Global Information Assurance Certification Paper

SANS GCFW Practical Assignment
v 2.0

GIAC Enterprises: Your Fortune is Secure

By: Ben Nelson
August 25, 2003

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Abstract:

GIAC Enterprises is a company that generates the majority of its revenue from the sale of online fortunes. These fortunes are much like those found inside the Chinese fortune cookies that you get in a restaurant. This paper describes the network infrastructure needed to run the day to day operations at GIAC Enterprises.

Our description will include a basic network overview, as well as a detailed description of the network. The detailed description will include hardware and software configuration needed to support the network infrastructure in the most secure way that is feasible.

The network design incorporates features which allow the employees of GIAC Enterprises access to the resources needed to make the company as successful as possible. In conjunction with accessibility requirements of employees, the network design allows partners and suppliers access to the appropriate resources. All of this access is implemented in the most secure fashion that is economically feasible for the IT group at GIAC Enterprises.

1 Assignment 1: Security Architecture

1.1 Secure Network Design Principles used:

1.1.1 Least privilege:

Users will be given the minimal amount of privilege required to effectively interact on the GIAC network.

1.1.2 Defense in Depth:

Multiple layers of security will be used to protect company resources. Wherever possible, a minimum of two layers of security will be in place to protect each resource. An example of this would be in our mail system, where our external and internal mail servers are running different MTA software to prevent vulnerabilities in a single MTA from giving out the ' keys to the kingdom' .

1.1.3 Consistent machine builds:

All servers and workstations will be installed from an internal kick start server. Each class of machine will have a kick start template from which it is built. This will ensure that all servers start from the same blueprint.

1.1.4 Secure package management:

Since all servers will be running RedHat Linux, the RPM (RedHat Package Management) system will be used for software updates, where possible. In conjunction with RPM, RedHat' s up2date utility will be used to apply patches to production machines. Subscriptions to this service will be purchased from RedHat.

1.1.5 Regular 3rd party security assessments:

All systems and infrastructures will be audited quarterly by a 3rd party. Regular audits will be conducted by IT staff as well, but having a 3rd party conduct audits as well will give us the opportunity to catch potential problems that may have been missed by IT staff.

1.2 Access Requirements:

Access requirement definitions have been divided up into the different classes of users who will be in need of resources on the GIAC Enterprises network. Following is the list of users, their access requirements, and how we have met these access requirements within our network.

1.2.1 Customers:

Customers are companies or individuals who will be purchasing bulk online fortunes. This class of customer will require access only to our public web site and our secure ordering and delivery web site.

Access to the public web site will be over port 80 using the HTTP protocol to talk to our public web server. Access to the secure ordering and delivery web site will be over port 443 using the SSL protocol. SSL access will be configured to only allow SSL version 3, since deficiencies in prior versions of the SSL protocol have been found. All access to the secure ordering and delivery web site will be authenticated. Authentication will be handled by a web portal developed in-house. This web portal will authenticate users based off of customer information stored on our internal database server. Customer information stored on the internal database server must initially be entered by a member of the sales force or the executive team. From the web portal, a customer will be able to update their own contact information and download bulk fortunes that they have paid for.

1.2.2 Suppliers/Partners:

Suppliers are companies or individuals that supply GIAC Enterprises with our fortune cookie sayings. Partners are companies or individuals that translate and resell fortunes. Since suppliers will be providing the data that our business' revenue is based on, we must keep all communications with them secure. Also, since partners will be downloading the same data that suppliers will be uploading, communications with partners should also be secured.

Access will be granted for our Suppliers and Partners only to our Supplier/Partner secure web server. Communications with this server will use the SSL protocol over port 443.

Suppliers will upload new fortunes using a web interface designed in-house to track different suppliers and the fortunes that they upload. Authentication will be used to track supplier interaction with the web portal.

Partners will be given access into a different web interface which is also developed in house. This web interface will allow partners to download fortunes and will track the fortunes that have been downloaded by each partner. Authentication will be used to track partner interaction with the web portal.

1.2.3 Employees located on GIAC Enterprises internal network:

These are employees of GIAC Enterprises who are physically located at company headquarters. We will refer to this class of employee as ' internal employee' .

Internal employees will be provided with a RedHat Linux workstation. Workstation details will be provided later in this document.

Access by internal employees to the internet is limited to the following

protocols:

- HTTP - through the internal web proxy only
- HTTPS - through the internal web proxy only

HTTP and HTTPS access to the internal web proxy are provided so that web surfing traffic to the internet can be monitored and limited. Also, using a web proxy that only understands HTTP and HTTPS traffic will prevent users on the internal network from using other protocols over the standard HTTP and HTTPS ports (80 and 443 respectively).

Access to other internal services will be as follows:

- LDAPS : LDAPS access will be allowed through the internal firewall in order for users on the internal network to authenticate themselves to their workstations. Only SSL secured LDAP is allowed into the LDAP servers. This will help prevent the sniffing of authentication information on the internal network.
- SMTP : SMTP access is allowed into the internal mail server so that users can send email. The ability to send outbound email is essential to the business operations of GIAC Enterprises.
- IMAPS : IMAP access is allowed to the internal mail server over the SSL protocol so that users can check and manage their email.
- DNS : DNS access to the internal name server has been allowed in order for internal users to resolve host names both on the internal network as well as external public host names.
- NFS - NFS access will be allowed through the firewall to allow access to user home directories will be stored on an internal file server running NFS.
- HTTPS - HTTPS access will be allowed to the internal employee web server. This access will allow the sales and support force to manage customer records. SSL is used because access to this service will be authenticated and we don' t wantuser names and passwords passing in clear-text over the network.

1.2.4 Employees (mobile sales force and teleworkers):

These are employees of GIAC Enterprises who either travel a lot or tele-commute. These employees are rarely, if ever, physically located on the GIAC internal network. We will refer to this class of employee as ' remote employee' .

Remote employees will be provided with a RedHat Linux laptop. Laptop details will be provided later in this document.

Access to the GIAC network will be allowed only after establishing a VPN connection to our VPN concentrator. Once a connection to our VPN concentrator has been established, the following protocols can be used to

access the internet:

- HTTP - through the internal web proxy only
- HTTPS - through the internal web proxy only

Just as with the internal employees, HTTP and HTTPS access to the internal web proxy are provided so that web surfing traffic to the internet can be monitored and limited.

Access to other internal services will be as follows and follow the same reasoning as access provided for the internal employees:

- SMTP - to internal mail server
- IMAPS - to the internal mail server
- DNS - to the internal DNS server
- HTTPS - to internal employee web server

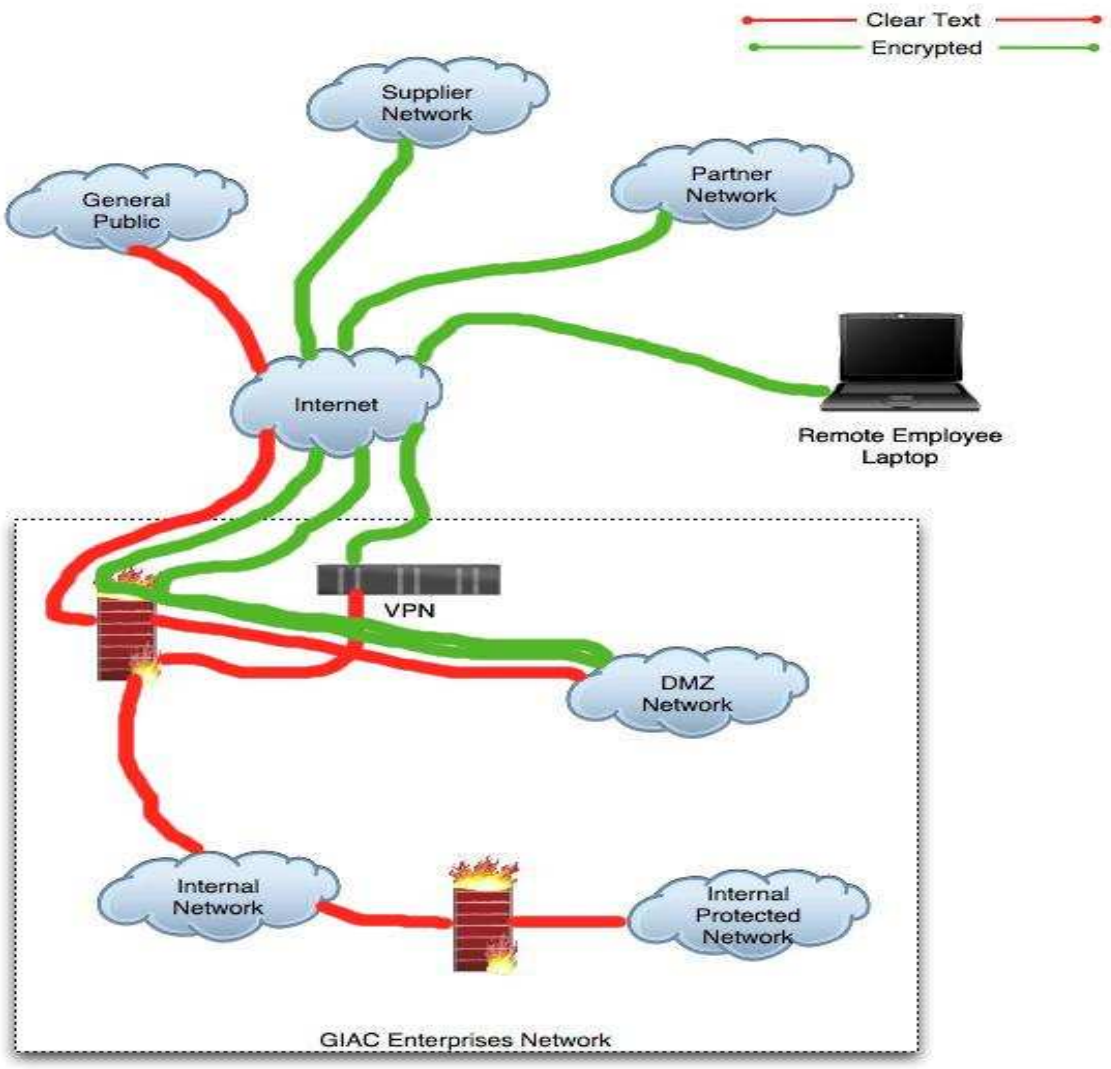
1.2.5 General Public:

