pkgadd' as well. In practice, this does not seem to be the case. The following steps are necessary:

```
# cd sparc-5.8
# cp ipf.pkg /var/spool/pkg
# cd /var/spool/pkg
# pkgadd -d ipf.pkg
```

Answer "all" when asked which packages to install, and "y" when asked if certain scripts should be executed with super-user privilege. The following message should appear after several dozen lines of output.

```
Installation of <ipf> was successful.
```

At this point, the machine must be rebooted to load IPFilter into the kernel.

```
# reboot
```

Following the reboot, IPFilter will be running but will not be protecting the system because no rules have yet been established. In addition, it is necessary at this point to check that the host will route packets from one interface to another.

```
# ndd -get /dev/ip ip_forwarding
```

If this command returns a zero, then “IP forwarding” must be turned on.

```
# ndd -set /dev/ip ip_forwarding 1
```

However, this change will not be permanent, and the setting may revert to its original value the next time the host is rebooted. The setting can be made permanent by writing and installing an 'ipforward' shell script. Instructions on how to do so can be found at unixcircle.com [23].

2.2.3 Configuring and Using IPFilter

2.2.3.1 Creating Rule Sets

After the initial install, IPFilter has an empty rule set. To install filtering rules, a file must be edited and the rules created. (There is a Linux front end that supports IPFilter [24], but no such GUI-based tool is available for Solaris.) The filename can be anything, but it is common practice to place the rules in `/etc/ipf.rules`. The following is an example of how to create and install a rule set, using two very basic rules.

```
# vi /etc/ipf.rules
```

Type in these two lines:

```
pass in all
pass out all
```

Save the file and exit ‘vi’. (These rules essentially do nothing – they let all packets through in either direction on every interface.)

Now issue the following command:

```
# ipf -Fa -f /etc/ipf.rules
```

This command writes the rule set from `/etc/ipf.rules` into the IPFilter active rule set in memory. IPFilter will then begin to apply these rules to all packets. (The example rule set used here does not result in any packets being blocked, of course.)

2.2.3.2 Rule Syntax

If the firewall is to perform as useful function, more meaningful rules than those above must be designed. The syntax of rules for IPFilter is quite detailed, but

straightforward. The discussion that follows is derived primarily from the descriptions of filter rules found in “IPFilter Based Firewalls HOWTO [25]” and “IP Filter Examples [7].”

Each rule begins with permission either granted or denied, i.e., “pass,” or “block.” Following the permission is a set of keywords that are used to determine if a packet matches the rule, or to specify an additional action to perform when a packet does match. It is perhaps easiest to explain the keywords in reference to an example:

```
block return-rst in log quick on <interface> proto <protocol> \
    from <source-address> to <destination-address> \
    port = <port-number> flags <flag-list> keep state keep frags \
    group <group-number>
```

The '`return-rst`' keyword applies only to blocked packets, and only to the TCP protocol. It is optional. If it is present in a rule, and a packet is blocked by that rule, then a TCP reset will be returned to the source address. The purpose is to mask the presence of the firewall by making the port appear to be closed rather than filtered. The equivalent keyword for UDP ports is '`return-icmp-as-dest (port unr)`.'

The '`in`' keyword defines a direction relative to the firewall. Its alternate is '`out`', and either one or the other of the two is required. Packets may pass through an interface in either direction, and this pair of keywords allows the firewall designer to fine-tune the behavior of the firewall. One set of rules can be applied to incoming packets and another to outgoing packets, on the same interface. When the firewall has multiple interfaces (which is the usual case), each packet is first compared to the '`in`' rules for the interface through which it enters the firewall, and is next compared to the '`out`' rules for the interface through which it leaves. A packet can be blocked by either set of rules, and care must be taken to block it at the appropriate place, or the firewall will not function as it should.

The '`log`' keyword is optional. It enables logging, or more accurately, writes the packet to `/dev/ipl`. To actually turn on logging, the ‘ipmon‘ utility should be running.

The ‘`quick`’ keyword has a unique role, which will be explained more completely in paragraph 2.2.3.4. The key point is that if a packet matches a rule, and the ‘`quick`’ keyword is present, then the packet will be passed or blocked immediately and no further rules will be checked.

‘`On <interface>`’ is used to specify the interface to which this rule applies. It is optional. If it is not present, then the rule applies to packets on any interface.

The '`proto <protocol>`' keyword defines the Internet protocol to which this rule applies. The protocol can be any from the list found in `/etc/protocols`. The protocol number can be used instead of the acronym, such as '`proto 6`' for TCP.

The '`from <source-address>`' and '`to <destination-address>`' keywords refer to the source and destination IP addresses on the packet. The word 'any' may be used instead of a specific address. It is also possible to negate the address with an exclamation mark. The expression '`from !10.1.1.0/28`' will match all packets that have a source address not in the 10.1.1.0/28 subnet. The addresses can be specified in several forms, which are illustrated in the following list.

- as a hostname: '`to localhost`'
- as an IP address: '`to 10.1.1.2`'
- as a subnet: '`to 10.1.1.0/28`'

If a netmask is present, it can be specified with a slash, as shown above, or with the keyword '`mask`.' The netmask can be expressed in CIDR notation, in dotted-decimal format, or in hexadecimal.

If the rule specifies TCP or UDP, a particular port can be singled out with the '`port = <port-number>`' keyword. This feature enables the firewall designer to block specific services while permitting others. A range of ports can also be specified, using the notation "><", as in '`port 1 >< 1024`', which will match port 1, port 1024, and all ports in between. The designer can also use ">", "<", "!=", ">=", or "<=," in place of the equal sign.

For the ICMP protocol, the equivalent of the port keyword is the '`icmp-type`' keyword, which specifies echo requests, echo replies, source quench, and so on.

The '`flags`' keyword is used to match against packets which have a specific TCP flag, or set of flags, set. A very common use of this keyword is to match on the TCP SYN flag, in conjunction with the '`keep state`' setting. The designer can then block all TCP packets that are not legitimate attempts to make connections, while passing in those that are the beginning of a connection, or are part of an established connection.

The '`keep state`' setting causes an entry to be made for the packet in the state table. Once a connection is established in the state table, additional packets associated with that connection are passed without further rules checking. This feature simplifies rule writing by quite a bit. A session established through the firewall involves packets moving both in and out on two interfaces. Without the ability to keep state, the designer would have to write four sets of rules to allow the session to occur. By using '`keep state`,' the firewall designer only needs to

check the original connection attempt against the firewall policy, so the only set of rules needed is that for the first inbound interface.

Packets may become fragmented as they move across a network. Without the '`keep frags`' keyword, fragments will be dropped. Using this keyword signals IPFilter to use a fragment cache, which will allow the fragmented packets to reach their destination, provided they otherwise match the rules. Related keywords not used in GIAC's firewall rules are '`with frag`' and '`with short`.' For more information, the IPFilter Examples web page should be referred to [7].

The '`group <group-number>`' keyword is associated with the '`head <group-number>`' keyword. These options allow the designer to organize the firewall rules, both for ease of understanding them, and to improve performance. A rule group is started by a rule that begins with the '`block`' keyword and ends with '`head <group-number>`.' In rules that use the '`head`' keyword, '`quick`' does not immediately interrupt rule processing for a packet that matches, as it would otherwise. Instead, rule processing for the packet is immediately transferred to the rule group specified, and the packet is only compared to those remaining rules. Packets that do not match the '`head`' rule skip over the rules in that group. Proper use of rule groups can greatly reduce the number of rules that have to be checked for a packet.

There are a few additional keywords that will not be used to configure the IPFilter firewall for the GIAC network, such as '`with options`' and '`fastroute`.' A description of the use of these keywords can be found in the source documents referred to above.

2.2.3.3 Active and Inactive Rule Sets

IPFilter actually uses two rule sets, one active, and the other inactive. By default, 'ipf' commands work on the active set. The 'ipf –s' command swaps the two sets. The following procedure provides a safeguard against mistakes while new rule sets are being developed:

Create a backup version of the existing `/etc/ipf.rules` file:

```
# cp /etc/ipf.rules /etc/ipf.rules.bak
```

Edit a new rule set and store it as `/etc/ipf.rules`. Merely overwriting the previous contents of `/etc/ipf.rules` does not cause the new rule set to be applied, because the rules have not yet been loaded into memory. Now, swap the active and inactive rule sets:

```
# ipf -s
```

The rules that have been in force are now in the inactive rule set. Use the following command to write the new rules from the file into the active rule set:

```
# ipf -Fa -f /etc/ipf.rules
```

Perform testing and/or auditing to determine if the firewall is behaving properly using the new rule set. If not, swap the old rule set back in:

```
# ipf -s
```

This swap sets the rules back to what they were prior to the change, which is a quick and easy way to undo a mistake.

2.2.3.4 Rule Set Behavior

It is essential to have a firm grasp on the behavior of the rule set. When packets are checked with IPFilter, by default they are checked against the entire rule set. Then, if the packet matched any rules, the permission granted or denied by the last rule that matched is applied. If no rules at all matched, the packet is passed. To some, this behavior means that IPFilter appears to work "backwards," because the expected behavior is that the permission will be applied based on the first rule that the packet matches [26].

The above behavior can, and often should be, overridden with the '`quick`' keyword. If a packet matches a rule containing 'quick,' then further rule checking is suspended and the packet is blocked or passed based on that rule. Using '`quick`' universally would result in the rules being applied in a more intuitive way, but there may be cases where the firewall designer wants the flexibility offered by his ability to choose between using and not using '`quick`.'

2.2.4 The 'ipfstat' Utility

The 'ipfstat' utility will be used during the audit of the GIAC firewall to demonstrate how IPFilter has responded to incoming packets. This utility is very useful. Used with no arguments, it returns a statistical summary of the traffic through IPFilter, including the number of incoming packets, the number of outgoing packets, the number of blocked packets, and so on.

Used with command line options, 'ipfstat' can give the following information:

- List the filter rules which apply to inbound packets.

  ```
  # ipfstat -i
  ```

- List the filter rules which apply to outbound packets.

  ```
  # ipfstat -o
  ```

- List the filter rules and show the number of hits against each rule.

```
# ipfstat [-ih|-oh]
```

- List the IPFilter state table.

```
# ipfstat -s
```

While an IPFilter is being tested or audit, using the –ih ad –oh options is very valuable to show which rules the packets are actually matching. If the firewall is not behaving as expected, the 'ipfstat' command can be very helpful in arriving at a diagnosis of the problem.

2.3 Firewall Configuration

2.3.1 Introduction

There are four interfaces on the GIAC firewall. The external, Internet-facing interface at address 10.1.1.1 is designated by `qfe0`. The service network at address 10.2.1.1 is designated by `qfe1`, and the proxied network at address 10.3.1.1 is designated by `qfe2`. Finally, the internal, private network at address 10.4.1.1 is designated by `qfe3`. In addition, the loopback interface is designated `lo0`.

2.3.2 IPFilter Rule Set

2.3.2.1 `qfe0` Interface – Inbound from the Internet

Traffic that arrives on this interface is coming in from the Internet or from GIAC's remote employees. Send these packets to rule group 101. (The '`head`' rule does not block packets even though the '`quick`' keyword is used. This syntax is how a rule group is started.) Packets that are not coming in on interface `qfe0` will skip over the rules in group 101.

```
# Inbound from the Internet
block in quick on qfe0 all head 101
```

Block and log all packets which claim to be coming from our internal networks but are arriving from the outside rather than the inside. These rules must appear before the rules below which drop (without logging) packets from private address space. Otherwise, the rule that blocks packets from 10.0.0.0/8 would prevent these rules from ever being applied.

```
# spoofed packets with our internal addresses
block in log quick on qfe0 from 10.1.2.0/29 to any group 101
block in log quick on qfe0 from 10.1.3.0/29 to any group 101
block in log quick on qfe0 from 10.1.4.0/24 to any group 101
```

Block and log anything coming in to our internal broadcast addresses.

```
# internal broadcast
block in log quick on qfe0 from any to 10.1.2.0/32 group 101
block in log quick on qfe0 from any to 10.1.2.8/32 group 101
block in log quick on qfe0 from any to 10.1.3.0/32 group 101
block in log quick on qfe0 from any to 10.1.3.8/32 group 101
block in log quick on qfe0 from any to 10.1.4.0/32 group 101
block in log quick on qfe0 from any to 10.1.4.255/32 group 101
```

Provide for the remote employees who are using telnet to get to the internal network. The telnet sessions will be proxied by a telnet proxy on the service network. Since these packets come from addresses in the 10.1.1.0/28 network, the rule passing them in must occur before the rule blocking general traffic from

10.0.0.0/8. The '`keep state`' keyword is used to allow the response packets back out without an additional rule on the outbound portion of the interface. Attempted telnet connections to anywhere else are dropped.

```
# Telnet
pass in quick on qfe0 proto tcp from 10.1.1.0/28 to 10.1.2.2 \
    port = 23 flags S keep state keep frags group 101
block return-rst in log quick on qfe0 proto tcp from any to \
    10.1.3.0/29 port = 23 group 101
block return-rst in log quick on qfe0 proto tcp from any to \
    10.1.4.0/24 port = 23 group 101
```

Block all packets from private network space, multicast space, and so on, without logging. The entire list of subnets that should be blocked here is not shown, for brevity. The rule blocking the 10.0.0.0/8 network is commented out to facilitate testing, because the scanning device used for the audit will use address 10.1.1.1.

```
# private address blocks
block in quick on qfe0 from 192.168.0.0/16 to any group 101
block in quick on qfe0 from 172.16.0.0/12 to any group 101
#block in quick on qfe0 from 10.0.0.0/8 to any group 101
#
# Additional special use and non-routable address space
# also included here [27][28]
```

Allow the router's log messages to go to the syslog server on the internal network, using the static NAT address that was set up for the server. This rule has to come before the two rules below (labeled with “protect” comments), which block everything else that tries to go directly to the proxied or internal networks.

```
# router syslog
pass in quick on qfe0 proto udp from 10.1.1.3 to 10.1.4.5 \
    port = 514 group 101
```

ICMP echo and echo-reply are very good candidates for being disallowed in the production system, but they are turned on here to make trouble-shooting easier while getting the network set up. (The echo rule has been moved farther down in the rule set.) Also, pass in destination unreachable and source-quench packets [29].

```
# ICMP echo reply
pass in quick on qfe0 proto icmp from any to any icmp-type 0 \
    group 101
# ICMP destination unreachable
pass in quick on qfe0 proto icmp from any to any icmp-type 3 \
    group 101
# ICMP source quench
pass in quick on qfe0 proto icmp from any to any icmp-type 4 \
    group 101
```

Block and log anything else heading straight from the Internet to the proxied network or the internal network. All traffic bound for the web server or FTP server should go to the Squid proxy server first. Telnet and SMTP traffic bound for the internal network should go to the telnet proxy and the mail relay server.

```
# protect proxied network
block in log quick on qfe0 from any to 10.1.3.0/29 group 101
# protect internal network
block in log quick on qfe0 from any to 10.1.4.0/29 group 101
```

By putting the echo request rule behind the rules above that protect the proxied network and the internal network, only the service network can be pinged from the Internet. All other ICMP is blocked and logged.

```
# ICMP echo request
pass in quick on qfe0 proto icmp from any to any icmp-type 8 \
    group 101
# block other ICMP
block in log quick on qfe0 proto icmp from any to any group 101
```

Block and log services that should not come in from the Internet, such as rlogin and rsh. Send back a reset rather than just dropping the packet silently. (The '`return-rst`' option only works for TCP, not for other protocols.) There is a much larger list of TCP and UDP ports that are candidates to be blocked at this point in the rule set, but they are not shown here for brevity. (See, for example, Appendix A – Common Vulnerable Ports in the SANS/FBI Top Twenty List [30].)

```
# rlogin
block return-rst in log quick on qfe0 proto tcp from any to any \
    port = 513 group 101
# rsh
block return-rst in log quick on qfe0 proto tcp from any to any \
    port = 514 group 101
# portmap
block return-rst in log quick on qfe0 proto tcp from any to any \
    port = 111 group 101
# NFS
block return-rst in log quick on qfe0 proto tcp from any to any \
    port = 2049 group 101
# X11
block return-rst in log quick on qfe0 proto tcp from any to any \
    port = 6000 group 101
```

Block and log incoming syslog requests. This will prevent an attacker from deliberately filling the syslog queue before starting his real attack. For UDP the firewall can return an ICMP error message. Using '`return-icmp-as-dest`' rather than '`return-icmp`' prevents the firewall from putting its own IP address on the return packet.

```
# syslog
block return-icmp-as-dest (port-unr) in log quick on qfe0 proto \
```

```
    udp from any to any port = 514 group 101
```

Block some additional UDP traffic

```
# portmap
block return-icmp-as-dest (port-unr) in log quick on qfe0 proto \
    udp from any to any port = 111 group 101
# NFS
block return-icmp-as-dest (port-unr) in log quick on qfe0 proto \
    udp from any to any port = 2049 group 101
```

Block all packets that claim to be from the loopback network.

```
# loopback
block in quick on qfe0 from 127.0.0.0/8 to any group 101
```

HTTP and HTTPS are passed through to the proxy server. The combination of '`keep state`' and '`flags S`' is an important security precaution. A TCP packet that is initiating the "three-way handshake" will be allowed through and a state table entry will be created. Thereafter, all packets that are part of that connection, in either direction, will be allowed to pass. A TCP packet with anything but the SYN flag set will be dropped unless it corresponds to a connection that is already in the state table. This combination of options prevents FIN and XMAS scans.

```
# HTTP
pass in quick on qfe0 proto tcp from any to 10.1.2.2 port = 80 \
    flags S keep state keep frags group 101
# HTTPS
pass in quick on qfe0 proto tcp from any to 10.1.2.2 port = 443 \
    flags S keep state keep frags group 101
```

To handle FTP requests, the following rules are included. The first two support active FTP. Passive FTP poses some problems for the IPFilter firewall [25]. Configuring the FTP server to use a limited range of ephemeral ports for its passive FTP sessions can solve that problem. Those ports then have to be opened through the firewall, as is shown here.

```
pass in quick on qfe0 proto tcp from any to 10.1.2.2 \
    port = 20 flags S keep state keep frags group 101
pass in quick on qfe0 proto tcp from any to 10.1.2.2 \
    port = 21 flags S keep state keep frags group 101
pass in quick on qfe0 proto tcp from any to 10.1.2.2 \
    port 50000 >< 51000 flags S keep state keep frags group 101
```

Inbound DNS and NTP traffic has to have been initiated by GIAC's DNS and NTP servers. The sessions will be in the state table due to the '`keep state`' keyword and no additional rules are needed here.

Enable the mail relay server to receive SMTP traffic.

```
# SMTP
pass in quick on qfe0 proto tcp from any to 10.1.2.4 port = 25 \
    flags S keep state keep frags group 101
```

The next two rules block and log any TCP and UDP traffic that has not been passed or explicitly blocked already. These rules cause returns to be sent that make the TCP and UDP ports appear to be closed. (Otherwise, a scanner may see that they are filtered.)

```
  block return-rst in log quick on qfe0 proto tcp from any to any
group 101
  block return-icmp-as-dest (port-unr) in log quick on qfe0 proto udp
from any to any group 101
```

The final rule of this group denies and logs all other traffic that has not been specifically permitted to pass.

```
block in log quick on qfe1 all group 101
```

2.3.2.2 `qfe0` Interface - Outbound to the Internet

Traffic outbound on this interface is headed for the Internet or back to one of GIAC's remote employees. Much of the traffic outbound on this interface will consist of HTTP, HTTPS, and FTP responses to requests from the Internet, and DNS and NTP requests from hosts on the service network. No specific rules are needed here to allow this traffic out through the firewall because entries were made in the state table when the packets arrived on an inbound firewall interface. As long as the packets belong to a session that is already in the state table, they will be allowed through.

```
block out quick on qfe0 all head 100
```

Block any traffic that appears to be addressed to our internal networks.

```
block out log quick on qfe0 from any to 10.1.2.0/29 group 100
block out log quick on qfe0 from any to 10.1.3.0/29 group 100
block out log quick on qfe0 from any to 10.1.4.0/24 group 100
```

Allow the limited set of ICMP traffic: echo, etc.

```
# echo request
pass out quick on qfe0 proto icmp from any to any icmp-type 8 \
    group 100
# echo reply
pass out quick on qfe0 proto icmp from any to any icmp-type 0 \
    group 100
# destination unreachable
pass out quick on qfe0 proto icmp from any to any icmp-type 3 \
    group 100
# source quench
```

```
pass out quick on qfe0 proto icmp from any to any icmp-type 4 \
    group 100
```

There is a known issue involving Solaris' handling of its TCP/IP stack, which prevents '`return-rst`' from working as it should [31]. To get around this problem, the following two rules are added to allow reset flags from the firewall itself back out to the Internet [32].

```
pass out quick on qfe0 proto tcp from 10.1.1.2 to any flags R \
    group 100
pass out quick on qfe0 proto tcp from 10.1.1.2 to any flags AR \
    group 100
```

The final rule of this group denies and logs all other traffic that has not been specifically permitted to pass.

```
block out log quick on qfe0 all group 100
```

2.3.2.3 `qfe1` Interface - Inbound from the Service Network

Traffic inbound on the `qfe1` interface is arriving at the firewall from the service network. The proxy server can initiate HTTP, HTTPS, and FTP sessions with the web server and the FTP server on the proxied network, or with the Internet on behalf of GIAC employees on the internal network. The sessions are initiated here and the '`keep state`' flag is set. Packets that are part of an established session will pass through the firewall without additional rules. However, some care is needed here. Since some of the packets can possibly be addressed to anywhere on the Internet, '`to any`' is needed in these rules. A problem arises because '`to any`' is too loose for the internal traffic, allowing HTTP to be addressed not only to the Internet but also to the FTP server or to the internal network. To remedy this, several specific versions of the rule must precede the more general version with '`to any`.' The first rule allows HTTP to the web server only from the proxy server. The second rule then denies HTTP from any other server on the proxied network. The third rule denies it from any server on the internal network. Finally, the fourth rule allows HTTP that is bound for the Internet.

```
block in quick on qfe1 all head 11
  # HTTP
  pass in quick on qfe1 proto tcp from 10.1.2.2 to 10.1.3.2 \
      port = 80 flags S keep state keep frags group 11
  block in log quick on qfe1 proto tcp from any to 10.1.3.0/29 \
      port = 80 flags S keep state keep frags group 11
  block in log quick on qfe1 proto tcp from any to 10.1.4.0/24 \
      port = 80 flags S keep state keep frags group 11
  pass in quick on qfe1 proto tcp from 10.1.2.2 to any port = 80 \
      flags S keep state keep frags group 11

  # HTTPS
  pass in quick on qfe1 proto tcp from 10.1.2.2 to 10.1.3.2 \
```

```
    port = 443 flags S keep state keep frags group 11
block in log quick on qfe1 proto tcp from any to 10.1.3.0/29 \
    port = 443 flags S keep state keep frags group 11
block in log quick on qfe1 proto tcp from any to 10.1.4.0/24 \
    port = 443 flags S keep state keep frags group 11
pass in quick on qfe1 proto tcp from 10.1.2.2 to any port = 443 \
    flags S keep state keep frags group 11

# FTP
pass in quick on qfe1 proto tcp from 10.1.2.2 to 10.1.3.3 \
    port = 20 keep state keep frags group 11
block in log quick on qfe1 proto tcp from any to 10.1.3.0/29 \
    port = 20 keep state keep frags group 11
block in log quick on qfe1 proto tcp from any to 10.1.4.0/24 \
    port = 20 keep state keep frags group 11
pass in quick on qfe1 proto tcp from 10.1.2.2 to any port = 20 \
    keep state keep frags group 11

# FTP-data
pass in quick on qfe1 proto tcp from 10.1.2.2 to 10.1.3.3 \
    port = 21 keep state keep frags group 11
block in log quick on qfe1 proto tcp from any to 10.1.3.0/29 \
    port = 21 keep state keep frags group 11
block in log quick on qfe1 proto tcp from any to 10.1.4.0/24 \
    port = 21 keep state keep frags group 11
pass in quick on qfe1 proto tcp from 10.1.2.2 to any port = 21 \
    keep state keep frags group 11
```

The DNS and NTP servers need to send out queries and get replies. These sessions also depend on the '`keep state`' flag.

```
pass in quick on qfe1 proto tcp/udp from 10.1.2.3 to any \
    port = 53 keep state group 11
pass in quick on qfe1 proto udp from 10.1.2.3 to any \
    port = 123 keep state group 11
```

Allow syslog messages bound for the log server on the internal network.

```
pass in quick on qfe1 proto udp from any to 10.1.4.5 \
    port = 514 group 11
```

Allow the telnet proxy to initiate sessions with the internal network or the proxied network on behalf of on-site or remote employees.

```
# Telnet
pass in quick on qfe1 proto tcp from 10.1.2.2 to 10.1.4.0/24 \
    port = 23 flags S keep state keep frags group 11
pass in quick on qfe1 proto tcp from 10.1.2.2 to 10.1.3.0/28 \
    port = 23 flags S keep state keep frags group 11
```

Allow SMTP traffic out from the mail relay server. SMTP traffic can be initiated either out to the Internet or back into the internal network.

```
# SMTP
pass in quick on qfe1 proto tcp from 10.1.2.4 to any port = 25 \
    flags S keep state keep frags group 11
```

Allow ICMP echo, echo replies, destination unreachable, and source quench messages through to the Internet.

```
# ICMP echo
pass in quick on qfe1 proto icmp from any to any icmp-type 8 \
    group 11
# ICMP echo reply
pass in quick on qfe1 proto icmp from any to any icmp-type 0 \
    group 11
# ICMP destination unreachable
pass in quick on qfe1 proto icmp from any to any icmp-type 3 \
    group 11
# ICMP source quench
pass in quick on qfe1 proto icmp from any to any icmp-type 4 \
    group 11
```

Allow any ICMP to the other two GIAC subnets.

```
# ICMP
pass in quick on qfe1 proto icmp from any to 10.1.3.0/29 group 11
pass in quick on qfe1 proto icmp from any to 10.1.4.0/24 group 11
```

The final rule of this group denies and logs all other traffic that has not been specifically permitted to pass.

```
block in log quick on qfe1 all group 11
```

2.3.2.4 `qfe1` Interface – Outbound to the Service Network

Traffic outbound on the `qfe1` interface is leaving the firewall to hosts on the service network. This traffic may include HTTP, HTTPS, and FTP requests from the Internet and responses from the web server and FTP server on the proxied network. All of this traffic however, will already correspond to entries in the state table that were created earlier. Therefore, there is no need for additional rules on this interface. Additional traffic will include DNS and NTP requests. This traffic also will match previously created state table entries.

```
block out quick on qfe1 all head 10
```

Allow ICMP echo, echo replies, destination unreachable, and source quench messages through from the Internet.

```
# ICMP echo request
pass out quick on qfe1 proto icmp from any to any icmp-type 8 \
    group 10
# ICMP echo reply
pass out quick on qfe1 proto icmp from any to any icmp-type 0 \
    group 10
```

```
# ICMP destination unreachable
pass out quick on qfe1 proto icmp from any to any icmp-type 3 \
    group 10
# ICMP source quench
pass out quick on qfe1 proto icmp from any to any icmp-type 4 \
    group 10
```

Allow ICMP from the other GIAC subnets.

```
# ICMP
pass out quick on qfe1 proto icmp from 10.1.3.0/29 to any group 10
pass out quick on qfe1 proto icmp from 10.1.4.0/24 to any group 10
```

The final rule of this group denies and logs all other traffic that has not been specifically permitted to pass.

```
block out log quick on qfe1 all group 10
```

2.3.2.5 `qfe2` Interface - Inbound from the Proxied Network

Traffic inbound on the `qfe2` interface is arriving at the firewall from the web server and FTP server on the proxied subnet. Traffic from these servers may include responses to telnet, rlogin, and rsh requests from the internal network. Specific rules are not needed here because '`keep state`' was set when the requests arrived inbound on interface `qfe3`. As long as the packets are part of a session in the state table, they will be allowed through.

```
block in quick on qfe2 all head 21
```

The servers on this subnet may send DNS and NTP queries to the service network.

```
# DNS
pass in quick on qfe2 proto tcp/udp from any to 10.1.2.3 \
    port = 53 keep state group 21
# NTP
pass in quick on qfe2 proto udp from any to 10.1.2.3 port = 123 \
    keep state group 21
```

Allow syslog messages bound for the log server on the internal network.

```
pass in quick on qfe2 proto udp from any to 10.1.4.5 \
    port = 514 group 21
```

Allow the ICMP echo, echo replies, destination unreachable, and source quench messages through.

```
# ICMP echo
pass in quick on qfe2 proto icmp from any to any icmp-type 8 \
    group 21
# ICMP echo reply
```

```
pass in quick on qfe2 proto icmp from any to any icmp-type 0 \
    group 21
# ICMP destination unreachable
pass in quick on qfe2 proto icmp from any to any icmp-type 3 \
    group 21
# ICMP source quench
pass in quick on qfe2 proto icmp from any to any icmp-type 4 \
    group 21
```

The rule set allows ICMP to the other two GIAC subnets.

```
# ICMP
pass in quick on qfe2 proto icmp from any to 10.1.2.0/29 group 21
pass in quick on qfe2 proto icmp from any to 10.1.4.0/24 group 21
```

The final rule of this group denies and logs all other traffic that has not been specifically permitted to pass.

```
block in log quick on qfe2 all group 21
```

2.3.2.6 `qfe2` Interface – Outbound to the Proxied Network

Traffic outbound on the `qfe2` interface is headed for the web server and FTP server on the proxied subnet. HTTP, HTTPS, and FTP packets will be outbound on interface `qfe2` to the web server and FTP server from the Squid proxy server. No rules are needed here because the sessions were loaded into the state table when the TCP SYN packets came inbound from the `qfe1` interface. The same reasoning applies to DNS and NTP packets that are responses to queries initiated by the proxied network, as well as telnet, rsh, and rlogin session initiated by employees on the internal network.

```
block out quick on qfe2 all head 20
```

Allow ICMP echo, echo replies, destination unreachable, and source quench messages through from the Internet.

```
# ICMP echo
pass out quick on qfe2 proto icmp from any to any icmp-type 8 \
    group 20
# ICMP echo reply
pass out quick on qfe2 proto icmp from any to any icmp-type 0 \
    group 20
# ICMP destination unreachable
pass out quick on qfe2 proto icmp from any to any icmp-type 3 \
    group 20
# ICMP source quench
pass out quick on qfe2 proto icmp from any to any icmp-type 4 \
    group 20
```

Allow ICMP from the other GIAC subnets, for maintenance.

```
# ICMP
```

```
pass out quick on qfe2 proto icmp from 10.1.2.0/29 to any group 20
pass out quick on qfe2 proto icmp from 10.1.4.0/24 to any group 20
```

The final rule of this group denies and logs all other traffic that has not been specifically permitted to pass.

```
block out log quick on qfe2 all group 20
```

2.3.2.7 `qfe3` Interface – Inbound from the Internal Network

Traffic inbound on the `qfe3` interface is arriving at the firewall from the internal network. This interface is not a big security concern since the packets are coming from the trusted network. Still, the interface should not be too loose, in order to be a good network neighbor, if nothing else. Allow GIAC employees to initiate HTTP, HTTPS, and FTP sessions with the proxy server. Allow telnet, rlogin, rsh, and ICMP in to either the service network or the proxied network, for maintenance.

```
block in quick on qfe3 all head 31
  # HTTP
  pass in quick on qfe3 proto tcp from any to 10.1.2.2 \
      port = 80 flags S keep state keep frags group 31
  # HTTPS
  pass in quick on qfe3 proto tcp from any to 10.1.2.2 \
      port = 443 flags S keep state keep frags group 31
  # FTP
  pass in quick on qfe3 proto tcp from any to 10.1.2.2 \
      port = 20 keep state group 31
  pass in quick on qfe3 proto tcp from any to 10.1.2.2 \
      port = 21 keep state group 31
```

Allow NTP and DNS requests in, if bound for the appropriate server on the service network. Use the '`keep state`' flag to add entries to the state table for these sessions.

```
  # DNS
  pass in quick on qfe3 proto tcp/udp from 10.1.4.4 to 10.1.2.3 \
      port = 53 keep state group 31
  # NTP
  pass in quick on qfe3 proto udp from any to 10.1.2.3 \
      port = 123 keep state group 31
```

Allow employees to initiate telnet, rlogin, rsh, and ICMP traffic from the internal network to the service network and the proxied network, for maintenance.

```
  # Telnet
  pass in quick on qfe3 proto tcp from any to 10.1.2.0/29 \
      port = 23 flags S keep state keep frags group 31
  pass in quick on qfe3 proto tcp from any to 10.1.3.0/29 \
      port = 23 flags S keep state keep frags group 31

  # rlogin
```

```
pass in quick on qfe3 proto tcp from any to 10.1.2.0/29 \
    port = 513 flags S keep state keep frags group 31
pass in quick on qfe3 proto tcp from any to 10.1.3.0/29 \
    port = 513 flags S keep state keep frags group 31
# rsh
pass in quick on qfe3 proto tcp from any to 10.1.2.0/29 \
    port = 514 flags S keep state keep frags group 31
pass in quick on qfe3 proto tcp from any to 10.1.3.0/29 \
    port = 514 flags S keep state keep frags group 31
```