This class of users includes everybody who does not fall into one of the above classes. Users in this class of people will be allowed access to the following services:

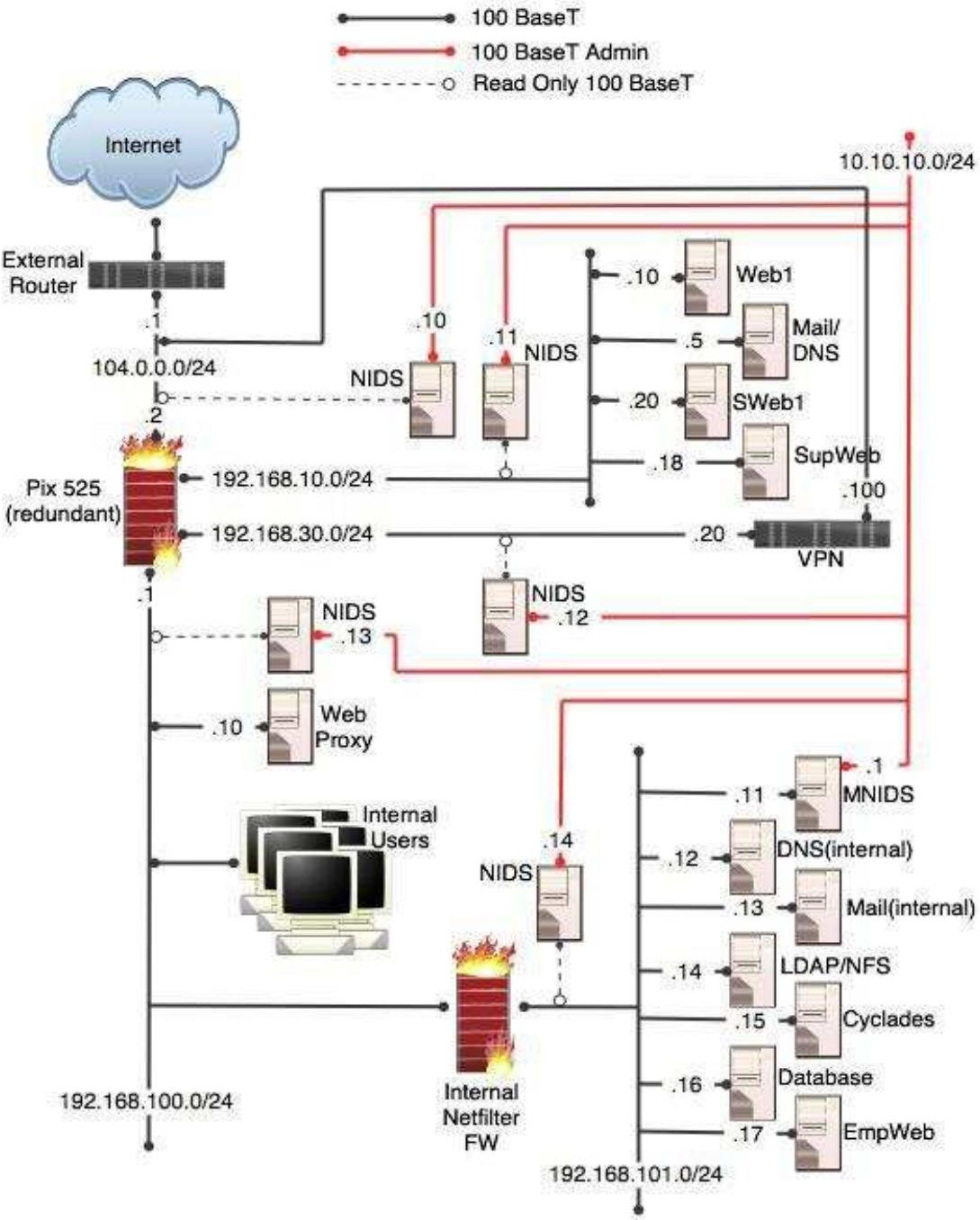
- HTTP - HTTP access will be allowed to the corporate web server so that potential new customers, partners and suppliers may learn about our company.
- SMTP - SMTP access will be allowed to the external mail server so that the delivery if internet mail to our employees.
- DNS - (tcp/udp) to the external DNS server so that we can answer name service queries for our external machines.

1.3 The GIAC Enterprises Network:

Data Flow:



Detailed network overview:



1.3.1 Network addressing:

The network is addressed internally using RFC 1918 address space. This addressing scheme was used to further obscure internal communications with the outside world. This addressing scheme also provides the IT group

with flexibility in assigning IP addresses to internal machines and networks.

All RFC 1918 addressed networks are addressed in the 192.168.0.0/16 address space (with one exception, which will be explained shortly). The third octet of each assigned network corresponds directly to the security level of the interface on the PIX firewalls which control its access to the network. For example, the internal interface of our PIX has a security level of 100, so the internal network is addressed using the 192.168.100.0/24 network. The networks in use are:

Address Space	***Description***
10.10.10.0/24	IDS back network
192.168.10.0/24	DMZ network
192.168.30.0/24	VPN network
192.168.100.0/24	Internal network
192.168.101.0/24	Internal Protected network
104.0.0.0/24	External network

You may have noticed that one of the networks listed does not follow IT' s standard addressing scheme. The 10.10.10.0/24 network is a network that is dedicated to the network of NIDS servers. A description of this network and the reasoning behind non-standard addressing is given below.

1.3.2 IDS Back Network (10.10.10.0/24):

The ' NIDS' network is used only by our Network Intrusion Detection Sensors for transmitting alert and log data to and from the master server. Currently, the NIDS network consists of 6 machines. 5 of these machines are NIDS sensors and 1 of them is a master server, which collects log information from each of the sensors and provides a centralized repository for alert and log data from the sensors.

The IP space 10.10.10.0/24 was used for this network for several reasons:

1. Any traffic for 10.10.10.0/24 seen anywhere but the NIDS network should be an immediate red flag. The fact that it uses a different numbering scheme will help administrators to pick this traffic out easily from ' normal' internal network traffic.
2. Using a different network further obfuscates network transmissions. A ' bad guy' will have to dig very deeply into the network to even know that this network address space is in use and that there is interesting information traversing it.

1.3.2.1 NIDS Sensors:

Each NIDS sensor is comprised of a minimal RedHat Linux installation

which is running the most current versions of Snort, SnortCenter client and Barnyard. As of this writing the most current versions of each of these is:

- Snort 2.0.1
- SnortCenter 1.0RC1
- MudPit 1.3

Each sensors Snort installation, including all rules and configurations, is managed via SnortCenter. From SnortCenter, each sensor has been configured to log both alert and log data using the Snort unified log format. If not using SnortCenter, this can be accomplished by using the following two configuration lines in the main Snort configuration file:

output log_unified: filename snort.log, limit 500
output alert_unified: filename snort.alert, limit 500

Both types of data are logged so that when an alert is generated, the maximum amount of available data is stored in a database for analysis. As you can see above, I have also asked that Snort rotate the log files once they reach a size of 500 MB. MudPit handles this just fine, and in fact will move the old log files out of the way for you. Once data is being logged using the above two directives, MudPit can be used to combine the data found in each of these files and generate alerts to insert into an SQL database.

Each NIDS sensor is connected to the production network using a ' phantom' network interface on a read-only monitor port. A phantom network interface is a network interface which does not have an IP address and therefore does not answer any traffic. It does, however, have the ability to listen to the network that it is attached to. A read-only switch monitor port is set up for each of these sensors.

1.3.2.2 NIDS Master Host:

The NIDS master host is another RedHat installation running Apache with PHP and MySQL. The SnortCenter management console is written in PHP and is running on this machine as well.

The SnortCenter management console gives us the ability to remotely manage the Snort rules and configurations on each of our NIDS sensors. SnortCenter allows us to write our own rules as well as download updated Snort rules right from the Snort website. All communications between the master NIDS host and the NIDS sensors is protected using SSL, which is a feature of SnortCenter.

It should be noted that all of the MySQL traffic generated from the NIDS sensors back to the NIDS master host is using the standard MySQL protocol, which normally would be in clear-text over the network. Version 4.1 of MySQL, which is due out by the end of this year, supports SSL transactions from the client to the server. When this functionality is

available, it will be implemented in our NIDS network. For the time being, we are using STunnel to secure our MySQL connections from the sensors to our master server.

1.3.3 The DMZ Network (192.168.10.0/24):

The DMZ network is where all of our externally accessible servers reside. While this network IS behind an interface of our firewall, it is considered an in-secure network due to the fact that the machines located in it are publicly accessible. No sensitive or proprietary data will be stored on any of these machines.