Allow the limited set of ICMP traffic out to the Internet.

```
# echo request
pass in quick on qfe3 proto icmp from any to any icmp-type 8 \
    group 31
# echo reply
pass in quick on qfe3 proto icmp from any to any icmp-type 0 \
    group 31
# destination unreachable
pass in quick on qfe3 proto icmp from any to any icmp-type 3 \
    group 31
# source quench
pass in quick on qfe3 proto icmp from any to any icmp-type 4 \
    group 31
```

Allow ICMP from the other GIAC subnets, for maintenance.

```
# ICMP
pass in quick on qfe3 proto icmp from any to 10.1.2.0/29 group 31
pass in quick on qfe3 proto icmp from any to 10.1.3.0/29 group 31
```

Allow the mail server to pass outgoing mail to the mail relay server on the service network.

```
# SMTP
pass in quick on qfe3 proto tcp from 10.1.4.4 to 10.1.2.4 \
    port = 25 flags S keep state keep frags group 31
```

The final rule of this group blocks and logs all other traffic that has not been specifically permitted to pass.

```
block in log quick on qfe3 all group 31
```

2.3.2.8 `qfe3` Interface – Outbound to the Internal Network

Traffic outbound on the `qfe3` interface is going from the firewall to the internal network. Much of the possible traffic moving in this direction is covered by '`keep state`' on the previous inbound rules. For example, GIAC employees may initiate HTTP sessions. The HTTP response packets will go out through this interface to the internal network. However, no rule is needed to specifically let them through because they will match an entry in the state table, and will be allowed through on that basis.

```
block out quick on qfe3 all head 30
```

Allow log messages from the router and from hosts on the other two networks to the syslog server.

```
pass out quick on qfe3 proto udp from 10.1.1.3 to 10.1.4.5 \
    port = 514 group 30
pass out quick on qfe3 proto udp from 10.1.2.0/29 to 10.1.4.5 \
    port = 514 group 30
pass out quick on qfe3 proto udp from 10.1.3.0/29 to 10.1.4.5 \
    port = 514 group 30
```

Allow ICMP from the other GIAC subnets, for maintenance.

```
pass out quick on qfe3 proto icmp from 10.1.2.0/29 to any group 30
pass out quick on qfe3 proto icmp from 10.1.3.0/29 to any group 30
```

Allow the limited set of ICMP traffic out to the Internet.

```
pass out quick on qfe3 proto icmp from any to any \
    icmp-type 0 group 30
pass out quick on qfe3 proto icmp from any to any \
    icmp-type 3 group 30
pass out quick on qfe3 proto icmp from any to any \
    icmp-type 4 group 30
pass out quick on qfe3 proto icmp from any to any \
    icmp-type 8 group 30
```

The final rule of this group blocks and logs all other traffic that has not been specifically permitted to pass.

```
block out log quick on qfe3 all group 30
```

2.3.2.9 Additional Rules

Allow the loopback interface to run unrestricted.

```
pass in quick on lo0 all
pass out quick on lo0 all
```

The final rules in the firewall rule set block and log any traffic not yet handled by all the previous rules. Almost all packets should have matched a rule that had '`quick`' set, and will not fall through to the end of the file.

```
block in log all
block out log all
```

3 Verify the Firewall Policy

3.1 Audit Plan

3.1.1 Network Used for the Audit

A minimal version of the GIAC network will be used for the audit of the firewall. Each of the subnets is present, so each interface of the firewall can be exercised, but in each case, the subnet is a stub with only one host attached. The four interfaces of the firewall are all connected to different nodes of the switch. The four hosts are connected to the corresponding nodes, leading to the arrangement shown in Figure 3.

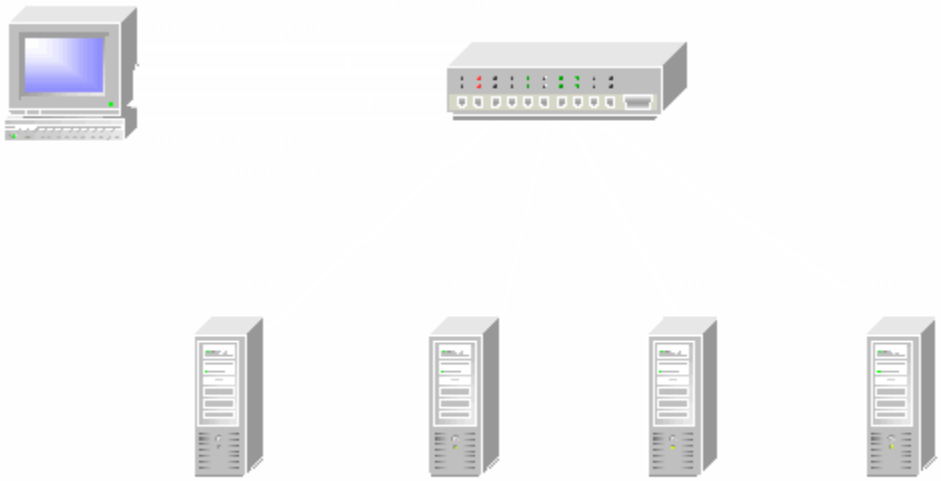

Figure 3: Network for Auditing the Firewall

3.1.2 Planned Tests

The network connectivity will be verified by using ICMP echo requests where the firewall rules allow. This action will also test the rules that forbid echo requests and replies across certain interfaces. The rules will be further tested by checking telnet and rlogin from each host to the others, to see if the firewall allows or denies the traffic in accordance with expectations.

Next, each interface of the firewall will be scanned with Nmap, using Nmap version 3.00 and NmapFE, a GUI front end for Nmap [33]. The scans will include a TCP probe of the full range of ports from 1 to 65535. More limited UDP scans will be done covering only ports 1-1024, due to the longer time involved in running UDP scans. At least one of the hosts used for the testing will be a Linux box, with Nmap installed.

Finally, Nmap will be used to make some specific scans directed across the firewall from one subnet to another. The reason for doing these scans is that the previous tests and scans do not fully exercise the rule set. Nmap will be used to generate packets directed to the ports that are used by HTTP, HTTPS, FTP, DNP, NTP, and SMTP. These scans do not constitute an audit of the hosts attached to the firewall, because, for this test, those hosts are not actually running the requested services. Rather, the goal of these tests is to show that the traffic will flow across the firewall as expected where the rule set allows it, and will be dropped otherwise.

3.1.3 Estimated Costs

If this audit is being performed after the GIAC network has already been put into production, a choice has to be made whether to take the network offline while the audit is being performed, or to perform the audit using equipment that is not a critical part of the network. If such equipment is not readily available, it may have to be purchased in order to perform the audit. (A side benefit would be that the equipment purchased for this reason would then be available to extend and expand the network, or as handy replacements if of the current network hosts failed.)

The audit itself should take no more than two full days to perform. If it can be arranged to perform the audit over a weekend, then there will be very little impact to the GIAC employees' ability to do their jobs. The labor costs for the audit will be about two man-weeks, possibly including overtime for a weekend, with a few days required to plan the audit, two days to set up the test network and perform the audit, and a few more days to analyze the results and write up a report and recommendations.

3.2 Audit Results and Analysis

3.2.1 ICMP Echo Requests

The first action performed after getting the network plugged in and the interfaces configured was to attempt to ping from the firewall to each host, and from each host back to the firewall and each other host. In the production environment, it would be wiser to disable some of this ICMP traffic, but for this audit it was useful to have it turned on, as an aid to correcting network and host configuration problems.

The results of this testing were that ICMP echo requests were sent and echo replies received for all attempts except two, which was as expected. In the firewall rules, echo requests coming from the Internet are dropped unless they are addressed to the service network. This policy provides a greater degree of protection to the internal network and the proxied network. The screen shot displayed as Figure 4 shows that the firewall at 10.1.1.2 and the service network host at 10.1.2.2 could be pinged from the host at 10.1.1.1, but that the pings

were not answered from the proxied network host at 10.1.3.2 and the internal network host at 10.1.4.2.

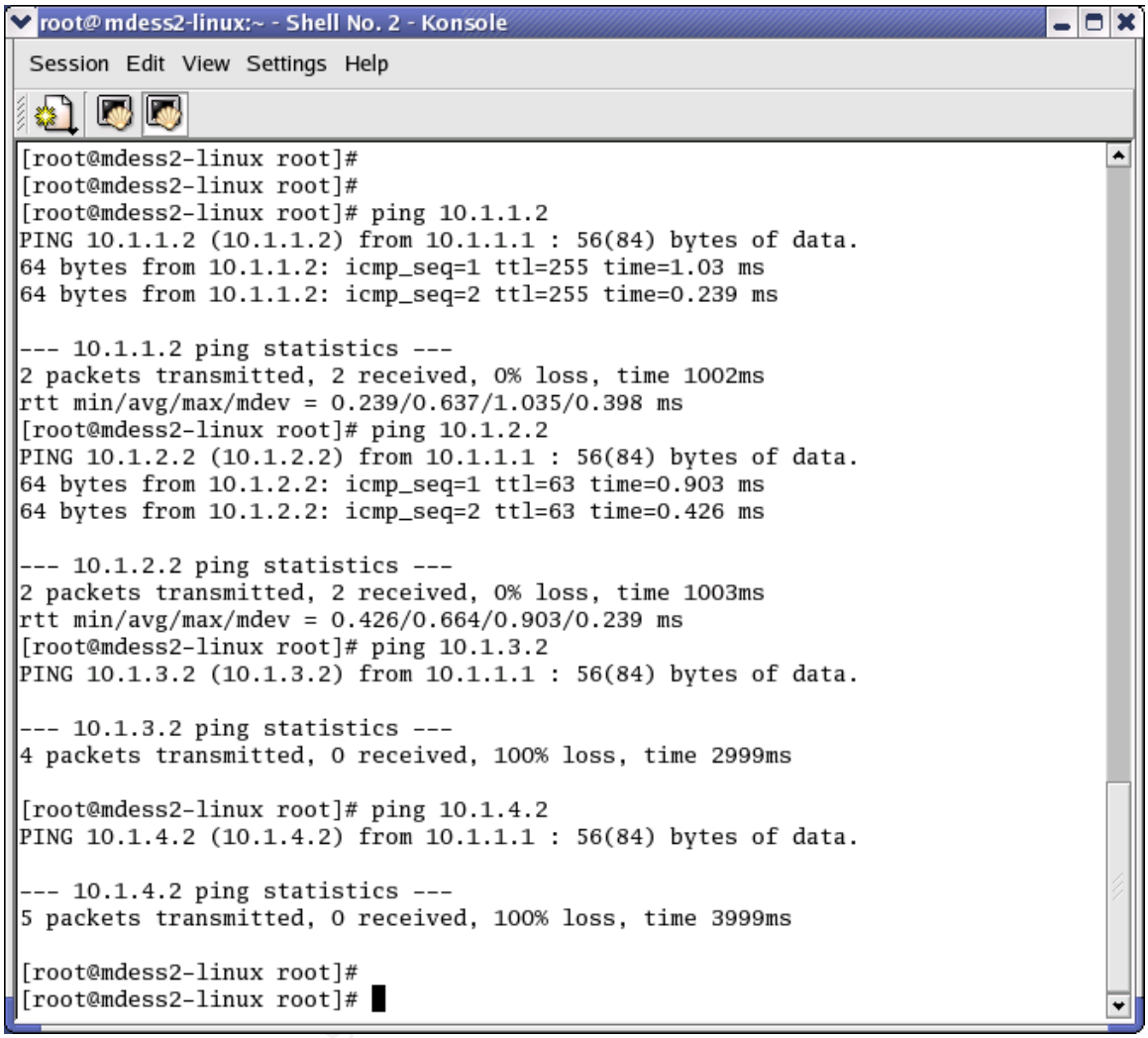

Figure 4: Echo Requests from 10.1.1.1

One change that should certainly be made before putting the firewall into the production system is to add rules to disallow echo replies from the firewall itself to the Internet. This step could be accomplished in several ways. One method is to modify the rule that reads:

```
# ICMP echo request
pass in quick on qfe0 proto icmp from any to any icmp-type 8 \
    group 101
```

The modified rule would be as follows.

```
# ICMP echo request
pass in quick on qfe0 proto icmp from any to 10.1.2.0/29 \
    icmp-type 8 group 101
```

This rule makes it more explicit that only the service network can be pinged from the Internet.

Another way to block ping replies from the firewall itself would be to add the following rule to group 101.

```
block in log quick on qfe0 proto icmp from any to 10.1.1.2 \
    icmp-type 8 group 101
```

A corresponding rule should be added to group 100.

```
block out log quick on qfe0 proto icmp from 10.1.1.2 to any \
    icmp-type 0 group 100
```

In general, the rule set in paragraph 2.3 should include more rules to protect the firewall itself. It is easy to fall into the trap of considering only the traffic that passes through the firewall, and to forget that the firewall itself may be left vulnerable. The only traffic allowed directly to the firewall, if any, should perhaps be an SSH session controlled by a systems administrator, in order to enable quick rule set changes and to perform other maintenance. If the firewall has its own monitor, such changes might be made directly on the machine itself. In that case, all traffic directly to the firewall could be denied.

3.2.2 Telnet and rlogin Connections

The second step performed to verify the firewall configuration was to attempt telnet and rlogin connections from each of the four subnets to the other three. Telnet is allowed from the remote employees' address pool to the telnet proxy server on the service network. The proxy must then be able to initiate telnet sessions to the internal network in order for the remote employees to get to their on-site workstations. Telnet is also allowed from the internal network to the proxy server. The proxy itself must be able to forward those requests over to the proxied network, in order to allow employee maintenance on the web server and FTP server.

The results of only one of these tests will be discussed in detail. For this test, an attempt was made to telnet to the service network from the external network. The output of snoop on the `qfe0` interface is shown below. This shows the telnet connection request coming in on the interface, and the response (in this case a reset packet) going back out. (This test was not intended to actually establish a telnet session, only to show that the traffic would correctly move across the firewall.)

```
# snoop -d qfe0
Using device /dev/qfe (promiscuous mode)
    10.1.1.1 -> 10.1.2.2     TELNET C port=39862
    10.1.2.2 -> 10.1.1.1     TELNET R port=39862
```

The next set of snoop output lines shows the same traffic from the point of view of the `qfe1` interface.

```
# snoop -d qfe1
Using device /dev/qfe (promiscuous mode)
    10.1.1.1 -> 10.1.2.2     TELNET C port=39862
    10.1.2.2 -> 10.1.1.1     TELNET R port=39862
```

Finally, the relevant output of tcpdump on the host at 10.1.2.2 is shown.

```
15:26:35.560079 10.1.1.1.29860 > 10.1.2.2.telnet: S
    2612224729:2613334729(0) win 5840 <mss 1460,sackOK,timestamp
    51367727 0,nop,wscale 0> (DF) [tos 0x10]
15:26:35.561023 10.1.2.2.telnet > 10.1.1.1.39860: R 0:0(0) ack
    2613334730 win 0 (DF) [tos 0x10]
```

This test established that the firewall is correctly allowing telnet traffic from the external network to the service network. Similar tests showed that telnet was also allowed as expected from 10.1.4.2 to 10.1.2.2, from 10.1.2.2 to 10.1.3.2, and from 10.1.2.2 to 10.1.3.2. All other telnet connection requests were denied at the firewall and were not passed on to the requested host.

Tests were also performed to show that the firewall was handling rlogin requests correctly. Shown below is an attempt to rlogin from the internal network to the service network. Under the current firewall rules, this traffic is allowed in order to provide maintenance capability to GIAC employees. The following blocks show output from snoop on the `qfe3` and `qfe1` firewall interfaces while the rlogin connection was attempted. As expected, the traffic is allowed across the firewall.

```
# snoop -d qfe3
Using device /dev/qfe (promiscuous mode)
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
^C#

# snoop -d qfe1
Using device /dev/qfe (promiscuous mode)
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
    10.1.4.2 -> 10.1.2.2     RLOGIN C port=1012
    10.1.2.2 -> 10.1.4.2     RLOGIN R port=1012
^C#
```

The following lines are output from tcpdump running on the host at 10.1.2.2. This output shows that the rlogin attempt was allowed through the firewall and arrived on this host, as expected. (This host is not configured to grant rlogin requests, so it sent back a reset.)

```
10:55:42.837699 10.1.4.2.1012 > 10.1.2.2.login: S
    124992565:124992565(0) win 24820 <nop,nop,sackOK,mss 1460> (DF)
10:55:42.837774 10.1.2.2.login > 10.1.4.2.1012: R 0:0(0) ack
    124992566 win 0 (DF)
```

An attempt was also made to rlogin from the service network to the internal network. According to the firewall rules, this connection should not be allowed, in order to protect the internal network from an attacker who has gained access to the service network, which is more exposed to the Internet. Here we see the traffic coming in on the `qfe1` interface, but nothing going out on the `qfe3` interface.

```
# snoop -d qfe1
Using device /dev/qfe (promiscuous mode)
    10.1.2.2 -> 10.1.4.2     RLOGIN C port=1023
    10.1.2.2 -> 10.1.4.2     RLOGIN C port=1023
    10.1.2.2 -> 10.1.4.2     RLOGIN C port=1023
    10.1.2.2 -> 10.1.4.2     RLOGIN C port=1023
    10.1.2.2 -> 10.1.4.2     RLOGIN C port=1023
    10.1.2.2 -> 10.1.4.2     RLOGIN C port=1023
^C#

# snoop -d qfe3
Using device /dev/qfe (promiscuous mode)
^C#
```

The next lines are output from snoop on the host at 10.1.4.2 during this rlogin attempt, showing that no packets were received.

```
# snoop
Using device /dev/hme (promiscuous mode)
^C#
```

The firewall blocked this rlogin attempt successfully.

Although the firewall worked as it should relative to telnet and rlogin, just having these functions enabled on the network is not a very safe choice. GIAC should disable telnet and all of the “r” commands and enable SSH where needed on the network [34]. The firewall rules would then need to be modified to handle the SSH traffic and to deny telnet, rlogin, rsh, and rcp traffic.

3.2.3 Firewall Scans with Nmap

All four interfaces of the firewall were scanned using Nmap. The first scan performed in each case was a TCP connect scan over the full range of addresses from 1 to 65535. A second scan was performed using UDP packets. Due to the substantially longer running times for UDP scans, these scans were limited to ports in the range from 1 to 1024.

3.2.3.1 Scan of the `qfe0` Interface

The `qfe0` interface of the firewall is configured to IP address 10.1.1.2 and is attached to a host configured to 10.1.1.1. The results of the TCP and UDP scans from 10.1.1.1 to 10.1.1.2 are shown as Nmap screen captures in Figure 5 and Figure 6.

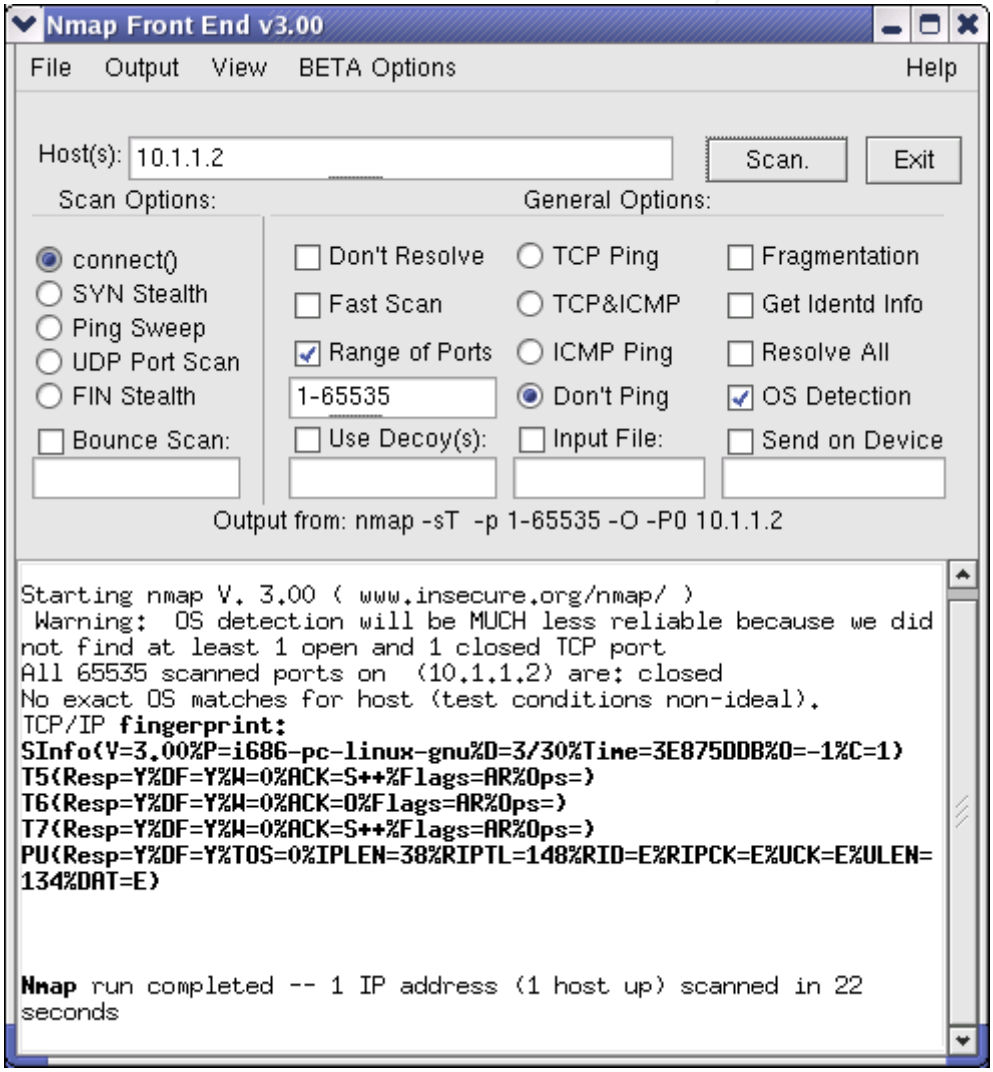

Figure 5: TCP Scan of `qfe0` Interface

The Nmap output shows that the IPFilter performed very well, at least against the simple scans used for this testing. Nmap reported all ports to be closed, and did not get any meaningful information about the operating system of the firewall host. There are two reasons for this good performance. First, the firewall host has been hardened with YASSP, so unneeded listening ports are not present. Second, the firewall rule set drops all TCP and UDP traffic that is not specifically addressed to one of the subnets beyond the firewall. All of the packets used for the scan were blocked, with a reset packet sent back to the scanning host.

Output from ipfstat following the scan, is shown below. Skipped lines (denoted by an ellipsis) matched no packets.

```
# ipfstat -ih | grep qfe0
65550 block in quick on qfe0 from any to any head 101
0 block in log quick on qfe0 from 10.1.2.0/29 to any group 101
...
0 block in log quick on qfe0 proto icmp from any to any group 101
1 block return-rst in log quick on qfe0 proto tcp from any to any
port = 513 group 101
1 block return-rst in log quick on qfe0 proto tcp from any to any
port = 514 group 101
1 block return-rst in log quick on qfe0 proto tcp from any to any
port = 111 group 101
1 block return-rst in log quick on qfe0 proto tcp from any to any
port = 2049 group 101
1 block return-rst in log quick on qfe0 proto tcp from any to any
port = 6000 group 101
0 block return-icmp-as-dest (port-unr) in log quick on qfe0 proto udp
from any to any port = 514 group 101
...
0 pass in quick on qfe0 proto tcp from any to 10.1.2.4/32 port = 25
flags S/FSRPAU keep state keep frags group 101
65541 block return-rst in log quick on qfe0 proto tcp from any to any
group 101
5 block return-icmp-as-dest (port-unr) in log quick on qfe0 proto udp
from any to any group 101
0 block in log quick on qfe0 from any to any group 101
```

The corresponding output for the "out" direction is shown below.

```
# ipfstat -oh | grep qfe0
65550 block out quick on qfe0 from any to any head 100
0 block out log quick on qfe0 from any to 10.1.2.0/29 group 100
0 block out log quick on qfe0 from any to 10.1.3.0/29 group 100
0 block out log quick on qfe0 from any to 10.1.4.0/24 group 100
0 pass out quick on qfe0 proto icmp from any to any icmp-type echorep
group 100
5 pass out quick on qfe0 proto icmp from any to any icmp-type unreach
group 100
0 pass out quick on qfe0 proto icmp from any to any icmp-type squench
group 100
0 pass out quick on qfe0 proto icmp from any to any icmp-type echo
group 100
```

```
0 pass out quick on qfe0 proto tcp from 10.1.1.2/32 to any flags
R/FSRPAUC group 100
65546 pass out quick on qfe0 proto tcp from 10.1.1.2/32 to any flags
RA/FSRPAUC group 100
0 block out log quick on qfe0 from any to any group 100
```

The decision to use the '`return-rst`' option is an important one. Without this option turned on, the packets are simply dropped. In that case, Nmap reports that the ports have been filtered, which would tell an attacker that the host being scanned was a firewall. With the '`return-rst`' option set, the attacker sees nothing but closed ports, which gives him no clues as to any further exploits to attempt.

The UDP scan of the `qfe0` interface gives similar results. All UDP ports that were included in the scan (1-1024) are reported by Nmap to be closed.

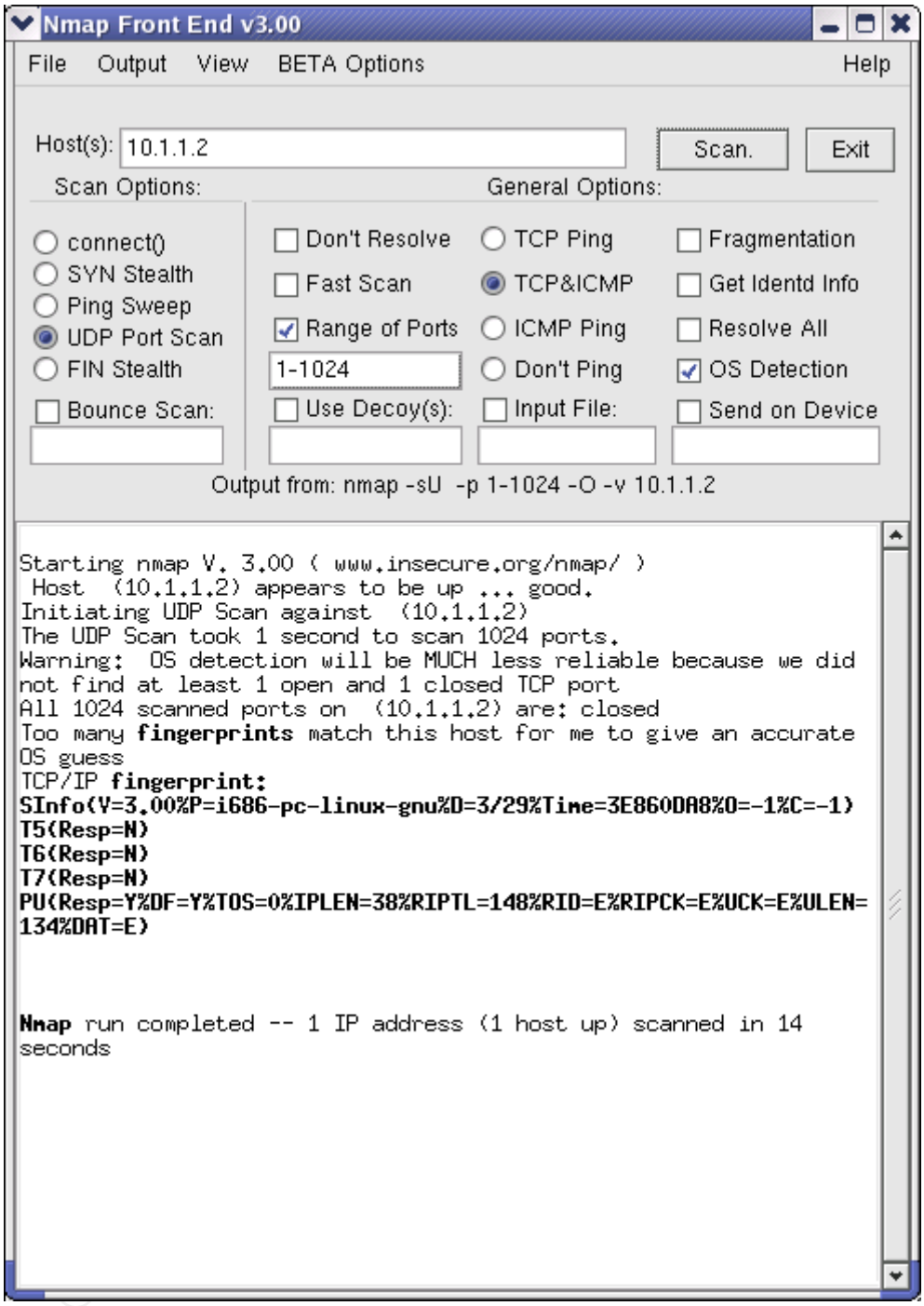

Figure 6: UDP Scan of `qfe0` Interface

3.2.3.2 Scans of the `qfe1`, `qfe2`, and `qfe3` Interfaces

The other three interfaces of the firewall were scanned in a similar manner to the scans of `qfe0` above. Interface `qfe1` was scanned from 10.1.2.2, interface `qfe2` was scanned from 10.1.3.2, and interface `qfe3` was scanned from 10.1.4.2. TCP and UDP scans were used. To save space, the results of all six

scans are not shown, but two representative scan results are shown as the Nmap screen captures in Figure 7 and Figure 8.

The TCP scans reported all ports as "filtered," except for the ports corresponding to traffic that is allowed to cross the interface. In those cases, the port is shown as closed. The unused ports are shown as filtered because IPFilter has not been configured to send back resets on these interfaces. Since GIAC owns the networks that are connected to these interfaces, it is not as important to return the resets as it is for the interface that is connected to the Internet. Nonetheless, GIAC should add the '`return-rst`' to the rules governing TCP traffic on these interfaces. Doing so would provide an additional layer of security. If an attacker succeeded in compromising a host on the service network, GIAC should make it difficult for him to penetrate further.

In every case, for the three interfaces `qfe1`, `qfe2`, and `qfe3`, the UDP scan reported all 1024 UDP ports closed.