Access from the DMZ network out to the internet will be limited to established connections, DNS queries (port 53 using both TCP and UDP protocols) from the external mail server and SMTP connections from the mail server.

1.3.3.1 Web1 - public web server:

Our public web server is a RedHat Linux box running Apache 2.0.47 (footnote: most recent version as of this writing). For dynamic content, PHP version 4.3.2 (footnote: most recent version available as of this writing) has been installed as an Apache module. PHP has been compiled with MySQL support built in. All non-essential services have been turned off.

1.3.3.2 Sweb1 - secure public web server:

Our secure public web server is also running Apache version 2.0.47 with SSL support compiled in. PHP has also been included on this server as an Apache module. SSL ciphers have been limited only to 128-bit transactions. The sales and delivery application on this server has been written in-house. Since the application was written in house, we had it audited by a 3rd party to check for common vulnerabilities like buffer overflows, SQL injection possibilities and cross site scripting.

1.3.3.3 Mail1/DNS1 - public mail and dns:

Due to budget constraints, we decided to combine mail and DNS functionality onto one piece of hardware.

With regard to mail on this server, we are only using this server as a mail relay. This machine will host no mailboxes and contain no authentication information for any of our user accounts. The MTA we have chosen for this server is PostFix. The PostFix server will accept and relay mail to and from the GIAC Enterprises domain. Mail inbound will get forwarded on to the internal mail server. Mail coming from the internal mail server will be queued (if necessary) and then sent on to it' s final destination from this server.

DNS on this server will be handled by ISC's BIND. Zone transfers outbound will be limited to a single secondary DNS server provided by our ISP. BINDs internal access controls will be used to limit these zone transfers. Remaining transfers will be inbound and will come only from our internal DNS server. Also, the version string in BIND which can be found using the CHAOS class will be obfuscated, so that we are not making the version of BIND that we are running obvious to anyone (or any tool) that knows how to query this field. In the public whois records, this server will be listed as authoritative for the GIAC Enterprises domain. However, this machine will be set up as a slave DNS server, using the internal DNS server as its master. Recursive queries will be disabled except for those originating from other DMZ hosts.

1.3.3.4 Supplier/Partner Secure Web Server:

This is the web server that our Suppliers and Partners will use to upload and download fortunes. This server will also be running Apache version 2.0.47 with SSL support built in. This web portal will also allow them to manage their own contact information. Access to this server will be limited to SSL only, since our company's bread and butter', our fortunes, will be submitted and downloaded via the web portal hosted by this server.

1.3.4 VPN Network (192.168.30.0/24):

The primary purpose of the VPN network is to isolate connections from the VPN device into the GIAC Enterprises network. After VPN connections are established to the VPN device, traffic will be NAT'd to the 192.168.30.0/24 network. This will allow us to put access controls in place on our firewalls limiting traffic from these addresses.

1.3.4.1 VPN concentrator:

The VPN concentrator is a device that should be monitored very closely. The reason for this is that the closest thing resembling external access to our internal network will be coming from the VPN network. We have chosen to implement FreeS/WAN on a Linux box to handle our VPN needs.

1.3.5 Internal Network (192.168.100.0/24):

The internal network is where all of our internal users workstations will reside, as well as all of our internal servers. Internal servers will be protected by an internal firewall running NetFilter. Since this network is where all of our proprietary information is kept, it is kept the most secure.

1.3.5.1 Internal User Workstations:

All internal user workstations will be running RedHat Linux 9.0, which is the most recent version of RedHat available at the time of this writing. Each workstation will use LDAP over SSL to authenticate users against our internal LDAP server. All user home directories will be stored on an internal

NFS server, which will make backing up all user data much easier. No user will have ' root' access to their workstation and all package installation will be handled by IT administrators.

1.3.5.2 Internal NetFilter Firewall:

The internal NetFilter firewall protecting internal servers will be a RedHat Linux machine NetFilter to NAT connections to the Internal Protected network as well as implement access controls to protect the machines behind it.

1.3.5.3 Internal Web Proxy:

This machine will be the gateway for internal users to access web sites on the internet. It will be running Squid version 2.5, which is the most recent version as of this writing. The primary function of this web proxy is to make sure that all port 80 and port 443 transactions are actually using the HTTP protocol, and not some other protocol tunneled over these very common ports.

No access will be limited unless IT administrators start seeing abuse of the corporate network by visiting in-appropriate or non work-related web sites. All web transactions will be logged and the logs will be reviewed regularly for abuse.

Also, caching will not be performed on the internal web proxy unless deemed necessary due to bandwidth utilization issues.

1.3.6 Internal Protected Network (192.168.101.0/24):

This network is where the majority of our proprietary information is held. Machines on the Internal Protected Network should be considered the most volatile and therefore, protected more than any other nodes on the network.

1.3.6.1 Internal DNS Server:

The internal DNS server will contain both the internal and external DNS zones for GIAC Enterprises. ISC' s BIND version 9.2 will be used as the name service software on this machine. Only internal clients will be allowed to query the internal DNS zone. BIND' s built in access controls will be used to limit access to zone data based on the source IP address of the querying host. Recursive queries will be disabled except for those originating on the internal network.

1.3.6.2 Internal Mail Server:

The internal mail server will run Sendmail. The version of Sendmail installed will be RedHat' s patched version 8.12.8. The patches applied by RedHat back-port fixes to security vulnerabilities that have been found after the official release of 8.12.8. An IMAP daemon will also be configured on

this box to accept SSL connections. Procmail and SpamAssassin will be used to filter and quarantine SPAM messages and messages containing viruses.

1.3.6.3 Internal LDAP/NFS Server:

An LDAP server will be configured using OpenLDAP. The LDAP service will be used to store user account information for all of GIAC Enterprises employees. Access to this service will only be allowed over the SSL protocol, so that transmission of account information does not happen in clear text over the network.

This machine will also be running an NFS server. NFS will be used so that users can mount their home directories over NFS. Storing all user home directories in one place will make backups much easier. To make access through the firewall easier, mountd (which normally is assigned a port dynamically by the port mapper) has been assigned a static port of 1039.

1.3.6.4 Cyclades Terminal Server:

A serial terminal server will be used to connect to the serial console of our network devices such as firewalls, routers and switches. The Cyclades product will be configured to allow authenticated terminal access to our network devices over the SSH protocol.

1.3.6.5 Internal Database Server:

Our internal database server will contain the majority of our company' s proprietary and sales database information. The database system that we decided to use is MySQL version 4.0.14, which is the most recent version as of this writing. Access to this server will be limited to the secure external web server, the secure supplier web server and the secure employee web server. The data stored in this database will be accessed primarily using the in house designed web portals.

1.3.6.6 Internal Employee Web Server:

This web server will be used for employees to manage customer information. The server will be running Apache version 2.0.47 with SSL built in. Access to this server will only be allowed using the SSL protocol, customer contact and billing information will be accessed using the web portal.

1.3.7 External Network (104.0.0.0/24):

The external network consists only of our Cisco router and Cisco PIX firewalls. There is an IDS sensor listening on a monitor port between the router and firewall. This IDS will help us determine whether or not our routers access lists are doing their job.

1.3.7.1 External Router:

The external router we have chosen is a Cisco model 2691 model. This model has more than enough capacity for our current bandwidth utilization, but will allow us to grow our bandwidth requirements as the company grows.

1.3.7.2 External Firewalls:

The firewall we have chosen to use externally is a pair of Cisco PIX 525 redundant firewalls. Licensing for this pair of firewalls will be unrestricted so that we can add network interfaces without worrying about licensing restrictions. We will also include licensing for 3DES, to make any VPN connections that become necessary as secure as possible.

1.3.7.3 Remote Employee Laptops:

All remote employees will be given a laptop computer running RedHat Linux version 9.0, which is the most recent version available as of this writing. All laptops will have basic NetFilter firewalls in place allowing only critical communications in and out. Each firewall will use the FreeS/WAN implementation of IPSEC to connect to the GIAC Enterprises VPN device. Configurations for the VPN will be given in section 2 along with the main VPN security policy.

2 Assignment 2: Security Policy and Tutorial

2.1 Border Router:

The border router chosen for GIAC Enterprises is the Cisco model 2691 router. This router was chosen because it more than meets our current performance requirements, allowing room for growth. Our router is running IOS version 12.2(15).

The configuration descriptions included below for our router do not comprise the entire router configuration. Only the security and access related portions of the configuration will be included in this section. For the entire router configuration, see the Router Configuration section of the appendix.

The router is our first line of defense against malicious traffic going into our network. The router will do basic filtering based on IP addresses and ports, leaving the complex filtering to our stateful and packet inspection firewalls.

All services on the router that may accept connections will be disabled to prevent a possible compromise of the router. Administration will be performed over a serial connection from our Cyclades serial concentrator in the internal protected network.

GIAC Enterprises is located in a business park where our internet service is delivered to us over ethernet. So our router has 2 ethernet interfaces in it.

This section only contains portions of the routers entire configuration. See the appendix for the router configuration in its entirety.

2.1.1 Setting IP addresses:

Our ISP has provided us with a very small subnet for the outside interface of the router. This IP subnet is 105.0.0.0/29. As documented throughout this paper, our primary network is 104.0.0.0/24. The section of our router configuration which sets the IP addresses of each interface is:

```
interface FastEthernet 0/0
 no shutdown
 description connected to Internet
 ip address 105.0.0.1 255.255.255.248
 ip access-group 3 in
 keepalive 10
!
interface FastEthernet 0/1
 no shutdown
 description connected to EthernetLAN
 ip address 104.0.0.1 255.255.255.0
 ip access-group 6 in
```

```
 keepalive 10
!
```
We have included an access list for each of the interfaces as well. This is included using the ' access-group' command.