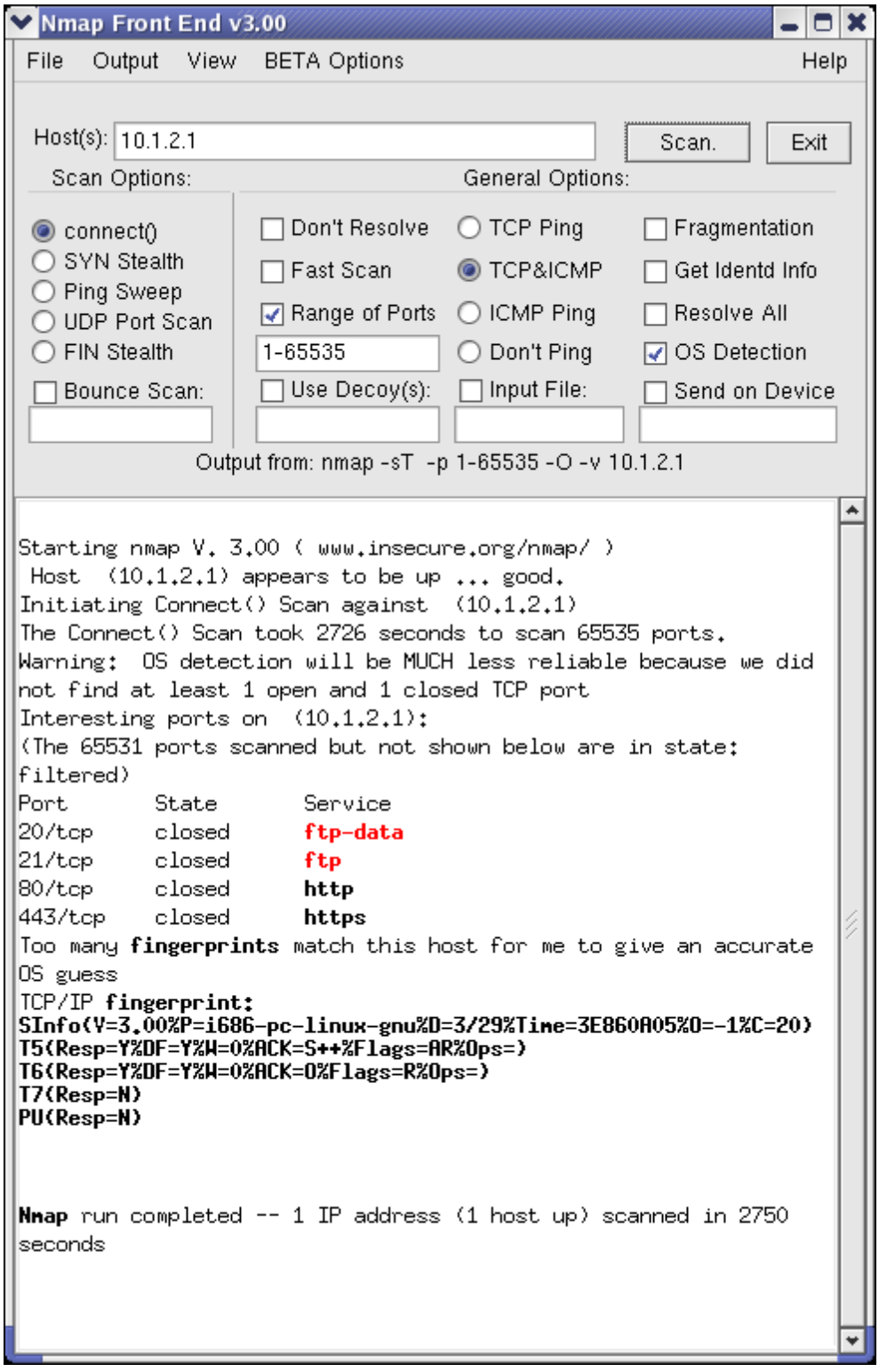

Figure 7: TCP Scan of Firewall Interface `qfe2`

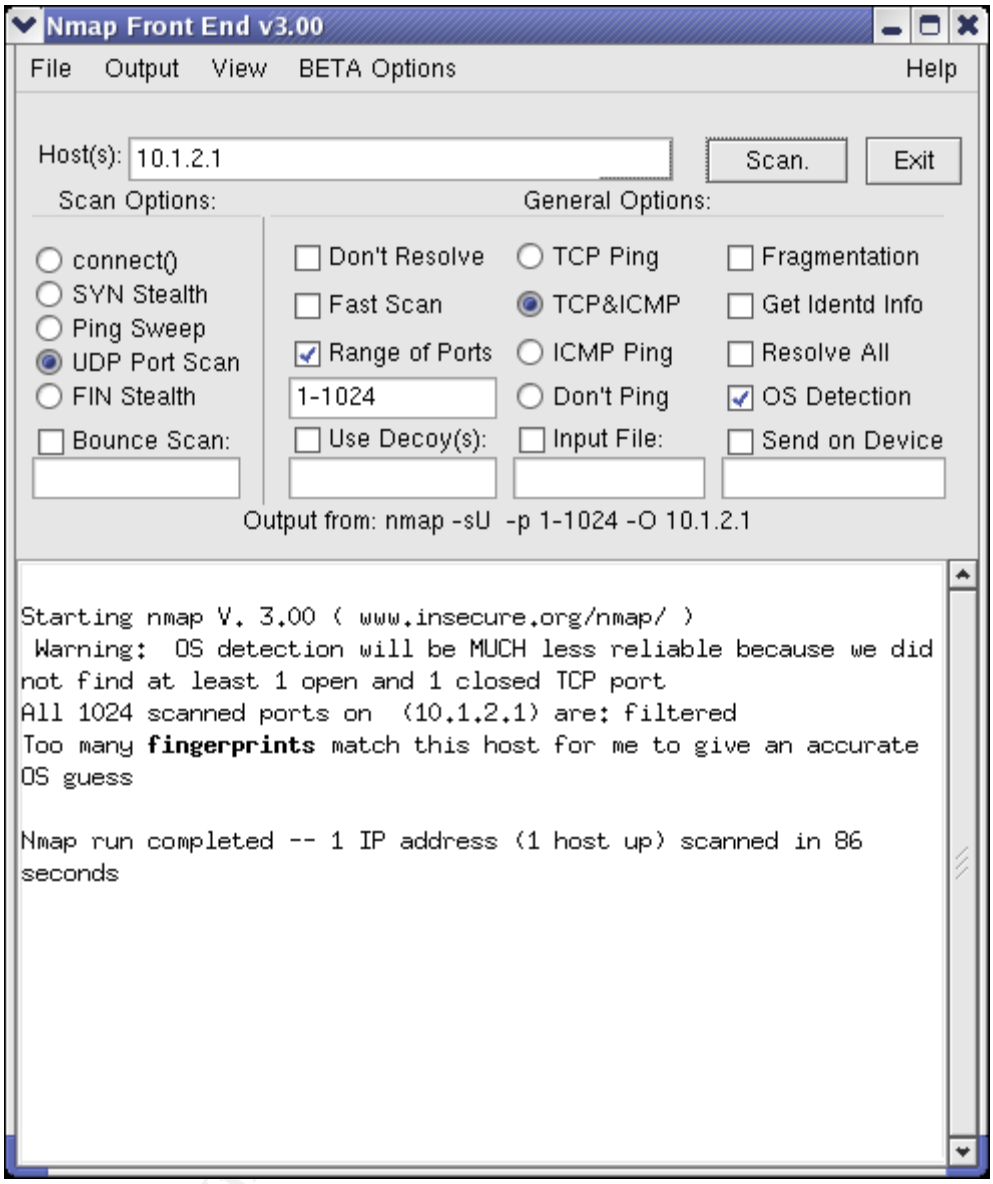

Figure 8: UDP Scan of Firewall Interface `qfe3`

3.2.4 Nmap Scans Across the Firewall

In order to exercise the firewall rules fully, it is not sufficient merely to scan the firewall interfaces. Such scans show that the firewall host itself is fairly secure, but do not show that the firewall is functional. A firewall which dropped every inbound packet would be secure ... and useless.

The goal of the following tests is to show that the firewall functions as designed. Nmap will be used to generate “mini-scans” using only the port numbers of the services that actually will be enabled on the GIAC network, including HTTP, HTTPS, FTP, DNS, NTP, and SMTP. The target of the scan will not be the firewall itself, but one or another of the hosts attached to the firewall.

Snoop and tcpdump will be used to verify that the traffic arrives at the firewall on the incoming interface. These tools will also be used to show that one of the two following events happens. If the firewall rule set allows the requested service to be accessed from the requesting host, then the traffic will be shown to leave the firewall on the appropriate interface and arrive at the requested host. If the firewall policy denies access to that service, then the tools will show that the packets have been dropped.

There is no intention to audit the hosts attached to the firewall. In most cases, for this test network, the hosts are not running the requested services. In these cases, the only return traffic is usually a TCP reset.

Not every possible scan will be reported here, for brevity. A set of three scans has been selected to illustrate the results of this part of the testing.

3.2.4.1 External Network to Service Network

From the external network (representing the Internet and GIAC's remote employees), several service requests are allowed to enter the service network. These requests include HTTP, HTTPS, FTP, and telnet. Nmap was used on the host at 10.1.1.1 and several scans were directed at the host at 10.1.2.2. For each of these scans, only a single TCP port was selected.

The snoop output on the `qfe0` interface shows that the packets arrived at the firewall and resets were sent back as replies. (The host at 10.1.2.2 is not running any of these services.)

```
# snoop -d qfe0
Using device /dev/qfe (promiscuous mode)
    10.1.1.1 -> 10.1.2.2     FTP-DATA C port=52845
    10.1.2.2 -> 10.1.1.1     FTP-DATA R port=52845
    10.1.1.1 -> 10.1.2.2     FTP C port=52846
    10.1.2.2 -> 10.1.1.1     FTP R port=52846
    10.1.1.1 -> 10.1.2.2     TELNET C port=52847
    10.1.2.2 -> 10.1.1.1     TELNET R port=52847
    10.1.1.1 -> 10.1.2.2     HTTP C port=52848
    10.1.2.2 -> 10.1.1.1     HTTP R port=52848
    10.1.1.1 -> 10.1.2.2     HTTPS C port=52849
    10.1.2.2 -> 10.1.1.1     HTTPS R port=52849
^C#
```

Snoop output on the `qfe1` interface shows that the firewall forwarded the service requests from the external network to the service network.

```
# snoop -d qfe1
Using device /dev/qfe (promiscuous mode)
    10.1.1.1 -> 10.1.2.2     FTP-DATA C port=52845
    10.1.2.2 -> 10.1.1.1     FTP-DATA R port=52845
    10.1.1.1 -> 10.1.2.2     FTP C port=52846
```

```
    10.1.2.2 -> 10.1.1.1     FTP R port=52846
    10.1.1.1 -> 10.1.2.2     TELNET C port=52847
    10.1.2.2 -> 10.1.1.1     TELNET R port=52847
    10.1.1.1 -> 10.1.2.2     HTTP C port=52848
    10.1.2.2 -> 10.1.1.1     HTTP R port=52848
    10.1.1.1 -> 10.1.2.2     HTTPS C port=52849
    10.1.2.2 -> 10.1.1.1     HTTPS R port=52849
^C#
```

Finally, the output from tcpdump shows the service requests arriving on host 10.1.2.2. The firewall performed as expected in allowing this specific network traffic.

```
17:36:27.102516 10.1.1.1.52845 > 10.1.2.2.ftp-data: S
    3199262647:3199262647(0) win 5840 <mss 1460,sackOK,timestamp
    99549487 0,nop,wscale 0> (DF)
17:36:27.102604 10.1.2.2.ftp-data > 10.1.1.1.52845: R 0:0(0) ack
    3199262648 win 0 (DF)
17:36:33.944762 10.1.1.1.52846 > 10.1.2.2.ftp: S
    3198963466:3198963466(0) win 5840 <mss 1460,sackOK,timestamp
    99552990 0,nop,wscale 0> (DF)
17:36:33.944830 10.1.2.2.ftp > 10.1.1.1.52846: R 0:0(0) ack
    3198963467 win 0 (DF)
17:36:43.553816 10.1.1.1.52847 > 10.1.2.2.telnet: S
    3206134808:3206134808(0) win 5840 <mss 1460,sackOK,timestamp
    99557910 0,nop,wscale 0> (DF)
17:36:43.553882 10.1.2.2.telnet > 10.1.1.1.52847: R 0:0(0) ack
    3206134809 win 0 (DF)
17:36:47.856721 10.1.1.1.52848 > 10.1.2.2.http: S
    3221231483:3221231483(0) win 5840 <mss 1460,sackOK,timestamp
    99560113 0,nop,wscale 0> (DF)
17:36:47.856781 10.1.2.2.http > 10.1.1.1.52848: R 0:0(0) ack
    3221231484 win 0 (DF)
17:36:52.689097 10.1.1.1.52849 > 10.1.2.2.https: S
    3215245800:3215245800(0) win 5840 <mss 1460,sackOK,timestamp
    99562588 0,nop,wscale 0> (DF)
17:36:52.689158 10.1.2.2.https > 10.1.1.1.52849: R 0:0(0) ack
    3215245801 win 0 (DF)
```

3.2.4.2 Service Network to Proxied Network

The service network may initiate connections with the proxied network to request HTTP, HTTPS, and FTP services. However, those services are running on two different servers, and the firewall rules are designed to only let packets through which are addressed to the correct server for the type of service requested. In addition, for maintenance purposes, telnet requests can be made to either server. To show that this portion of the firewall rule set is functioning correctly, the host on the proxied network will first be set to address 10.1.3.1 (corresponding to the HTTP server) and a series of packets will be sent from the service network using Nmap. The packets will include TCP ports 80, 443, 20, 21, 23, and 25. In addition, UDP ports 53 and 123 will be targeted. Then the host on the proxied network will be reconfigured to address 10.1.3.3 (corresponding to the FTP server), and the same series of packets will be sent. For these scans,

Nmap is configured to include an ICMP echo request along with the TCP or UDP packet.

The following block of output is from snoop running on the `qfe1` interface of the firewall. This output shows packets inbound to the firewall from the service network. It also shows the traffic from 10.1.3.2 back to the host on 10.1.2.2. (Blank fill was edited from these lines to prevent wrap onto the next line.) It can be seen that Nmap was used to generate HTTP, HTTPS, FTP, FTP-data, telnet, DNS, and NTP traffic.

```
# snoop -d qfe1
Using device /dev/qfe (promiscuous mode)
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 7194 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=39390
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 7194 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=52949
10.1.3.2 -> 10.1.2.2 HTTP R port=52949
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 43137 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=63043
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 43137 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTPS C port=52950
10.1.3.2 -> 10.1.2.2 HTTPS R port=52950
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 34731 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=44045
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 34731 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 FTP-DATA C port=52951
10.1.2.2 -> 10.1.3.2 FTP-DATA C port=52952
10.1.2.2 -> 10.1.3.2 FTP-DATA C port=52953
10.1.2.2 -> 10.1.3.2 FTP-DATA C port=52954
10.1.2.2 -> 10.1.3.2 FTP-DATA C port=52955
10.1.2.2 -> 10.1.3.2 FTP-DATA C port=52956
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 5117 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=37790
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 5117 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 FTP C port=52957
10.1.2.2 -> 10.1.3.2 FTP C port=52958
10.1.2.2 -> 10.1.3.2 FTP C port=52959
10.1.2.2 -> 10.1.3.2 FTP C port=52960
10.1.2.2 -> 10.1.3.2 FTP C port=52961
10.1.2.2 -> 10.1.3.2 FTP C port=52962
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 41935 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=48694
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 41935 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 TELNET C port=52963
10.1.3.2 -> 10.1.2.2 TELNET R port=52963
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 19862 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=41665
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 19862 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 SMTP C port=52964
10.1.2.2 -> 10.1.3.2 SMTP C port=52965
10.1.2.2 -> 10.1.3.2 SMTP C port=52966
10.1.2.2 -> 10.1.3.2 SMTP C port=52967
10.1.2.2 -> 10.1.3.2 SMTP C port=52968
10.1.2.2 -> 10.1.3.2 SMTP C port=52969
```

```
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 39721 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 39721 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 UDP D=53 S=51369 LEN=8
10.1.2.2 -> 10.1.3.2 UDP D=53 S=51370 LEN=8
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 34973 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 34973 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 NTP C port=36360
10.1.2.2 -> 10.1.3.2 NTP C port=36361
^C#
```

This corresponding section of snoop output shows the traffic on the `qfe2` interface. The host on this subnet is configured to have IP address 10.1.3.2, corresponding to the GIAC web server. The only traffic that was allowed through the firewall was the appropriate HTTP, HTTPS, and telnet traffic. The firewall blocked the FTP, FTP-data, SMTP, DNS, and NTP traffic.

```
# snoop -d qfe2
Using device /dev/qfe (promiscuous mode)
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 7194 Sequence number: 0)
10.1.3.2 -> (broadcast)  ARP C Who is 10.1.3.1, sparky-qfe2 ?
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 7194 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTP C port=52949
10.1.3.2 -> 10.1.2.2 HTTP R port=52949
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 43137 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 43137 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 HTTPS C port=52950
10.1.3.2 -> 10.1.2.2 HTTPS R port=52950
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 34731 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 34731 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 5117 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 5117 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 41935 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 41935 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 TELNET C port=52963
10.1.3.2 -> 10.1.2.2 TELNET R port=52963
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 19862 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 19862 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 39721 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 39721 Sequence number: 0)
10.1.2.2 -> 10.1.3.2 ICMP Echo request (ID: 34973 Sequence number: 0)
10.1.3.2 -> 10.1.2.2 ICMP Echo reply (ID: 34973 Sequence number: 0)
^C#
```

The following is the tcpdump output from the proxied network while being scanned from the service network. This output also shows that only packets of the appropriate types arrived at the host at 10.1.3.2.

```
16:36:54.488838 10.1.2.2 > 10.1.3.2: icmp: echo request
16:36:54.489133 10.1.3.2 > 10.1.2.2: icmp: echo reply
16:37:04.796710 10.1.2.2.52949 > 10.1.3.2.http: S
    483384993:483384993(0) win 5840 <mss 1460,sackOK,timestamp
    141962901 0,nop,wscale 0> (DF)
16:37:04.796769 10.1.3.2.http > 10.1.2.2.52949: R 0:0(0) ack
    483384994 win 0 (DF)
```

```
16:37:15.383396 10.1.2.2 > 10.1.3.2: icmp: echo request
16:37:15.383457 10.1.3.2 > 10.1.2.2: icmp: echo reply
16:37:25.691588 10.1.2.2.52950 > 10.1.3.2.https: S
    512616657:512616657(0) win 5840 <mss 1460,sackOK,timestamp
    141973599 0,nop,wscale 0> (DF)
16:37:25.691651 10.1.3.2.https > 10.1.2.2.52950: R 0:0(0) ack
    512616658 win 0 (DF)
16:37:30.080103 10.1.2.2 > 10.1.3.2: icmp: echo request
16:37:30.080161 10.1.3.2 > 10.1.2.2: icmp: echo reply
16:37:45.041764 10.1.2.2 > 10.1.3.2: icmp: echo request
16:37:45.041836 10.1.3.2 > 10.1.2.2: icmp: echo reply
16:37:59.868063 10.1.2.2 > 10.1.3.2: icmp: echo request
16:37:59.868131 10.1.3.2 > 10.1.2.2: icmp: echo reply
16:38:10.175472 10.1.2.2.52963 > 10.1.3.2.telnet: S
    550944751:550944751(0) win 5840 <mss 1460,sackOK,timestamp
    141996375 0,nop,wscale 0> (DF)
16:38:10.175542 10.1.3.2.telnet > 10.1.2.2.52963: R 0:0(0) ack
    550944752 win 0 (DF)
16:38:13.608741 10.1.2.2 > 10.1.3.2: icmp: echo request
16:38:13.608806 10.1.3.2 > 10.1.2.2: icmp: echo reply
16:38:33.539574 10.1.2.2 > 10.1.3.2: icmp: echo request
16:38:33.539640 10.1.3.2 > 10.1.2.2: icmp: echo reply
16:38:48.496416 10.1.2.2 > 10.1.3.2: icmp: echo request
16:38:48.496487 10.1.3.2 > 10.1.2.2: icmp: echo reply
```