2.1.2 Access Lists:

Our access lists are very minimal, only filtering out very obviously bad traffic. We are relying on our firewall to do the majority of the packet filtering. We have implemented 3 access lists, 2 of which are identical. We used 2 separate access lists for the ingress and egress filters because if we ever want the ingress and egress access lists to diverge, we won' t have to re-configure each interface. The 3rd access list is a simple access list basically denying access to log into the router over the network. Our 3 access lists are:

```
access-list 1 permit 127.0.0.1 0.0.0.0
access-list 3 deny 10.0.0.0 0.255.255.255
access-list 3 deny 172.16.0.0 0.15.255.255
access-list 3 deny 192.168.0.0 0.0.255.255
access-list 3 deny 224.0.0.0 31.255.255.255
access-list 3 deny 127.0.0.0 0.255.255.255
access-list 3 permit any
access-list 6 deny 10.0.0.0 0.255.255.255
access-list 6 deny 172.16.0.0 0.15.255.255
access-list 6 deny 192.168.0.0 0.0.255.255
access-list 6 deny 224.0.0.0 31.255.255.255
access-list 6 deny 127.0.0.0 0.255.255.255
access-list 6 permit any
```

As stated, the above two access lists only filter out traffic that should never be routed on a public network.

2.1.3 Administrative Logins:

All logins to the router should come from the serial interface. Therefore, we have created a simple access list that includes only the localhost address and have limited access to vty0 to this list. We have also used random alpha-numeric strings as our passwords for logging into either the console or vty0. Here are the pertinent configuration sections:

```
line console 0
 exec-timeout 0 0
 password W5smdy5R
 login
!
line vty 0 4
 access-class 1 in
 password 9Jd9KWkY
 login
```

2.1.4 Disable un-needed services and functionality:

We have disabled many of the services that the router can offer over IP. We have also disabled certain classes of traffic flow through the router. These steps are all taken in an effort to harden the router against attack.

no service finger
no service tcp-small-servers
no service udp-small-servers
no ip name-server
no ip source-route
no ip finger
no ip domain-lookup
no ip http server
no ip bootp
no ip direct-broadcast
no ip unreachables
no snmp
no snmp-server location
no snmp-server contact

2.2 Primary Firewall Configuration:

Our primary firewall is a redundant pair of Cisco Pix 525 firewalls. The Pix 525 has far more capacity than GIAC Enterprises requires right now, but will give us the room that we need for growth. Also, since the Pix 525 will be used to terminate our major VPN connections, the extra processing power and throughput possibilities will be nice to have. We are running version 6.2(2) of the PIX software.

The configuration lines provided below are only those which diverge from the default Pix configuration. A full copy of our Pix configuration is provided in the appendix.

2.2.1 Interface security levels:

We have 6 available interfaces in our firewall, but we will only need to use 4 of them initially. Our first 4 interfaces will be used for an outside interface, a dmz interface, an inside interface and an interface for stateful failover; each with their associated security level.

nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 dmz security10
nameif ethernet3 vpn security30
nameif ethernet4 intf4 security25
nameif ethernet5 sfail security15

2.2.2 Interface IP address definitions:

Here, we' ll provide the IP addresses of the interfaces in the firewall. We will also

provide definitions for the interfaces of the redundant firewall.

2.2.2.1 Primary Pix IP addresses:

Here, we define the IP addresses that the primary Pix unit will use, as well as their corresponding subnet masks.

ip address outside 104.0.0.2 255.255.255.0
ip address inside 192.168.100.1 255.255.255.0
ip address dmz 192.168.10.1 255.255.255.0
ip address vpn 192.168.30.1 255.255.248.0
ip address intf4 127.0.0.1 255.255.255.255
ip address sfail 10.0.0.1 255.255.255.252

2.2.2.2 Failover Pix IP addresses:

Here, we define the IP addresses that the standby Pix unit will use while waiting for the primary unit to fail or be taken offline so that it can take over.

failover ip address outside 104.0.0.3
failover ip address inside 192.168.100.2
failover ip address dmz 192.168.10.2
failover ip address vpn 192.168.30.2
failover ip address intf4 0.0.0.0
failover ip address sfail 10.0.0.2

2.2.3 Static IP translations:

The following static NAT IP translations are used to map our external address space to our multiple internal IP space allocations. We only need 3 static translations for our network. These three translations are needed to provide external access to machines in our DMZ.

static (dmz,outside) 104.0.0.10 192.168.10.10 netmask 255.255.255.255
static (dmz,outside) 104.0.0.15 192.168.10.5 netmask 255.255.255.255
static (dmz,outside) 104.0.0.18 192.168.10.18 netmask 255.255.255.255
static (dmz,outside) 104.0.0.20 192.168.10.20 netmask 255.255.255.255

2.2.4 Access Lists:

Access lists provide the means by which the Pix can determine which IP addresses are allowed to connect to other IP address/Port number combinations. We have several batches of access lists providing access into the different interfaces in the firewall. At the end of each access list definition, you' ll see an access group command binding the access list to an interface.

2.2.4.1 Outside In:

access-list outside_in permit icmp any any echo-reply
access-list outside_in permit icmp any any unreachable
access-list outside_in deny ip 10.0.0.0 255.0.0.0 any
access-list outside_in deny ip 172.16.0.0 255.240.0.0 any
access-list outside_in deny ip 192.168.0.0 255.255.0.0 any

access-list outside_in permit tcp any host 104.0.0.10 eq www
access-list outside_in permit tcp any host 104.0.0.15 eq 25
access-list outside_in permit tcp any host 104.0.0.15 eq 53
access-list outside_in permit udp any host 104.0.0.15 eq 53
access-list outside_in permit tcp any host 104.0.0.20 eq https
access-group outside_in in interface outside

The first 2 lines of our outside_in access list permit echo replys as well as destination unreachable ICMP messages. This is so that internal hosts can receive responses to ICMP echo requests and ICMP Type 3 Code 4, which is ' Destination Unreachable, fragmentation needed' . Allowing ICMP Type 3 Code 4 prevents path MTU discovery black holes.

The next 3 lines drop all traffic from RFC 1918 address space, which should never exist on our outside network. Our external router should have dropped this traffic already, but we are practicing defense in depth.

The next 5 lines allow access to the services outlined earlier in this document on each of our DMZ servers.

The next 3 lines allow ESP, AH and IKE traffic to reach our VPN concentrator. ESP (Encapsulating Security Payload) protocol uses IP protocol number 50. AH (Authentication Header) protocol uses IP protocol number 51. IKE (Internet Key Exchange) traffic uses UDP port 500.

Lastly, there are no rules allowing connections into our Supplier/Partner secure web server. These will be added as needed, but will look like this (presuming that our partner is at IP address 105.0.0.1):
access-list outside_in permit tcp host 105.0.0.1 host 104.0.0.18 eq https

2.2.4.2 DMZ In:

access-list dmz_in permit icmp any any echo-reply
access-list dmz_in permit icmp any any unreachable
access-list dmz_in permit tcp host 192.168.10.20 host 192.168.100.16 eq 3306
access-list dmz_in permit tcp host 192.168.10.18 host 192.168.100.16 eq 3306
access-list dmz_in permit tcp host 192.168.10.5 host 192.168.100.13 eq 25
access-list dmz_in permit tcp host 192.168.10.5 any eq 53
access-list dmz_in permit udp host 192.168.10.5 any eq 53
access-group dmz_in in interface dmz

The first 2 lines of our dmz_in access list permit hosts on the DMZ network to respond to echo requests as well as fragment packets if requested.

The next 2 lines permits our secure DMZ web server access to the internal

MySQL database server, as well as our secure Supplier/Partner web server. This is necessary so that our customers can have access to their account data via the web portal hosted on our secure external web server and our suppliers and partners can have access to their web portal.

The next line permits our external mail server access to the SMTP port on our internal mail host. This is necessary so that mail received by our external mail relay can be relayed in to the internal main mail server.

The last two lines of the access list permit our external DNS server access to go out and query external name servers on the internet. This will be necessary in order to do recursive queries requested by other DMZ machines.

2.2.4.3 VPN In:

access-list vpn_in permit icmp any any echo-reply
access-list vpn_in permit icmp any any unreachable
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.10 eq www
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.10 eq https
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.13 eq 25
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.13 eq 993
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.12 eq 53
access-list vpn_in permit udp 192.168.30.0 255.255.255.0 host 192.168.100.12 eq 53
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.17 eq https
access-group vpn_in in interface vpn

The first 2 lines permit hosts on the VPN network to respond to echo requests and to fragment packets if requested.

The final 7 lines of the access list permit remote employees access to the resources that they need in order to to business remotely.

2.2.4.4 Inside In:

access-list inside_in permit icmp any any echo-request
access-list inside_in permit tcp host 192.168.100.10 any eq 80
access-list inside_in permit tcp host 192.168.100.10 any eq 443
access-list inside_in permit tcp host 192.168.100.12 any eq 53
access-list inside_in permit udp host 192.168.100.12 any eq 53

access-group inside_in in interface inside

The first line of the inside_in access list allows all internal hosts to make ping requests to anywhere. This is a useful diagnostic functionality to have available for all internal hosts.

The next 2 lines of the access list allow our web proxy to make HTTP and HTTPS connections out to anywhere. Since this is the only way for internal users to access web sites on the internet, this is necessary.

The last 2 lines allow our internal DNS server to do name service queries out to the internet. This is necessary so that internal hosts (mostly the web proxy) can perform recursive queries. As noted in the DNS server section recursive queries will be limited to those originating on the internal network.

2.2.5 Nat and Global definitions:

In this section, we will define a nat as well as a global for traffic from our internal network. This will allow internal hosts the ability to ping outside hosts for troubleshooting. Also, all traffic from our internal web proxy will appear to originate from this address.

nat (inside) 1 0.0.0.0 0.0.0.0 0 0
global (outside) 1 104.0.0.4 netmask 255.255.255.0

2.3 VPN Configuration:

Our VPN configuration consists of a Linux machine running the latest release of FreeS/WAN, an open source IPSEC package designed for Linux. The decision was made to use this package, in conjunction with NetFilter to handle our VPN and NAT duties.

One of the really neat features of FreeS/WAN is that it gives you the ability to run an external program when a new client or gateway establishes a VPN tunnel to the FreeS/WAN box, or tears one down. In our case, we will use this capability to insert and delete rules from our NetFilter NAT table when connections are established, or broken.

Included below are the configuration file /etc/ipsec.conf from our VPN server, as well as a sample /etc/ipsec.conf file from one of our remote employees' laptops. The policy uses an RSA key to authenticate and encrypt data between the clients and the server, rather than a pre-shared key. Pre-shared keys, due to their size, are far less secure than using an RSA key, which is why we chose to do it this way.

2.3.1 VPN Server Policy:

version 2.0 # conforms to second version of ipsec.conf specification

```
# basic configuration
config setup

# Add connections here.
conn remote-employee
   left=104.0.0.100
   leftsubnet=192.168.100.0/24
   leftid=@vpn1.giac.com
   leftupdown="ipsec remote-employees"
   leftrsasigkey=0sAQOUx/OT+/f82EEpMZp4QQ87
   gfpcAwx3HotSZr0dLwyPu0qhAYGdHTRhhbDQU
   wJ41ptEXmCxIDbac/8EMj5ruHjGHBrRe1FjbcgU
   0TGbKzyc4fxxkfKEHgwc8BkvYpcF8KXTM0GABa
   FL8FSTklLNsGWYbFrnLNQ7TfZPUV4lPxBxm17i
   EflmAd3vEn3DsY7OrmBn5Wv0fg5RK3eQDtDRp
   DjeDSj/cZRQAdTBt8cmj5VDAEADUFK72QKVCn
   PNPZZcH01jYEiCTw8I56rcc4bEN7rVv/25NwCUj
   GkgGvtXL0cqIg4bl/mIWaH7Ho5Fvi0MdfKddVHzff
   TTIGlRv0CHQlpBDk8OgEjnfWNsiRZH91wfP321
   rightnexthop=%defaultroute
   right=%any
   rightid=@empl-laptop.giac.com
   rightrsasigkey=0sAQOJDu6+oDlSrj1KryqQSJ0W3
   rCrqzrXVfnqbslmkWdbDmvWnyvLCF9RE1arDu8P
   lsP3RF9rBoPJn5gePU1ZgiEXsicPBIGepX7sg1CJ
   m1+Q+lPvKtFXr+bNalhi4imtSYbvBlZcTTFmEyRG
   B06fOy1zAVRbmfdDPNmk3/trj75vypeYS0ILnFtmV
   5X38MlrpCZiQSA58mpORo8y2UdGT/LtqVoUJlIaq
   eS0faZ6xkhMRdDwjDGGcmAtsYBPKVi9pW8AKw
   gFrZL1Aij4fPIBaom5YCrZLNqci6pYVvAFLWqV9cK
   7nkNekJ5rP3Ufob0S8mBHxQDc47M9o79VxCSbOl
   Ct+xBgetAroNuOSGj82SOg7m/3
   auto=add
```

This policy is called ' remote-employee' and can only be accessed using the IP address 104.0.0.100. The network that it is protecting is 192.168.100.0/24, meaning it will only accept and tunnel connections that are destined for this IP subnet.

On the line that reads ' leftupdown=”ipsec remote-employees”' , we are defining a script to run whenever a connection is established using this policy. The script that we are running is simply a modified version of the default FreeS/WAN _updown script. Our modifications tell the ipsec server to add NAT translations to the firewall when a new tunnel is established, and remove them once the tunnel is disconnected. A full copy of the remote-employees script is included in the appendix. The specific changes that we made are:

```
up-client:)
        # connection to my client subnet coming up
        # If you are doing a custom version, firewall commands go here.
        /sbin/iptables -t nat -A POSTROUTING -s $PLUTO_PEER -o eth1 -j
SNAT --to-source 192.168.30.30
        ;;
down-client:)
        # connection to my client subnet going down
        # If you are doing a custom version, firewall commands go here.
        /sbin/iptables -t nat -D POSTROUTING -s $PLUTO_PEER -o eth1 -j
SNAT --to-source 192.168.30.30
        ;;
```

The changes above use environment variables that are passed into all scripts called by the connection daemon. ' PLUTO_PEER' is an environment variable that contains the IP address of the remote client establishing the tunnel to the VPN device. Having this variable available to us and having the ability to run this script gives us a lot of flexibility in what we' d like to to do with the packets that have come through the VPN tunnel.

2.3.2 VPN Server firewall script:

Since the VPN server will have one interface outside of the firewall, it is imperative that a firewall be protecting the outside interface of the firewall. The internal interface will be managed dynamically by the automatic tunnel-establishment script mentioned above.

In order for IPSEC to work, 3 things need to be allowed into the outside interface of the firewall. These three things are:

- IP protocol number 50, which is the ESP (Encapsulating Security Payload) protocol
- IP protocol number 51, which is the AH (Authentication Header) protocol
- UDP port 500, which is where IKE (Internet Key Exchange) protocol talks

The second item on the list is the reason that our VPN device must be outside of the corporate firewall. Authentication header authenticates packets based on the IP addresses contained within them, so when a NAT translation happens between two endpoints of a VPN using AH, the tunnel establishment fails, because the IP addresses have changed en route. It is possible to run a VPN without using AH, but not recommended.

The script we have used is included in its entirety in the appendix, but the parts pertaining to VPN establishment are included here:

```
iptables -A INPUT -p 50 -d ${OUTSIDE_ADDR} -j ACCEPT
iptables -A INPUT -p 51 -d ${OUTSIDE_ADDR} -j ACCEPT
iptables -A INPUT -p udp -d ${OUTSIDE_ADDR} --dport 500 -j ACCEPT
```

2.3.3 Remote Employee VPN configuration:

```
version 2.0     # conforms to second version of ipsec.conf specification

# basic configuration
config setup

# Add connections here.
conn giac-vpn
    left=%defaultroute
    leftid=@empl-laptop.giac.com
    leftrsasigkey=0sAQOJDu6+oDlSrj1KryqQSJ0W3
rCrqzrXVfnqbslmkWdbDmvWnyvLCF9RE1arDu8P
IsP3RF9rBoPJn5gePU1ZgiEXsicPBIGepX7sg1CJ
m1+Q+lPvKtFXr+bNalhi4imtSYbvBlZcTTFmEyRG
B06fOy1zAVRbmfdDPNmk3/trj75vypeYS0ILnFtmV
5X38MlrpCZiQSA58mpORo8y2UdGT/LtqVoUJIlaq
eS0faZ6xkhMRdDwjDGGcmAtsYBPKVi9pW8AKw
gFrZL1Aij4fPIBaom5YCrZLNqci6pYVvAFLWqV9cK
7nkNekJ5rP3Ufob0S8mBHxQDc47M9o79VxCSbOl
    Ct+xBgetAroNuOSGj82SOg7m/3
    right=104.0.0.100
    rightsubnet=192.168.100.0/24
    rightid=@vpn1.giac.com
    rightrsasigkey=0sAQOUx/OT+/f82EEpMZp4QQ87
gfpcAwx3HotSZr0dLwyPu0qhAYGdHTRhhbDQU
wJ41ptEXmCxIDbac/8EMj5ruHjGHBrRe1FjbcgU
0TGbKzyc4fxxkfKEHgwc8BkvYpcF8KXTM0GABa
FL8FSTklLNsGWYbFrnLNQ7TfZPUV4lPxBxm17i
EFlmAd3vEn3DsY7OrmBn5Wv0fg5RK3eQDtDRp
DjeDSj/cZRQAdTBt8cmj5VDAEADUFK72QKVCn
PNPZZcH01jYEiCTw8I56rcc4bEN7rVv/25NwCUj
GkgGvtXL0cqIg4bl/mIWaH7Ho5Fvi0MdfKddVHzff
    TTIGlRv0CHQlpBDk8OgEjnfWNsiRZH91wfP321
    auto=add
```

As you can see, the configuration file for the remote employee laptop VPN contains much of the same information as the server VPN configuration. We do not need to run any extra scripts on this machine after the tunnel completes, because FreeS/WAN will automatically add the necessary routes to the machine after the tunnel has been established.

2.4 Pix Configuration Tutorial:

This section will give a basic tutorial on how to implement our security policy on a Pix firewall. The Cisco Pix firewall is a firewall that is used very widely across the industry. Syntax for the Pix firewall can be very similar to IOS syntax, although it is important to note that the Pix does not run IOS.