For the next block of output, the host on the proxied subnet has been reconfigured to have address 10.1.3.3, corresponding to the GIAC FTP server. The same set of scans as above is run. The only difference is that the scan is targeted to 10.1.3.3 instead of 10.1.3.2. The incoming traffic is essentially the same as before.

```
# snoop -d qfe1
Using device /dev/qfe (promiscuous mode)
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 52682 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTP C port=42167
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 52682 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTP C port=52928
10.1.2.2 -> 10.1.3.3 HTTP C port=52929
10.1.2.2 -> 10.1.3.3 HTTP C port=52930
10.1.2.2 -> 10.1.3.3 HTTP C port=52931
10.1.2.2 -> 10.1.3.3 HTTP C port=52932
10.1.2.2 -> 10.1.3.3 HTTP C port=52933
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 30996 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTP C port=58325
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 30996 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTPS C port=52934
10.1.2.2 -> 10.1.3.3 HTTPS C port=52935
10.1.2.2 -> 10.1.3.3 HTTPS C port=52936
10.1.2.2 -> 10.1.3.3 HTTPS C port=52937
10.1.2.2 -> 10.1.3.3 HTTPS C port=52938
10.1.2.2 -> 10.1.3.3 HTTPS C port=52939
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 7241 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTP C port=57189
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 7241 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 FTP-DATA C port=52940
```

```
10.1.3.3 -> 10.1.2.2 FTP-DATA R port=52940
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 61654 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTP C port=38415
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 61654 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 FTP C port=52941
10.1.3.3 -> 10.1.2.2 FTP R port=52941
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 36998 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTP C port=51585
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 36998 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 TELNET C port=52942
10.1.3.3 -> 10.1.2.2 TELNET R port=52942
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 28684 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 HTTP C port=61504
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 28684 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 SMTP C port=52943
10.1.2.2 -> 10.1.3.3 SMTP C port=52944
10.1.2.2 -> 10.1.3.3 SMTP C port=52945
10.1.2.2 -> 10.1.3.3 SMTP C port=52946
10.1.2.2 -> 10.1.3.3 SMTP C port=52947
10.1.2.2 -> 10.1.3.3 SMTP C port=52948
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 58533 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 58533 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 UDP D=53 S=49773 LEN=8
10.1.2.2 -> 10.1.3.3 UDP D=53 S=49774 LEN=8
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 53669 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 53669 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 NTP C port=49520
10.1.2.2 -> 10.1.3.3 NTP C port=49521
^C#
```

This time, the traffic allowed through the firewall includes only the FTP, FTP-data, and telnet, as expected.

```
# snoop -d qfe2
Using device /dev/qfe (promiscuous mode)
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 52682 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 52682 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 30996 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 30996 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 7241 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 7241 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 FTP-DATA C port=52940
10.1.3.3 -> 10.1.2.2 FTP-DATA R port=52940
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 61654 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 61654 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 FTP C port=52941
10.1.3.3 -> 10.1.2.2 FTP R port=52941
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 36998 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 36998 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 TELNET C port=52942
10.1.3.3 -> 10.1.2.2 TELNET R port=52942
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 28684 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 28684 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 58533 Sequence number: 0)
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 58533 Sequence number: 0)
10.1.2.2 -> 10.1.3.3 ICMP Echo request (ID: 53669 Sequence number: 0)
```

```
10.1.3.3 -> 10.1.2.2 ICMP Echo reply (ID: 53669 Sequence number: 0)
^C#
```

As above, the output from tcpdump shows that only the appropriate packets made it across the firewall to the host at 10.1.3.3.

```
16:31:45.411668 10.1.2.2 > 10.1.3.3: icmp: echo request
16:31:45.411725 10.1.3.3 > 10.1.2.2: icmp: echo reply
16:32:06.967721 10.1.2.2 > 10.1.3.3: icmp: echo request
16:32:06.967782 10.1.3.3 > 10.1.2.2: icmp: echo reply
16:32:24.029466 10.1.2.2 > 10.1.3.3: icmp: echo request
16:32:24.029530 10.1.3.3 > 10.1.2.2: icmp: echo reply
16:32:34.337719 10.1.2.2.52940 > 10.1.3.3.ftp-data: S
    204228872:204228872(0) win 5840 <mss 1460,sackOK,timestamp
    141824424 0,nop,wscale 0> (DF)
16:32:34.337789 10.1.3.3.ftp-data > 10.1.2.2.52940: R 0:0(0) ack
    204228873 win 0 (DF)
16:32:40.314135 10.1.2.2 > 10.1.3.3: icmp: echo request
16:32:40.314203 10.1.3.3 > 10.1.2.2: icmp: echo reply
16:32:50.622667 10.1.2.2.52941 > 10.1.3.3.ftp: S
    209471148:209471148(0) win 5840 <mss 1460,sackOK,timestamp
    141832762 0,nop,wscale 0> (DF)
16:32:50.622741 10.1.3.3.ftp > 10.1.2.2.52941: R 0:0(0) ack 209471149
    win 0 (DF)
16:32:56.548132 10.1.2.2 > 10.1.3.3: icmp: echo request
16:32:56.548195 10.1.3.3 > 10.1.2.2: icmp: echo reply
16:33:06.854989 10.1.2.2.52942 > 10.1.3.3.telnet: S
    222996315:222996315(0) win 5840 <mss 1460,sackOK,timestamp
    141841073 0,nop,wscale 0> (DF)
16:33:06.855057 10.1.3.3.telnet > 10.1.2.2.52942: R 0:0(0) ack
    222996316 win 0 (DF)
16:33:11.750993 10.1.2.2 > 10.1.3.3: icmp: echo request
16:33:11.751074 10.1.3.3 > 10.1.2.2: icmp: echo reply
16:33:34.463134 10.1.2.2 > 10.1.3.3: icmp: echo request
16:33:34.463200 10.1.3.3 > 10.1.2.2: icmp: echo reply
16:33:49.729589 10.1.2.2 > 10.1.3.3: icmp: echo request
16:33:49.729663 10.1.3.3 > 10.1.2.2: icmp: echo reply
```

3.2.4.3 Proxied Network to Internal Network

The firewall rules do not allow the proxied network to initiate any connections to the internal network. It may accept incoming connections, but not initiate any. Nmap was used to probe the internal network from the host on the proxied network. The following output from snoop shows the message traffic generated by a sequence of Nmap "miniscans," each directed to one specific TCP or UDP port. An ICMP echo request is part of each scan. Interface `qfe2` is the interface from the firewall to the proxied network, so this snoop output shows the traffic before it is filtered by the firewall. The only outbound traffic (back to the proxied network) is the ICMP echo replies coming from the host on the internal network. (Some blank fill was edited from this output so the lines would not wrap onto a second line.)

```
# snoop -d qfe2
```

```
Using device /dev/qfe (promiscuous mode)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 55586 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 55586 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 HTTP C port=55439
10.1.3.2 -> 10.1.4.2 HTTP C port=34671
10.1.3.2 -> 10.1.4.2 HTTP C port=34672
10.1.3.2 -> 10.1.4.2 HTTP C port=34673
10.1.3.2 -> 10.1.4.2 HTTP C port=34674
10.1.3.2 -> 10.1.4.2 HTTP C port=34675
10.1.3.2 -> 10.1.4.2 HTTP C port=34676
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 32735 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 32735 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 HTTP C port=46224
10.1.3.2 -> 10.1.4.2 HTTPS C port=34677
10.1.3.2 -> 10.1.4.2 HTTPS C port=34678
10.1.3.2 -> 10.1.4.2 HTTPS C port=34679
10.1.3.2 -> 10.1.4.2 HTTPS C port=34680
10.1.3.2 -> 10.1.4.2 HTTPS C port=34681
10.1.3.2 -> 10.1.4.2 HTTPS C port=34682
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 48909 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 48909 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 HTTP C port=35949
10.1.3.2 -> 10.1.4.2 FTP-DATA C port=34683
10.1.3.2 -> 10.1.4.2 FTP-DATA C port=34684
10.1.3.2 -> 10.1.4.2 FTP-DATA C port=34685
10.1.3.2 -> 10.1.4.2 FTP-DATA C port=34686
10.1.3.2 -> 10.1.4.2 FTP-DATA C port=34687
10.1.3.2 -> 10.1.4.2 FTP-DATA C port=34688
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 45278 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 45278 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 HTTP C port=36793
10.1.3.2 -> 10.1.4.2 FTP C port=34689
10.1.3.2 -> 10.1.4.2 FTP C port=34690
10.1.3.2 -> 10.1.4.2 FTP C port=34691
10.1.3.2 -> 10.1.4.2 FTP C port=34692
10.1.3.2 -> 10.1.4.2 FTP C port=34693
10.1.3.2 -> 10.1.4.2 FTP C port=34694
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 14037 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 14037 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 HTTP C port=52470
10.1.3.2 -> 10.1.4.2 TELNET C port=34695
10.1.3.2 -> 10.1.4.2 TELNET C port=34696
10.1.3.2 -> 10.1.4.2 TELNET C port=34697
10.1.3.2 -> 10.1.4.2 TELNET C port=34698
10.1.3.2 -> 10.1.4.2 TELNET C port=34699
10.1.3.2 -> 10.1.4.2 TELNET C port=34700
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 836 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 836 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 HTTP C port=56702
10.1.3.2 -> 10.1.4.2 SMTP C port=34701
10.1.3.2 -> 10.1.4.2 SMTP C port=34702
10.1.3.2 -> 10.1.4.2 SMTP C port=34703
10.1.3.2 -> 10.1.4.2 SMTP C port=34704
10.1.3.2 -> 10.1.4.2 SMTP C port=34705
10.1.3.2 -> 10.1.4.2 SMTP C port=34706
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 51733 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 51733 Sequence number: 0)
```

```
10.1.3.2 -> 10.1.4.2 UDP D=53 S=54125 LEN=8
10.1.3.2 -> 10.1.4.2 UDP D=53 S=54126 LEN=8
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 43804 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 43804 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 NTP C port=47595
10.1.3.2 -> 10.1.4.2 NTP C port=47596
^C#
```

The next block of snoop output shows the corresponding traffic, outbound from the firewall. Only the ping requests and replies show up on this side of the firewall. The firewall rules have blocked all of the other traffic.

```
# snoop -d qfe3
Using device /dev/qfe (promiscuous mode)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 55586 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 55586 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 32735 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 32735 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 48909 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 48909 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 45278 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 45278 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 14037 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 14037 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 836 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 836 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 51733 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 51733 Sequence number: 0)
10.1.3.2 -> 10.1.4.2 ICMP Echo request (ID: 43804 Sequence number: 0)
10.1.4.2 -> 10.1.3.2 ICMP Echo reply (ID: 43804 Sequence number: 0)
^C#
```

Finally, the output of tcpdump, run on the host at 10.1.4.2, verifies that no traffic but the expected ICMP echo requests was received on the internal network.

```
13:56:24.042692 10.1.3.2 > 10.1.4.2: icmp: echo request
13:56:24.042765 10.1.4.2 > 10.1.3.2: icmp: echo reply
13:57:00.817775 10.1.3.2 > 10.1.4.2: icmp: echo request
13:57:00.817825 10.1.4.2 > 10.1.3.2: icmp: echo reply
13:57:20.265400 10.1.3.2 > 10.1.4.2: icmp: echo request
13:57:20.265466 10.1.4.2 > 10.1.3.2: icmp: echo reply
13:57:37.316942 10.1.3.2 > 10.1.4.2: icmp: echo request
13:57:37.317013 10.1.4.2 > 10.1.3.2: icmp: echo reply
13:57:54.655849 10.1.3.2 > 10.1.4.2: icmp: echo request
13:57:54.655914 10.1.4.2 > 10.1.3.2: icmp: echo reply
13:58:10.898462 10.1.3.2 > 10.1.4.2: icmp: echo request
13:58:10.898532 10.1.4.2 > 10.1.3.2: icmp: echo reply
13:58:44.154384 10.1.3.2 > 10.1.4.2: icmp: echo request
13:58:44.154455 10.1.4.2 > 10.1.3.2: icmp: echo reply
13:59:06.449338 10.1.3.2 > 10.1.4.2: icmp: echo request
13:59:06.449406 10.1.4.2 > 10.1.3.2: icmp: echo reply
```

A surprise during this part of the testing was that Nmap reported the UDP ports on 10.1.4.2 to be open, as shown by Figure 9.

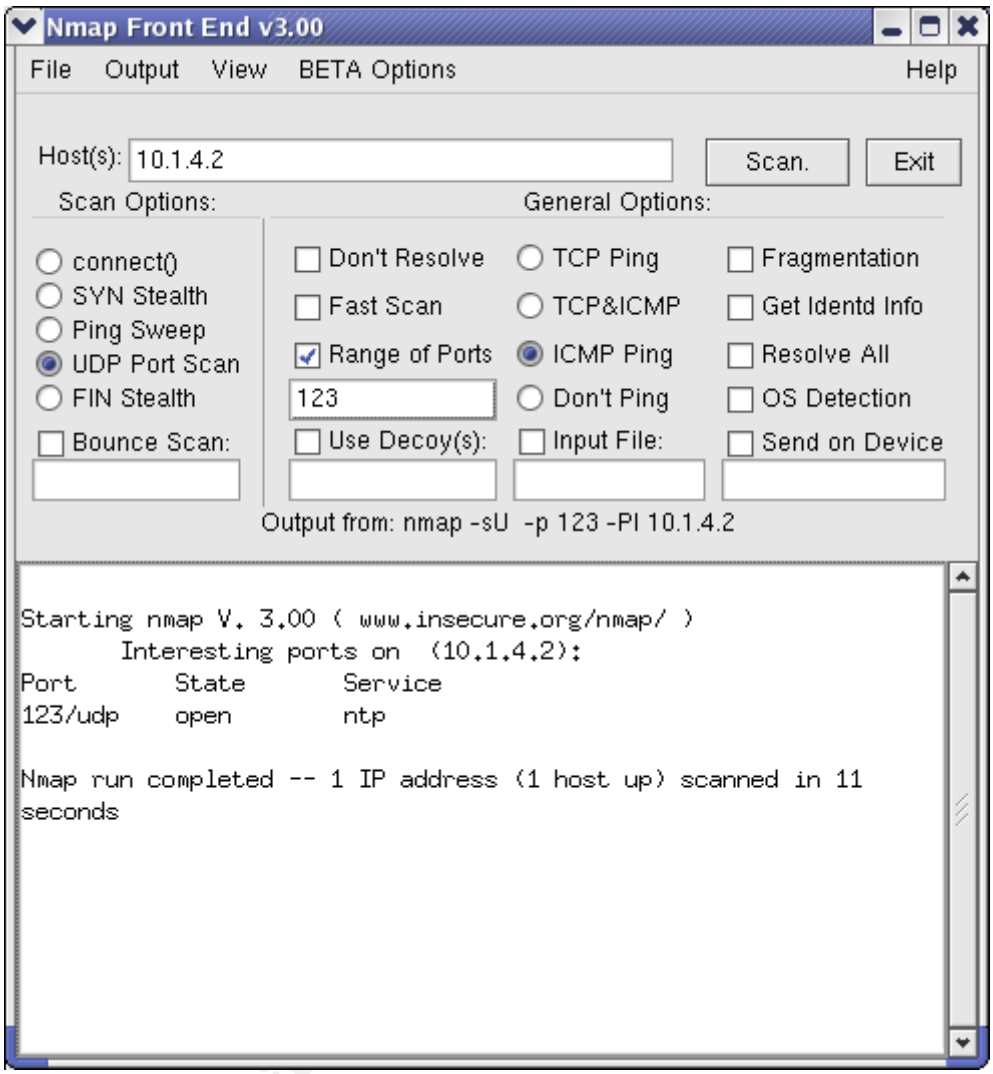

Figure 9: Nmap Scan of UDP Port 123 on 10.1.4.2 from 10.1.3.2

The TCP ports tested in this step were reported by Nmap to be filtered, as expected. Nmap did not receive any "destination unreachable" replies to the UDP messages, and reported them to be open. GIAC should correct the rules that block this UDP traffic, and add the flag that causes the "destination unreachable" message to be returned.

3.3 Audit Summary

In general, the rule set configured for the IPFilter firewall performed well in the audit. It did not reveal any unnecessary information to a scan attempted from outside of the GIAC network. It allowed traffic to pass where the designer intended it to pass, and blocked the traffic otherwise.

Several changes should be made to improve the security of GIAC's network. The first of those changes is to disable the dangerous "r" commands and telnet

entirely from the network. As it is, they are disabled from coming in through the firewall from outside the network, but that is not sufficient protection. An attacker who compromised a machine on the service network would find that the rest of the system was easy prey. The "r" commands and telnet should be replaced with SSH.