2.4.1 Connecting to your Pix:

To initially connect to a Pix firewall, one must use a serial connection. The serial connection must be established using the following serial connection settings:

- 9600 baud
- 8 data bits
- 1 stop bits
- no parity

Once you make a connection, you will see a prompt similar to:

pix>

The ' >' character tells you that you are in un-privileged mode. In order to view configurations you' ll need to get into ' enable' mode. You can do this by typing enable at the ' >' prompt.

pix> enable
*Password: ************
pix#

Now that you are in enable mode, you' ll be able to at least view configurations and settings on the Pix. In order to make any changes, you' ll need to be in configuration mode. You can enter configuration mode by typing ' configure terminal' . This command tells the Pix that you' d like to make configuration changes from the terminal you are connected through. Once you enter configuration mode, you' ll see a prompt that looks something like this:

pix(config)#

Note the ' (config)' in the prompt. If, in any mode, you are unsure of what commands you have available to you, you can get a listing by typing the ' ?' character.

2.4.2 Basic Configuration Commands:

There are six basic commands that must be used in order to configure a Pix firewall:

- nameif
- interface
- ip address
- nat
- global
- route

So we need to gather a bit of information before we can complete the above 6 commands. We need to know what interfaces in the Pix we' ll be using as well as their associated security levels. Security levels in the Pix are assigned to each network interface. No two interfaces can have the same security level. The idea is that an interface with a higher security level should be allowed to talk to an

interface with a lower security level, but not vice-versa. The ' outside' interface is always security level ' 0' and the ' inside' interface is always security level ' 100' . Your remaining interfaces must be spread out somewhere in between the inside and outside interfaces. We also need to know the IP addresses of our individual Pix interfaces. If we are going to do any NAT at all from the ' inside' interface to the ' outside' , we need to know what IP, or range of IP' s, we' ll use to translate internal address to the outside IP space. Finally, we need to know what the default route on the outside interface will be.

2.4.2.1 The 'nameif' command:

For GIAC Enterprises, we need to use 5 of our 6 available interfaces. We already said that the inside and outside interfaces must have security levels of 100 and 0 respectively, so that leaves 3 interfaces to figure out. Usually, the next lowest security interface is the DMZ interface. This is because the machines on the DMZ network are generally accessible in some form or another via the public internet. We have chosen to use a security level of ' 10' for our DMZ interface.

The next lowest security level for us will be the ' sfail' interface. The ' sfail' interface is the interface that we' ll use to transmit stateful failover information from our primary Pix to our secondary. This is not something that has to be done unless you have a redundant pair of Pix firewalls. We have given the ' sfail' interface a security level of 15.

The last interface that we' re concerned about for GIAC enterprises is the ' vpn' interface. This is the interface where all traffic from the VPN device will be coming into the network. Since, we' re still a bit uncertain about the data coming in on this interface, we' ve given it a security level of ' 30' .

Interface	*Name*	*Security Level*
Ethernet0	Outside	0
Ethernet1	Inside	100
Ethernet2	Dmz	10
Ethernet3	Vpn	30
Ethernet5	Sfail	15

The basic syntax for the ' nameif' command is as follows:

nameif <hardware_id> <interface name> <security level>

The ' interface name' field can be any alpha-numeric that you wish that is less than 48 characters in length. By default, the outside interface is named ' outside' , the inside interface is named ' inside' and remaining interfaces are named ' intfN' where ' N' is 2 through 5. We accomplish configuring the interface names and security levels using the following commands:

nameif ethernet0 outside security0

nameif ethernet1 inside security100
nameif ethernet2 dmz security10
nameif ethernet3 vpn security30
nameif ethernet4 intf4 security25
nameif ethernet5 sfail security15

2.4.2.2 The ' interface' command:

The next command that we need to use is the ' interface' command. This command will tell the Pix which interfaces to turn on or off, and at which network speeds to turn them on at. All of our interfaces will be set to auto, allowing them to auto-negotiate their speed and duplex settings with the devices that they are connected to. The basic syntax for this command is:
interface <hardware_id> <speed> [shutdown]
So, in order to configure the interfaces in the GIAC Pix, the following commands were used:

interface ethernet0 auto
interface ethernet1 auto
interface ethernet2 auto
interface ethernet3 auto
interface ethernet4 auto shutdown
interface ethernet5 auto

2.4.2.3 The ' ip address' command:

Now that we' ve got our interfaces named, security levels set for each one and speed and duplex settings figured out; we need to give each interface an IP address. It is safe to give interfaces an IP address before configuring anything else ONLY if you have started with a default Pix configuration. The default Pix configuration will not pass any traffic until you tell it to. The ' ip address' command is what we' ll use to set IP addresses on each interface. The syntax for the ' ip address' command is as follows:

ip address <interface name> <IP address> [netmask]

We set the IP addresses on all of our interfaces using the following commands:

ip address outside 104.0.0.2 255.255.255.0
ip address inside 192.168.100.1 255.255.255.0
ip address dmz 192.168.10.1 255.255.255.0
ip address vpn 192.168.30.1 255.255.255.0
ip address intf4 127.0.0.1 255.255.255.255
ip address sfail 10.0.0.1 255.255.255.252

2.4.2.4 The ' nat' and ' global' commands:

We' ve got IP addresses all figured out now, so we need to use the ' nat' and ' global' commands to allow traffic to flow out of the network, to the outside world. This step may not be desirable depending on your network security policy. In the case of GIAC enterprises, we have an access list on the

inside interface that is limiting what traffic can get out of the network, so we have set up a generic nat translation for all ' inside' addresses. The general syntax for the ' nat' command is:

nat (<interface name>) <nat_id> <local IP> <netmask>

The nat ID is a unique identifier that will identify each nat ' pool' . This is so that you can have multiple pools of addresses for different applications and policies. Note that a nat ID of ' 0' is reserved and means ' do not translate' . So, if you are using non RFC 1918 address space on your inside interface, then ' nat 0' is your friend. The ' global' command goes hand-in-hand with the ' nat' command. The ' global' command tells the Pix what address(es) to translate a ' nat pool' to. The general syntax of the global command is as follows:

global (<interface name>) <nat id> <global IP> [netmask <netmask>]

The ' nat' and ' global' commands that we used to translate all inside traffic to our PAT (Port Address Translation) address of 104.0.0.4 is:

nat (inside) 1 0.0.0.0 0.0.0.0
global (outside) 1 104.0.0.4 netmask 255.255.255.0

2.4.2.5 The ' route' command:

Finally, we need to use the route command to specify our default external route. The general syntax for the ' route' command is:

route <interface name> <IP address> <netmask> <gateway IP>

So, to complete our basic setup for GIAC Enterprises, we used the following route command:

route outside 0.0.0.0 0.0.0.0 104.0.0.1

This command says to route everything (0.0.0.0) out the gateway at 104.0.0.1.

Once this step is complete you will have a firewall that will protect all internal hosts from any traffic at all and will allow internal hosts out onto the internet using a shared IP address of 104.0.0.4. This is less functionality, however, than most organizations are wanting out of their firewalls. For GIAC Enterprises, we still need to be able to:

- statically translate external IP addresses into IP addresses residing on the DMZ network.
- Use access lists to allow or disallow traffic going into each interface

2.4.3 Setting up static translations and access lists:

We now need to be able to set up some static IP address translations, as well as access lists for each interface on our firewall.

2.4.3.1 Using the ' static' command for static NAT translations:

The ' static' command is the command that you' ll use to create static network address translations. This command associates two IP addresses on two separate network interface with each other. So that when the

' external' address is used in a connection, the connecting program is actually connecting to a completely different IP address, but is none the wiser.

The basic syntax for the ' static' command is as follows:

static (<internal interface name>, <external interface name>) <external IP> <internal IP> netmask <netmask>

In the above syntax, <netmask> is usually 255.255.255.255, unless you are trying to nat multiple internal IP addresses to a single external address. In the case of GIAC Enterprises, all of our static NAT translations are one-to-one. The only machines that we need to have static NAT translations for are those machines in our DMZ network, which will be contacted by the outside world. So, we used the following commands to set up the static NAT translations for the GIAC firewall:

static (dmz,outside) 104.0.0.10 192.168.10.10 netmask 255.255.255.255
static (dmz,outside) 104.0.0.15 192.168.10.5 netmask 255.255.255.255
static (dmz,outside) 104.0.0.18 192.168.10.18 netmask 255.255.255.255
static (dmz,outside) 104.0.0.20 192.168.10.20 netmask 255.255.255.255
static (dmz,outside) 104.0.0.30 192.168.10.30 netmask 255.255.255.255

To use one of the above as an example; if an outside host were to contact 104.0.0.10, the Pix would translate that address from the ' outside' interface address of 104.0.0.10 to the ' dmz' interface address of 192.168.10.10 . This operation is completely transparent to the connecting host.

2.4.3.2 Using the ' access-list' command to control traffic:

Access lists allow us to group together a bunch of access rules into a sort of access policy (or ' list'). This group of rules is bound to an interface on the Pix and then all traffic going into that interface must match one of the rules, or it will be dropped. So, before you create any access lists, you' ll need to characterize the traffic that will be flowing into each interface on your Pix. In the case of GIAC Enterprises, we have already done this in section 1.2 of this document. We will briefly review those access requirements and implement them as Pix access lists.