Another change that should be done is to consistently use the "return-rst" and "return-icmp-as-dest (port-unr)" settings for blocked TCP and UDP ports. These settings help to hide the presence of the firewall. When these settings are not used, blocked packets are just dropped. An attacker can infer the presence of the firewall from the fact that nothing was returned to his probe. When a reset is returned, the port appears to be closed, and the attacker cannot distinguish the firewall from a normal system that is not listening on the selected port. In the firewall rule set as given, return-rst is used sporadically, but is not found everywhere where it should be used. It should be noted that using "return-rst" does complicate the rule set, because it only applies to TCP. In some cases, it is necessary to write rules that are nearly duplicates of each other, one to handle TCP and another to handle everything else. The order of the rules also becomes an even more important factor.

A third change that should be recommended is to consider adding rules to block almost all access to the firewall itself, from within the GIAC network as well as from the outside. This action would safeguard GIAC from a malicious or disgruntled employee who otherwise might find it too easy to modify the firewall from within the network and disable some of the security settings.

Another security concern is that ICMP echo requests and replies are allowed free travel throughout the GIAC network. While it is convenient to have the ability to ping one machine from another to in order to more easily diagnose problems during the initial setup of the network, most of the ICMP traffic should be disabled once that initial setup is complete and the network bugs have been eliminated.

A final change that might be considered involves an additional option that can be used on IPFilter firewall rules: '`fastroute`'. If this option is used, packets crossing the firewall do not have the TTL value decremented. An attacker analyzing the traffic does not see a "hop" for the firewall and may not suspect its presence.

4 Design under Fire

4.1 Introduction

"If you believe you have a foolproof system, you have failed to take into consideration the creativity of fools," says former con-artist Frank Abagnale, the subject of the recent Spielberg film "Catch Me if You Can [35]." Network security engineers would be well advised to take this advice to heart.

The goal of this section of the paper is to demonstrate how a firewall-protected network may be attacked despite the security precautions, using a previously submitted SANS GCFW practical. Nick Read, GCFW Analyst 0357, presented the practical chosen for this exercise in December 2002 [36]. Mr. Read designed a multi-tiered architecture with a service network, or DMZ, behind the primary firewall. The DMZ contains a web server, DNS server, NTP server, and mail server. Additional company-private networks and servers are protected behind a secondary firewall. The network layout is shown in Figure 10.

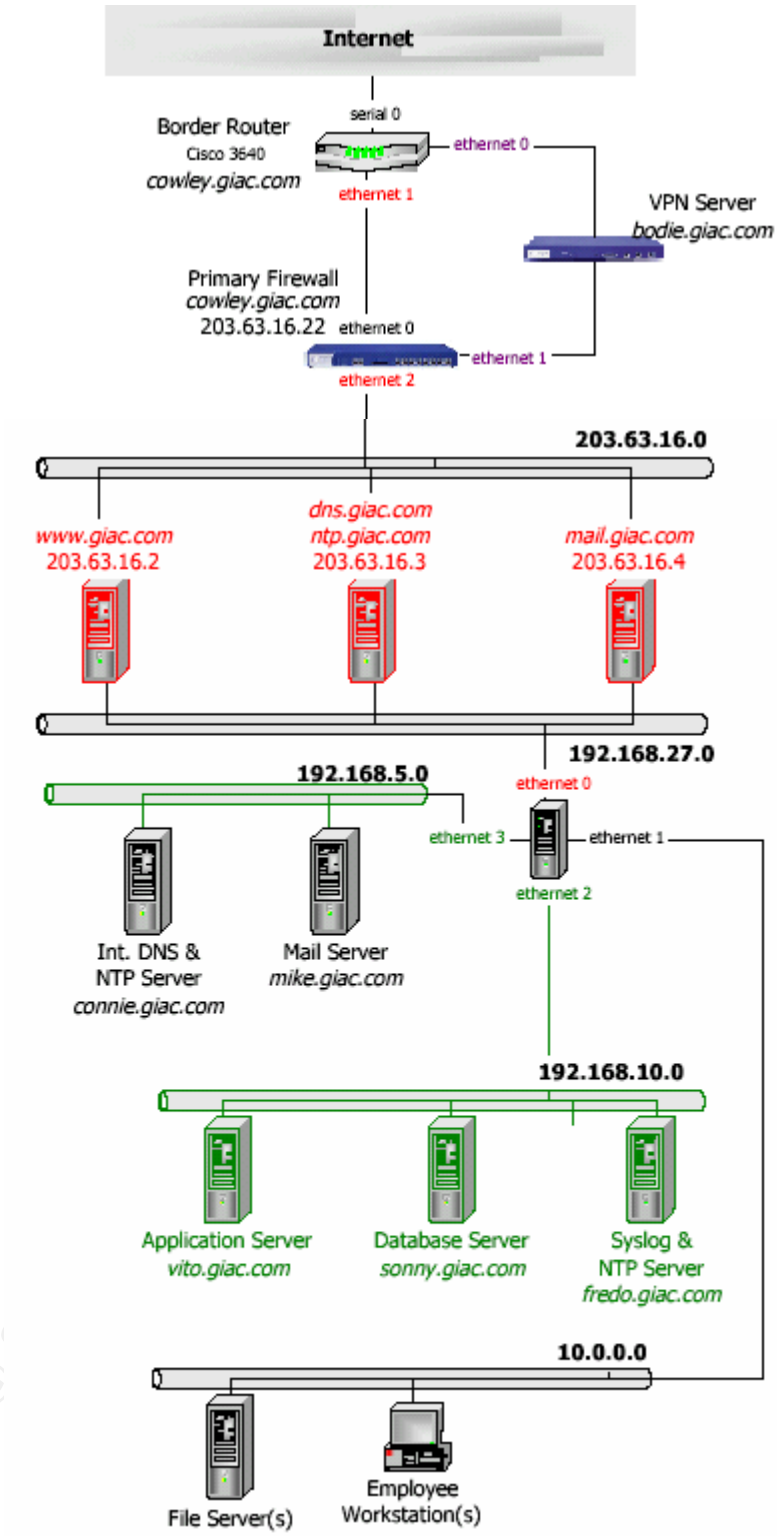

Figure 10: GIAC Network Diagram from Nick Read's Practical

4.2 Attack on the Firewall

The firewall used in Mr. Read's design is a NetScreen 204 Security Appliance running ScreenOS 4.0.0r2. An advisory was posted concerning a vulnerability in NetScreen firewalls on November 25, 2002 [37]. The vulnerability affected all ScreenOS versions up to and including 4.0.0. The problem is that the ScreenOS produces predictable TCP sequence numbers. If an attacker can gain access to network traffic, he may be able to analyze the pattern of sequence numbers. If so, he can inject packets into the network that will be accepted as legitimate traffic. The ability to inject packets may further enable the attacker to hijack the session. Additional background information on the problems posed by predictable TCP sequence numbers has been published by the CERT Coordination Center [38].

The NetScreen advisory also noted, "This vulnerability is not exploitable on TCP traffic secured via IPSec, SSH, or other mechanisms that make interception and modification of traffic detectable [37]."

Further information may be found at the SecurityFocus website:

> This [vulnerability] may allow an attacker to inject packets or launch man-in-the-middle attacks against a target network session. … For this issue to be exploitable the attacker must be able to access to network session traffic, possibly requiring access to a local network [39].

In order to exploit this vulnerability, an attacker could attempt to perform a "man-in-the-middle" attack against the NetScreen firewall. One tool that is available to implement such an attack is the "Hunt" tool [40]. Hunt, by Pavel Krauz, is a program that sniffs packets on the network, collects information about the addresses and sequence numbers, and injects modified or entirely new packets into the traffic stream.

Bhavin Bharat Bhansali's paper in the SANS reading room describes how to use Hunt to execute a man-in-the-middle attack [41]. (The link for downloading Hunt given in that paper is out-of-date.) The attacker must be running as root in a session on the same network as one of GIAC's customers, suppliers, or remote employees. This might happen if a customer had a disgruntled employee, but more likely would happen by the attacker having previously compromised and gained root privileges on one of the customer's machines. The attacker would start up Hunt on the compromised machine and wait for the customer to begin to make a connection to the GIAC network. At this point, the attacker could intercept data being exchanged by the customer and GIAC. This data might include the customer's user name and password, or credit card information. A much more damaging possibility is that the attacker would use Hunt's RST daemon to reset the connection on the customer's end and hijack the session. The customer would lose the connection, but more than likely would just shrug it

off as being some unknown network problem, either at GIAC's end or on the path between the two sites. However, the session would remain open at GIAC's end and the attacker would now be in control of the session. He could then attempt additional exploits in order to raise his level of privilege within GIAC's network.

This attack would be even more dangerous if the session being hijacked belonged to a GIAC mobile employee, who would have more privileges within the GIAC network. However, in Mr. Read's security design, "most online interaction is encrypted to prevent interception by an unauthorised party," and "mobile employees' connections are encrypted by the VPN server, using IPSEC" [36]. In order to succeed, the Hunt tool needs to find an unencrypted session [41]. Therefore, this attack should not succeed against GIAC's mobile employees.

The simplest and best counter-measure against this type of man-in-the-middle attack or any other attack based on predicting the TCP sequence numbers is to upgrade the NetScreen Security Appliance to ScreenOS 4.0.1, which was made available by NetScreen on November 25, 2002, i.e., the same date on which they issued the advisory about the vulnerability [37].

4.3 DDoS Attack

There is, unfortunately, any number of tools that an attacker might choose from in order to execute a distributed denial of service (DDoS) attack against the GIAC network. One of the most common such attacks is the SYN flood attack. In a SYN flood attack, the attacker starts by compromising a reasonably large number of hosts. For purposes of this discussion, it is assumed that the attacker has the ability to control fifty systems that are connected to the Internet via cable-modem or DSL. Such systems most often belong to home users who are unlikely to have applied any patches or installed a firewall, and are easy to find [42]. The attacker then installs and runs agent code on each of the compromised systems. The agent will then listen over the Internet for commands from a handler. Some tools use the TCP, UDP, or ICMP protocols [43] for handler-agent communication, while some more recent tools use Internet Relay Chat (IRC) channels [44]. When everything is in place, the attacker has only to execute a single command at the console of his computer. The master computer will signal each of the agents, which will in turn begin to bombard the victim with TCP packets with the SYN flag set. Since TCP SYN is the first step in opening a TCP connection, the target computer will allocate resources to what it thinks is a genuine connection, and will send back a TCP SYN-ACK message. The agent computers, however, will not respond with the final TCP ACK packet to complete the connection. The target host will have an increasing number of half-open connections and will eventually run out of resources, at which point it will hang or crash.

If such an attack were to be tried on the GIAC network as designed by Mr. Read, it should not be entirely successful. The NetScreen firewall has been set up with the following commands [36]:

```
set firewall syn-flood       Block SYN flood
set syn-threshold 200        Set SYN threshold to 200 packets per sec
```

These commands should enable the firewall to detect the SYN flood attack and to take some counter-measures. It reacts by caching the connection attempts and not passing them to the server until the TCP handshake is complete. While this action may protect the server, the firewall itself must use resources to respond to the attack in this way, and the GIAC network may suffer at least a slowdown, if not a full stoppage.

A persistent attacker might try a different approach. The HoneyNet project occasionally features the "Reverse Challenge," in which participants attempt to reverse engineer a sample of an attack tool captured in one of the project's hosts [45]. The most recent challenge featured a tool that allows the attacker to execute two different types of DNS floods.

As reported by Dion Mendel, the winner of the challenge, "The DNS flooding attack is a custom attack to cause many thousands of machines on the Internet to send unasked for DNS replies to the victim. This huge amount of network traffic is intended to prevent the victim from being able to utilise their network connection [46]." The attack agent contains the addresses of 8000 DNS servers hard-coded into its instructions. The agent can be commanded to send SOA queries to these servers with the victim's IP address spoofed as the source of the query [47]. The victim then is overwhelmed with a flood of replies.

The second type of DNS flood used by this agent sends SOA queries directly to the victim's DNS server [47]. The source IP address in the queries is spoofed to random addresses.

If it is assumed that the attacker in this case is able to obtain the code for both the agent and the handler for this attack tool, he can install copies of the agent on the fifty DSL/cable-modem hosts that he compromised earlier. He would then be in a position to launch either of these DNS flood attacks against the GIAC network. He would first need to scan the network to determine the IP address of a machine accepting DNS queries. This activity is facilitated by the fact that the IP addresses of the hosts on GIAC's service network, as designed by Mr. Read, are not protected by Network Address Translation (NAT). Once the DNS server is identified, the attacker would then use the handler to instruct the agents to begin the attack.

Either of these DNS flood attacks should be successful against the GIAC network as designed by Mr. Read. His router and firewall rules explicitly allow UDP traffic to port 53 on his DNS server, in order to allow name resolution. Thus all of the traffic generated by this attack should get through and flood the DNS server, either causing it to crash or at least making it so busy that all other network activity is greatly slowed.

The DNS flood attack does not have simple counter measures. One thing the GIAC IT staff could do is to rate-limit UDP traffic in the router, which would cause it to start dropping UDP traffic when the limit is exceeded. This action would help protect the DNS server but could still deny access to the GIAC network to some customers, because legitimate traffic would be dropped along with the bogus traffic from the attacking agents.

4.4 Attack on an Internal System

Mr. Read does not specify in his paper what web server software is being used in his version of the GIAC network. However, the paper contains other references to Windows 2000-based workstations and laptops, so it seems a safe assumption that the web server is a Windows 2000 machine running Microsoft's Internet Information Server (IIS), which has a reputation as a highly vulnerable system. An attacker, lacking any more specific information, would need to spend some time carefully probing the network to validate these guesses.

However, under the assumption that the web server is IIS, perhaps version 5.0, a recently published advisory points out an avenue worth exploring, which ought to result in a successful attack [48][49]. The vulnerability is a buffer overflow in the ntdll.dll component of WebDAV (World Wide Web Distributed Authoring and Versioning), which is supported by IIS 5.0.

WebDAV is an extension to the HTTP/1.1 protocol. It enables remote web authoring operations, including "the management of resource properties, creation and management of resource collections, namespace manipulation, and resource locking (collision avoidance) [50]."

The following excerpt from RFC 2518 is an example of a simple lock request using WebDAV [50].

```
LOCK /workspace/webdav/proposal.doc HTTP/1.1
Host: webdav.sb.aol.com
Timeout: Infinite, Second-4100000000
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="ejw",
   realm="ejw@webdav.sb.aol.com", nonce="...",
   uri="/workspace/webdav/proposal.doc",
   response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:lockinfo xmlns:D='DAV:'>
  <D:lockscope><D:exclusive/></D:lockscope>
  <D:locktype><D:write/></D:locktype>
  <D:owner>
     <D:href>http://www.ics.uci.edu/~ejw/contact.html</D:href>
   </D:owner>
</D:lockinfo>
```

In order to exploit the buffer overflow vulnerability, the attacker must either obtain some exploit code or devise his own. At the time of this writing, although it is suspected that this method has been used "in the wild," no publicly available source of the exploit code is known [51]. However, if the attacker has access to an IIS server on which he can run some tests, it should be possible for him to develop the attack on his own. The approach would be to replace one of the strings in the command above with a very long string, send it to the server, and inspect the results. This process would have to be repeated with different WebDAV commands, using different strings within each command, and varying the lengths of the strings, until the desired result was obtained. When the attacker has determined which command, variable, and string length causes an overflow, he can then add his own commands to the appropriate places in the string in order to have his own code executed once the overflow has happened. More details on how to exploit buffer overflows can be found on the Internet [52].

Once the details of the specific exploit have been discovered and ironed out, the attacker can send the crafted WebDAV command to the GIAC web server. This can be easily done with a script in Perl, Java, or some other programming language, based on the following pseudocode, which was modified from a sample HTTP POST operation [53].

```
$reqBody = "&<?xml version="1.0" encoding="utf-8" ?>\n" +
    "<D:lockinfo xmlns:D='DAV:'>\n" +
    "<D:lockscope><D:exclusive/></D:lockscope>\n" +
    "<D:locktype><D:write/></D:locktype>\n" +
    "<D:owner>\n" +
    "<D:href>http://www.foo.com/theevilhacker.html</D:href>\n" +
    "</D:owner></D:lockinfo>";

$contentLength = length($reqBody);

$requestData =
     "LOCK /workspace/webdav/somefile HTTP/1.1\n" +
     "Host: www.giac.com\n" +
     "Timeout: Infinite, Second-4100000000\n" +
     "Content-Type: text/xml; charset="utf-8"\n" +
     "Content-Length: " + $contentLength + "\n" +
     "Authorization: Digest username=\"teh\",\n" +
     "realm=\"teh@www.foo.com\", nonce=\"...\",\n" +
     "uri=\"/workspace/webdav/somefile\",\n" +
     "response=\"...\", opaque=\"...\"\n\n" +
     $reqBody + "\n";

$socket = openASocket("www.giac.com", kPort80);
writeToASocket($socket, $requestData);
$responseData = readToEOF($socket);
closeASocket($socket);
```

Under the assumption that the GIAC network is running Microsoft IIS version 5.0 and has not yet patched the machine or disabled WebDAV, this attack should

succeed. The GIAC IT staff should apply counter-measures as soon as possible. Microsoft has already issued a patch for this vulnerability [54]. This patch should be applied immediately, especially if there is any need for the web server to be running WebDAV.