The basic syntax of an ' access-list' command is:

access-list <acl name> <permit|deny> <protocol> <source address> <destination address> eq <port number>

There are many other available parameters to the ' access-list' command, but the ones just listed are the very basic parameters needed to implement the GIAC access lists. In the case of both <source address> and

<destination address>, a combination of an IP address and subnet mask can be used, or a single host can be defined using the ' host' keyword. If you only want to specify, for example, ' 10.10.10.10' , then the syntax using the ' host' keyword would be ' host 10.10.10.10' . You could also have specified this host using a an IP address and netmask like so: ' 10.10.10.10 255.255.255.255' .

One common mistake while using the ' access-list' command to create access lists, is forgetting to use the ' access-group' command to attach the access list to an interface. Writing an access list does nothing if it is not put into action using an ' access-group' command. The general syntax of ' access-group' is:

access-group <access list name> in interface <interface name>

Since this is a common thing to forget, every access list definition below will include the appropriate ' access-group' at the end.

Our first access list will be the access list bound to the ' outside' interface of the Pix. We have chosen to name this access list ' outside_in' . In addition to the access requirements set forth in section 1.2 of this document, we' ll allow some forms of ICMP traffic into the outside interface for diagnostics.

Our access list looks like the following:

access-list outside_in permit icmp any any echo-reply
access-list outside_in permit icmp any any unreachable
access-list outside_in deny ip 10.0.0.0 255.0.0.0 any
access-list outside_in deny ip 172.16.0.0 255.224.0.0 any
access-list outside_in deny ip 192.168.0.0 255.255.0.0 any
access-list outside_in permit tcp any host 104.0.0.10 eq www
access-list outside_in permit tcp any host 104.0.0.15 eq 25
access-list outside_in permit tcp any host 104.0.0.15 eq 53
access-list outside_in permit udp any host 104.0.0.15 eq 53
access-list outside_in permit tcp any host 104.0.0.20 eq https
access-group outside_in in interface outside

We have allowed echo-replys into the outside interface so that if a host inside the firewall makes an echo-request out, it will get a response back. We have allowed ICMP unreachable messages, because Code 4 of this Type of ICMP traffic tells hosts that ' fragmentation is needed' . This is important for clients connecting over connections that require smaller than usual fragments.

We also explicitly deny traffic from the 3 RFC 1918 address ranges. This traffic should never hit our firewall if our router is configured correctly, but it is good practice to make sure and drop this traffic.

Next we allow connections to the primary service associated with each

machine in our DMZ. We have allowed port 80 traffic ('www') to the web servers, port 443 traffic ('https') to each of the secure web servers, port 25 traffic ('smtp') to our mail servers, and finally port 53 traffic ('dns') to our external DNS host.

The next access list that we needed to implement was an access list to bind to the 'dmz' interface. This access list will look similar to the access list we just built for the 'outside' interface' but, will include source host requirements, which we did not have in our previous access list. Our 'dmz_in' access list looks like this:

access-list dmz_in permit icmp any any echo-reply
access-list dmz_in permit icmp any any unreachable
access-list dmz_in permit tcp host 192.168.10.20 host 192.168.100.16 eq 3306
access-list dmz_in permit tcp host 192.168.10.18 host 192.168.100.16 eq 3306
access-list dmz_in permit tcp host 192.168.10.5 host 192.168.100.13 eq 25
access-list dmz_in permit tcp host 192.168.10.5 any eq 53
access-list dmz_in permit udp host 192.168.10.5 any eq 53
access-group dmz_in in interface dmz

In this example, we have included the same ICMP traffic that we allowed into the outside interface, and for the same reasons.

On the 3rd, 4th and 5th lines you'll see access-list definitions that have a source host requirement to them in addition to the destination host requirement. We have used a source host requirement so that only specific hosts are allowed to connect to the services mentioned. The 3rd and 4th lines allow our two secure web servers access to pull data from our internal MySQL database. Since this may be sensitive data, the extra access restrictions are necessary. On the 5th line, we have asked that only the external mail server can contact the internal mail server. This is to help prevent bad mail from getting into our internal network.

The final two lines of our 'dmz' access list allow the external DNS server the ability to query external name servers on the internet for the few recursive queries that it should need to perform. We need to open up the DNS port using both the 'tcp' and 'udp' protocols, since both may be used for large query results and zone transfers.

Our remaining 2 access lists were created using the same train of thought and commands that the above two access lists have used.

3 Assignment 3: Verify the Primary Firewall Policy:

3.1 The Plan:

The first step to verifying our firewall policy is to come up with a plan of attack. (no pun intended) We need to figure out which tools we are going to use, the basic approach to auditing each access-list rule, when we' d like to do the verification, the cost associated with the whole process and the possible risks involved.

3.1.1 The overall approach:

When validating each of the access rules in our primary firewall, the easiest process is going to be:

- Place a sniffer on the destination interface. In many cases, this sniffer may already be in place, since we have deployed Snort Sensors at key points in our network.
- Scan important hosts on the destination interface using several different scanning modes.
- All scans performed will be non-invasive unless precautions have been put in place to eliminate data loss potential and minimize downtime associated with destructive probes.

So, an example of one complete interface validation would go something like this:

1. A machine is placed on the origination network using an IP address that is not supposed to be allocated to anything.
2. A sniffer is placed on the destination network listening for traffic from the bogus IP address.
3. A portscan of each of the important hosts on the destination network is performed from the machine set up in step 1

3.1.2 Cost and level of effort:

Using automated tools to do most of the auditing would be the ideal way to reduce cost in this operation. However, many automated tools produce too many false-positive alerts that must be filtered through. So, a high level verification of the primary firewall rules using some manual tools will require approximately 1 day of an IT administrators time. This is a fairly low time/cost commitment to do basic auditing that could save GIAC from a potentially harmful break-in allowed by firewall rules that have not been kept up to date. Therefore, once every other week, an IT administrator will audit the firewall rules and correct any discrepancies found.

3.1.3 Possible risks involved:

In the worst case scenario, a malformed packet could be sent to a host that was not supposed to receive the packet and an interruption of service is the result. For this reason, only basic scanning and probing will be done of each firewall

interface during business hours. Any further testing, including possible penetration tests must be performed after business hours and after verification of a successful backup.

In the event of a system failure, all efforts will be made to bring the system back online without having to go to our tape backups for a restoration. All system failures will be logged and investigated further on a test system after the primary system is brought back online.

3.2 The Validation:

3.2.1 Testing from the internet to our firewall:

3.2.1.1 Nmap Syn scan (nmap -sS 104.0.0.2) output:

All 1643 scanned ports on fw.giac.com (104.0.0.2) are: closed

3.2.1.2 Nmap Xmas scan (nmap -sX 104.0.0.2) output:

All 1643 scanned ports on fw.giac.com (104.0.0.2) are: closed

3.2.1.3 Nmap Fin scan (nmap -sF 104.0.0.2) output:

All 1643 scanned ports on fw.giac.com (104.0.0.2) are: closed

3.2.2 Testing from the internet to our DMZ hosts:

For this test example we will run scans against our public web server, which is likely to be the first target explored by someone trying to penetrate the GIAC network.

3.2.2.1 Nmap Syn scan (nmap -sS 104.0.0.10) output:

Interesting ports on web1.giac.ocom (104.0.0.10):
(The 1642 ports scanned but not shown below are in state: filtered)

Port	State	Service
80/tcp	open	http

3.2.2.2 Nmap Xmas scan (nmap -sX 104.0.0.10) output:

All 1643 scanned ports on web1.giac.com (104.0.0.10) are: closed

3.2.2.3 Nmap Fin scan (nmap -sF 104.0.0.10) output:

All 1643 scanned ports on web1.giac.com (104.0.0.10) are: closed

3.2.3 Testing from the DMZ network to internal hosts:

An example of testing from the DMZ network to the internal network will be scanning the database server using a source IP address that should not be able to connect to the database server on any port. As a control test, during off-hours a scan was conducted from one of the secure web servers which SHOULD have access to the database server on a single port.

3.2.3.1 Nmap Syn scan (nmap -sS 192.168.100.16) from good IP output:

Note: Host seems down. If it is really up, but blocking our ping probes, try -P0

3.2.3.2 Telnet probe to port 3306 from good IP:

```
 $ telnet 192.168.100.16 3306
Trying 192.168.100.16...
Connected to 192.168.100.16
Escape character is '  ^]'  .
4
4.0.14-standard-log<Qy.t~?Z,^]
telnet> q
Connection closed.
```

3.2.3.3 Nmap Syn scan (nmap -sS 192.168.100.16) from bad IP output:

Note: Host seems down. If it is really up, but blocking our ping probes, try -P0

3.2.3.4 Telnet probe to port 3306 from bad IP:

```
$ telnet 192.168.100.16 3306
Trying 192.168.100.16...
telnet: connect to address 192.168.100.16: Connection refused
```

Our nmap scans were failing because we are not allowing icmp echo-requests out of the dmz network. Nmap, by default, relies on ICMP to do a host alive check before it starts portscanning the host. So, I used a simple telnet connection test to verify that the important port in question was indeed blocked when accessed from an un-authorized IP address. NetCat (' nc') could have been used as well to test connectivity.

3.2.4 Testing from the VPN network to internal hosts:

Much like the tests in the previous section, all of our nmap scans will return a message stating that they think the ' Host seems down.' because we are not allowing ICMP echo-requests out of the VPN network. So we will simply to a series of telnet tests from good IP addresses.