It seems more likely that there is no real need for GIAC to keep WebDAV enabled on this server, and a stronger counter-measure would be to turn off the WebDAV functionality, since such action would protect the server, not merely from this particular exploit, but also from future vulnerabilities, that have not yet been discovered.

4.5 Conclusion

The GIAC network as designed by Mr. Read is fairly well secured. However, as with any network attached to the Internet, a determined and clever attacker who is willing to work hard enough will be very likely to find some way through the defenses. An attacker who stays very up-to-date on the latest advisories and published exploits is even more likely to find a hole into the internal network. These possible attacks illustrate the absolute necessity for the IT staff to apply patches and hotfixes immediately. Also, additional layers of defense, such as host-based firewalls and intrusion detection systems, are important as a second line of protection for the times when the primary firewall does get breached.

A Appendix 1: Description of IPFilter

A.1 Introduction

IPFilter is a stateful, packet-filtering, firewall program, originally written in 1993 by Darren Reed, an Australian software designer and consultant. Mr. Reed still actively maintains and updates the code. The latest fully tested and supported version of IPFilter at the time of this writing is version 3.4.31, but an alpha version 4.0 is also available. It is recommended that IPFilter be installed as a loadable kernel module, but it can also be incorporated into the base kernel. IPFilter has been deployed successfully on a variety of UNIX systems including FreeBSD, OpenBSD, NetBSD, IRIX, HP-UX, SunOS, as well as Solaris 2.3 – 9.

The rule sets for IPFilter allow it to pass or drop any packet, based on several factors, including the interface through which the packet is traveling, the source and destination IP addresses, the IP protocol, the port number, the type/code for ICMP packets, and whether or not the packet is fragmented. IPFilter keeps packet state information for TCP, UDP, and ICMP packet flows. It keeps fragment state information for any IP packet and applies the same rule to all fragments. It can act as a Network Address Translator (NAT), and can use redirection to set up transparent proxy connections. It can provide packet header details to an independent user program for to perform authentication [6].

A separate program, 'ipmon,' is provided that can be configured to log passed packets, blocked packets, or packets which match a rule set defining suspicious packets. Another separate program, 'ipfstat,' records and displays statistical data about IPFilter's performance.

A.2 IPFilter Documentation

The IPFilter homepage can be found at http://www.ipfilter.org, (although that address immediately redirects to http://coombs.anu.edu.au/~avalon/ip-filter.html) [6]. The IPFilter web site also contains a page with a description of the IPFilter rule sets, at http://coombs.anu.edu.au/~avalon/rules.html. Another page at the web site contains additional examples of rule sets written to fit certain specific situations, at http://coombs.anu.edu.au/~avalon/examples.html.

Two other important web sites are the Frequently Asked Questions (FAQ) list, found at http://home.earthlink.net/~jaymzh666/ipf/index.html [31], and the "How To" document at http://www.obfuscation.org/ipf/ [25].

Several man pages will be installed along with IPFilter. They may be found in `/opt/ipf/man` after installation. In order to access them, it is necessary either to add `/opt/ipf/man` to the MANPATH environment variable or to copy the files to the more standard location for man pages. There are man pages for the following IPFilter functions: 'ipftest', 'mkfilters', 'ipf', 'ipl', 'ipnat', 'ipfs', 'ipfstat', and 'ipmon.'

A.3 IPFilter Support

With any freeware package, the possible lack of technical support has to be a significant concern. There is no one to call if difficulties are encountered during installation, configuration, or operation. Since there is no seller, there is no warranty, and no organization that is obligated to assist the user to get problems solved. However, to offset these concerns, IPFilter does have an active user community with an online forum, which records from 200 to 500 messages per month. Many of the participants are experts in the operation of IPFilter and are willing to help other users get problems solved.

In order to post a question to the user's forum, a subscription to the mailing list is required. The list may be subscribed to by sending an e-mail with "subscribe ipfilter" in the body of the message to majordomo@coombs.anu.edu.au. As an alternative to subscribing to the list, the messages from the list are archived at http://marc.theaimsgroup.com/?l=ipfilter [55].

A.4 Pre-compiled Binary

There is a pre-compiled Solaris 2.8 binary for IPFilter that can be downloaded and installed instead of downloading the source files and compiling from them. The web site where the binary can be found happens to have a rather odd and suspicious-sounding URL: maraudingpirates.org [56]. A 'whois' inquiry reveals that the domain is owned by David F. Newman, Newman Consulting of Byfield, Massachusetts [57]. Further web searches turned up no reason to be suspicious of this person or organization. The binaries are signed with a PGP key.

References

[1] Cisco Systems, Inc. "Cisco VPN Client." URL: http://www.cisco.com/en/US/products/sw/secursw/ps2308/index.html. (15 Mar 2003).

[2] Rainbow Technologies. "iKey USB token, Authentication Key." 25 Jul 2002. URL: http://www.rainbow.com/ikey/. (25 Mar 2003).

[3] Cisco Systems, Inc. "Cisco 2600 Series Multiservice Platforms." URL: http://www.cisco.com/en/US/products/hw/routers/ps259/index.html. (15 Mar 2003)

[4] Cisco Systems, Inc. "Cisco 2600 Series Multiservice Platforms: Cisco 1700, 2600, 3600, 3700 Series Secure VPN Router Bundles." URL: http://www.cisco.com/en/US/products/hw/routers/ps259/products_data_sheet09186a00800921d5.html. (15 Mar 2003).

[5] Cisco Systems, Inc. "Understanding 4- and 8-Port Async/Sync Network Modules." URL: http://www.cisco.com/warp/public/107/hw_as.shtml. (26 Mar 2003).

[6] Reed, Darren. "IPFilter - TCP/IP Firewall/NAT Software." URL: http://coombs.anu.edu.au/~avalon/. (15 Mar 2003.)

[7] Reed, Darren. "IP Filter Examples." URL: http://coombs.anu.edu.au/~avalon/examples.html. (22 Mar 2003).

[8] Sun Microsystems. "SunSolve™ Online." URL: http://sunsolve.sun.com/. (19 Mar 2003).

[9] Chouanard, Jean, "How to install Solaris and have a good host security." 19 Nov 2000. URL: http://www.yassp.org/. (17 Mar 2003).

[10] Tripwire, Inc. "Tripwire Academic Source Release - Intrusion Detection & More." URL: http://www.tripwire.com/products/tripwire_asr/. (19 Mar 2003).

[11] Cisco Systems, Inc. "Cisco ConfigMaker." URL: http://www.cisco.com/warp/public/cc/pd/nemnsw/cm/index.shtml. (10 Mar 2003).

[12] Akin, Thomas. "Hardening Cisco Routers." O'Reilly & Associates, Inc., Sebastopol, CA, 2002. p. 86.

[13] Thomas, Rob. "Secure IOS Template Version 2.7." 13 Feb 2003. URL: http://www.cymru.com/Documents/secure-ios-template.html. (10 Mar 2003).

[14] Cisco Systems, Inc. "PAD and X.25 Connection Setup Commands." URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/cs/csprtd/csprtd2/cspad.htm. (25 Mar 2003).

[15] Greene, Barry, and Philip Smith. "Cisco ISP Software and Router Management." InformIT.com, 7 Jun 2002. URL: http://www.informit.com/isapi/product_id~%7B76F30246-FBA6-4C05-BBC9-1E679DCBD05D%7D/element_id~%7BA4CCE7D9-ECBF-4457-9680-617821C116B5%7D/st~%7B82A60075-2BDA-4E61-B46C-75C270CC4DE5%7D/content/articlex.asp. (21 Mar 2003).

[16] Cisco Systems, Inc. "Using service tcp-keepalives to Avoid Hung Telnet Sessions." 3 Mar 2003. URL: http://www.cisco.com/warp/public/471/tcpkeepalive.html. (10 Mar 2003).

[17] Khatibi, Ramin. "Cisco logging config explained." Badapple.net, 18 Nov 2002. URL: http://www.badapple.net/tech/logging-config.html. (21 Mar 2003).

[18] Cisco Systems, Inc. "Configuring TCP Intercept (Prevent Denial-of-Service Attacks)." URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt3/scdenial.htm. (21 Mar 2003).

[19] Cisco Systems, Inc. "RIP Commands." 17 Jan 2003. URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/np1_r/1rprt1/1rrip.htm. (1 Apr 2003).

[20] Cisco Systems, Inc. "Internet Key Exchange Security Protocol Commands." URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12supdoc/12cmdsum/12cssec/csike.htm#5631. (26 Mar 2003).

[21] INRGI Ebusiness Webhosting Solutions. "Secure IOS Template Case Study." URL: http://www.inrgi.net/security/secure_ios_template.htm. (23 Mar 2003).

[22] Christensen, Steve. "Freeware for Solaris." Steven M. Christensen and Associates, Inc., 9 Mar 2003. URL: http://www.sunfreeware.com/. (24 Mar 2003).

[23] Thang, and Hoang. "Building a Firewall with Solaris." Unixcircle.com, 9 Dec 2001. URL: http://unixcircle.com/features/BuildingSolarisFW.php. (29 Mar 2003).

[24] Kurland, Vadim, and Vadim Zaliva. "Firewall Builder." 2001. URL: http://www.fwbuilder.org/. (1 April 2003).

[25] Conoboy, Brendan, and Erik Fichtner. "IP Filter Based Firewalls HOWTO." 11 Dec 2002. URL: http://www.obfuscation.org/ipf/. (22 Mar 2003).

[26] Bengt. "IPFilter rules beeing [sic] complicated (backwards to me." OpenBSD Journal. 29 May ?. URL: http://www.deadly.org/article.php3?sid=20000527083510. (1 Apr 2003).

[27] Thomas, Rob. " Bogon List v1.7." 13 Feb 2003. URL: http://www.cymru.com/Documents/bogon-list.html. (27 Mar 2003).

[28] Network Working Group. "Request for Comments 3330, Special-Use IPv4 Addresses." Internet Assigned Numbers Authority (IANA), Sep 2002. URL: http://www.rfc-editor.org/rfc/rfc3330.txt. (27 Mar 2003).

[29] Farrow, Rik, "ICMP Stands For Trouble." NetworkMagazine.com, 5 Sep 2000. URL: http://www.networkmagazine.com/article/NMG20000829S0003. (27 Mar 2003).

[30] Campione, Jeff, et.al. "The Twenty Most Critical Internet Security Vulnerabilities (Updated) ~ The Experts' Consensus." The SANS Institute, 3 Mar 2003. URL: http://www.sans.org/top20/. (31 Mar 2003).

[31] Dibowitz, Phil. "IPFilter FAQ." 15 Jan 2003. URL: http://home.earthlink.net/~jaymzh666/ipf/index.html. (22 Mar 2003).

[32] Sandoz, Jim. "Re: is return-rst broken?" IPFilter Mailing List Archive, 22 Nov 2000. URL: http://false.net/ipfilter/2000_11/0343.html. (30 Mar 2003).

[33] Insecure.org. "Nmap -- Free Stealth Port Scanner For Network Exploration & Security Audits." URL: http://www.insecure.org/nmap/. (31 Mar 2003).

[34] Zwamborn, Damian. "An Introduction to SSH Secure Shell." SANS Info Sec Reading Room, May 15, 2001. URL: http://www.sans.org/rr/encryption/intro_SSH.php. (31 Mar 2003).

[35] Baker, Bob. "Portrait of the Con Artist as a Young Man." Los Angeles Times, 6 Dec 2002.

[36] Read, Nick. "GIAC Certified Firewall Analyst (GCFW) Practical Assignment v. 1.7." Jan 2002. URL: http://www.giac.org/practical/GCFW/Nick_Read_GCFW.pdf. (15 Mar 2003).

[37] NetScreen Technologies Inc. "NetScreen Security Alert 51897." 25 Nov 2002. URL: http://www.netscreen.com/support/alerts/Predictable_TCP_Initial_Sequence_Numbers.html. (10 Mar 2003).

[38] CERT® Coordination Center. "CERT® Advisory CA-2001-09 Statistical Weaknesses in TCP/IP Initial Sequence Numbers." Carnegie Mellon Software Engineering Institute, 13 Sep 2002. URL: http://www.cert.org/advisories/CA-2001-09.html. (17 Mar 2003).

[39] SecurityFocus. "SecurityFocus HOME Vulns Info: NetScreen ScreenOS Predictable Initial TCP Sequence Number Vulnerability." 25 Nov 2002. URL: http://www.securityfocus.com/bid/6249/info/. (10 Mar 2003).

[40] Krauz, Pavel. "Pavel Krauz's Home Page." URL: http://lin.fsid.cvut.cz/~kra/index.html#HUNT. (10 Mar 2003).

[41] Bhansali, Bhavin Bharat. "Man-In-the-Middle Attack - A Brief." SANS Info Sec Reading Room, 16 Feb 2001. URL: http://www.sans.org/rr/threats/middle.htm. (10 Mar 2003).

[42] Thorsberg, Frank. "Does Your PC Harbor Zombies Waiting to Attack?" PCWorld.com, 15 May 2001. URL: http://www.pcworld.com/resource/printable/article/0,aid,50084,00.asp. (17 Mar 2003).

[43] Dittrich, David. "The 'Tribe Flood Network' distributed denial of service attack tool." University of Washington, 21 Oct 1999. URL: http://staff.washington.edu/dittrich/misc/tfn.analysis.txt. (17 Mar 2003).

[44] Hansen, Evan. "New Web attack tools exploit chat technology." CNET News.com, 4 Sep 2000. URL: http://news.com.com/2100-1023-245311.html?legacy=cnet. (17 Mar 2003).

[45] The HoneyNet Project. "The Reverse Challenge." 7 Jul 2002. URL: http://project.honeynet.org/reverse/. (18 Mar 2003).

[46] Mendel, Dion. "HoneyPot University Security Advisory 2002-001 - (summary)." URL: http://project.honeynet.org/reverse/results/sol/sol-06/summary.html. (18 Mar 2003).

[47] Tate, Steve, Vandana Gunupudi, Sandeep Nijsure, and Sachin Joglekar. "Answers to Challenge Questions." Computer Privacy and Security Lab, University of North Texas. 31 May 2002. URL: http://project.honeynet.org/reverse/results/sol/sol-21/answers.html. (18 Mar 2003).

[48] CERT® Coordination Center. "CERT® Advisory CA-2003-09 Buffer Overflow in Microsoft IIS 5.0." Carnegie Mellon Software Engineering Institute, 17 Mar 2003. URL: http://www.cert.org/advisories/CA-2003-09.html. (18 Mar 2003).

[49] SecurityTracker.com. "Microsoft IIS Web Server WebDAV Buffer Overflow Lets Remote Users Execute Arbitrary Code." 18 Mar 2003. URL: http://www.securitytracker.com/alerts/2003/Mar/1006305.html. (19 Mar 2003).

[50] Goland, Y., E. Whitehead, A. Faizi, S. Carter, and D. Jensen. "Request for Comments 2518, HTTP Extensions for Distributed Authoring -- WEBDAV " The Internet Engineering Task Force, Feb 1999. URL: http://www.ietf.org/rfc/rfc2518.txt. (19 Mar 2003).

[51] SecurityFocus.com. "Microsoft Windows 2000 ntdll.dll WebDAV Interface Buffer Overflow Vulnerability." 19 Mar 2003. URL: http://www.securityfocus.com/bid/7116/exploit/. (19 Mar 2003).

[52] Mixter. "Writing Buffer Overflow Exploits - a Tutorial for Beginners." Beyond Security Ltd, 4 Oct 2002. URL: http://www.securiteam.com/securityreviews/5OP0B006UQ.html. (19 Mar 2003).

[53] Hill, Brian. "Generating a HTTP post request." 17 Jun 2000. URL: http://www.omnigroup.com/mailman/archive/webobjects/2000-June/006512.html. (19 Mar 2003).

[54] Microsoft Corporation. "Windows 2000 Security Patch: IIS Remote Exploit from ntdll.dll Vulnerability." 17 Mar 2003. URL: http://microsoft.com/downloads/details.aspx?FamilyId=C9A38D45-5145-4844-B62E-C69D32AC929B&displaylang=en. (19 Mar 2003).

[55] Leininger, Hank. "The Mailing list ARChives." URL: http://marc.theaimsgroup.com/?l=ipfilter. (22 Mar 2003).

[56] Newman, David F. "Precompiled Solaris binaries of IPFilter." Newman Consulting, 12 Dec 2002. URL: http://www.maraudingpirates.org/ipfilter/. (22 Mar 2003).

[57] Network Solutions. "WHOIS Search Results." 22 Mar 2003. URL: http://www.networksolutions.com/cgi-bin/whois/whois?STRING=maraudingpirates.org&SearchType=do. (22 Mar 2003).