3.2.4.1 Telnet probe to web proxy on port 80:

```
$ telnet 192.168.100.10 80
Trying 192.168.100.10...
Connected to 192.168.100.10
Escape character is '  ^]'  .
^]
telnet> q
Connection closed.
```

3.2.4.2 Telnet probe to web proxy on port 81:

```
$ telnet 192.168.100.10 81
Trying 192.168.100.10...
telnet: connect to address 192.168.100.10: Connection refused
```

3.2.4.3 Telnet probe to master NIDS box on port 22:

```
$ telnet 192.168.100.11 22
Trying 192.168.100.11...
telnet: connect to address 192.168.100.11: Connection refused
```

3.2.4.4 Telnet probe to internal mail server on port 25:

```
$ telnet 192.168.100.13 25
Trying 192.168.100.13...
Connected to 192.168.100.13.
Escape character is '  ^]'  .
220 mail.giac.org ESMTP Sendmail 8.12.8/8.12.8; Mon, 25 Aug 2003
07:06:19 -0600
QUIT
221 2.0.0 mail.giac.org closing connection
Connection closed by foreign host.
```

The examples above of trying arbitrary ports and hosts on the internal network is a great validation of our firewall rules. A complete validation would be much more exhaustive, but would contain a lot of repetitive information and therefore has been omitted from this paper.

3.3 Firewall validation evaluation of results:

Over all, the evaluation performed yielded very good results. Everything that we thought was closed to prying eyes was indeed closed. In doing this validation, though, it has become obvious that our ' dmz_in' access list may be a bit too permissive.

Our access list on the DMZ network allows all hosts on the network to make connections to internal hosts. Since our current VPN NAT configuration only NAT' s hosts to a single IP address in the 192.168.30.0 network, we could probably limit connections to just that 1 IP address. Should the need arise later on for a more permissive access list, then we can implement one at that time. We could accomplish this change by re-writing the following rules:

```
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host
192.168.100.10 eq www
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host
192.168.100.10 eq https
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host
192.168.100.13 eq 25
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host
192.168.100.13 eq 993
```

access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.12 eq 53
access-list vpn_in permit udp 192.168.30.0 255.255.255.0 host 192.168.100.12 eq 53
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.17 eq https

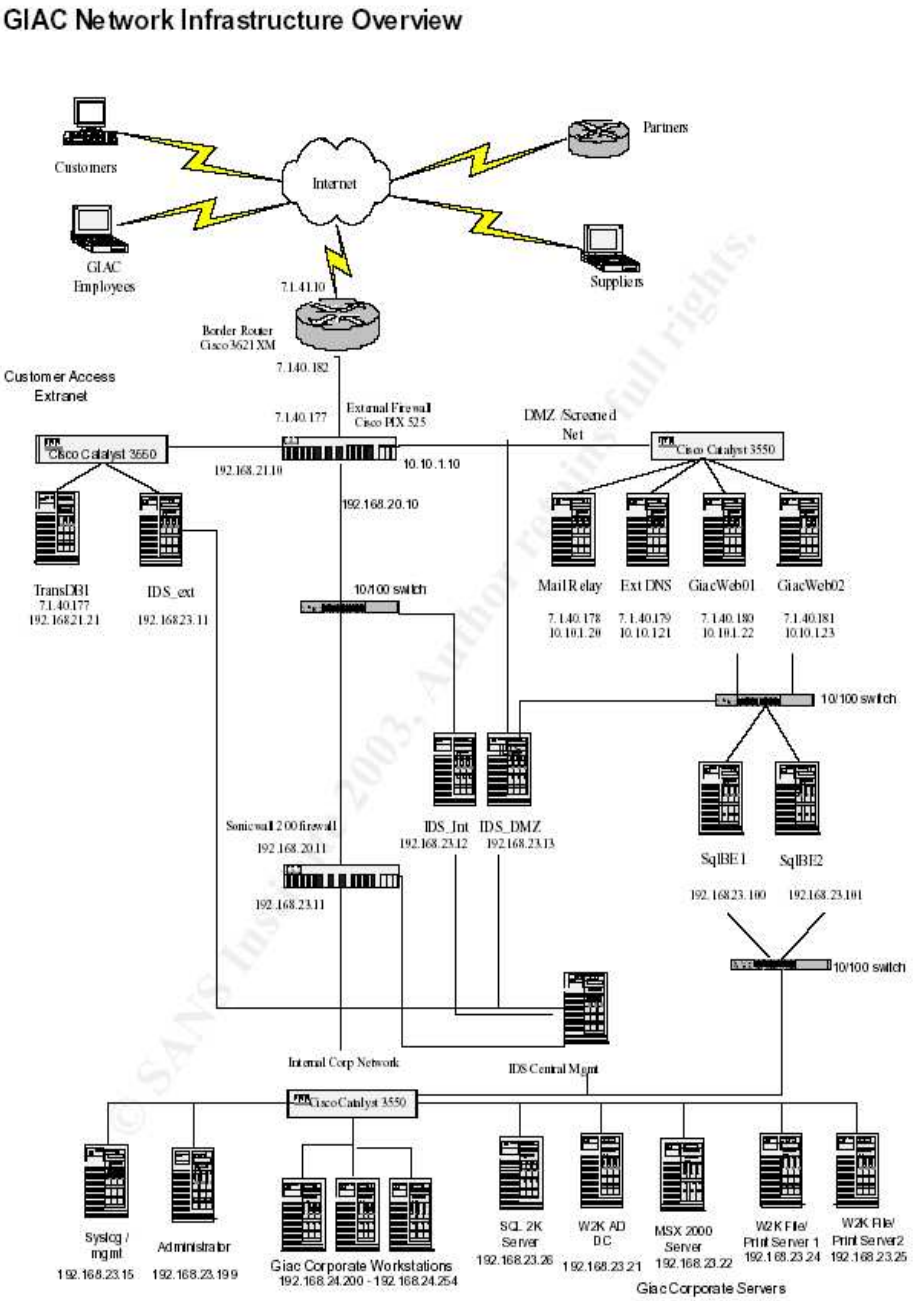
To look like the following:

access-list vpn_in permit icmp any any echo-reply
access-list vpn_in permit icmp any any unreachable
access-list vpn_in permit tcp host 192.168.30.30 host 192.168.100.10 eq www
access-list vpn_in permit tcp host 192.168.30.30 host 192.168.100.10 eq https
access-list vpn_in permit tcp host 192.168.30.30 host 192.168.100.13 eq 25
access-list vpn_in permit tcp host 192.168.30.30 host 192.168.100.13 eq 993
access-list vpn_in permit tcp host 192.168.30.30 host 192.168.100.12 eq 53
access-list vpn_in permit udp host 192.168.30.30 host 192.168.100.12 eq 53
access-list vpn_in permit tcp host 192.168.30.30 host 192.168.100.17 eq https

In preparing to make changes to the firewall, it also occurs to me that having some form of configuration version control would be helpful in debugging. So that if one of our firewall validations turns up something that has been opened up that should NOT have been, we can go back through different versions of the firewall configuration to see where and when the hole was introduced.

4 Assignment 4: Design Under Fire:

For the design under fire portion of this document, I have chosen Will Whitings network design. His full practical can be downloaded at: http://www.giac.org//practical/GCFW/Will_Whiting.pdf



4.1 Attack on the firewall:

Will has chosen to use a Cisco PIX 525 firewall for his network implementation. The version of PIX software running on his PIX is 6.2(2). Currently, there are no known vulnerabilities in this version of the PIX software. All software releases that have come out after 6.2(2) have been feature releases.

It also appears that no TCP services are listening on the firewall, so Will must be administrating his firewall over a serial connection only. Since it appears that there is no current way into the Pix, we will have to try to social engineer our way into the Pix. In order to do this, we' ll have to pose as the the head security engineer and try to trick one of his employees into allowing us access into the PIX and maybe hinting at the password.

We will first pose as a customer who would like to purchase fortunes from GIAC, and ask about delivery of fortunes. Since fortunes are delivered to customers via a web interface in Will' s design, we will pose concerns regarding uptime and availability of web systems so that we can download fortunes whatever time of day we like (we are a global business you know;). All of these concerns and questions are allowing us to map out the network architecture and determine when scheduled maintenance occurs.

Now, a couple of hours after the next scheduled maintenance, (which is late at night) I will call one of Will' s employees at home, waking them up. Posing as Will, I will tell them that since the maintenance we have seen some strange behavior from the firewall and would he/she please do the following:

- Go back into work and reboot the firewall.
- Call me back at my mother-in-laws number, since we are taking care of poor sick grandpa tonight (The number provided will be a number we have access to answer). That way I can check to see if our issue has gone away.
- If I give them the go ahead, they can head back home and go to sleep

When the employee gives me a call back, I' ll tell them that there is still something strange going on....weird packet loss issues. Could they please log into the serial console and enable SSH access on the external interface for the following IP address, which is the IP I' m coming from over my mother-in-law' s cable modem. Also, I forgot my PDA back at work, could you read me off the passwords as you type them in....it' s been a while since I had to log into this thing.

So, if all goes well, I now have full SSH access to the firewall. Since we perform this bit of social engineering late at night and after a maintenance window, the employee is less likely to be suspicious of my requests.

4.1.1 Countermeasures:

Several things could have mitigated this attack on the GIAC network. Training all employees who have any responsibility at all for security to be very paranoid is a

great start. Having policies in place that forbid things like giving out passwords over the phone to anybody is a good thing too.

Also, creating an atmosphere at work where suggestions and feedback are acceptable would help this situation as well. By this, I mean that if the employee felt threatened at all by Will or any of the senior administrators he would be less likely to 2nd guess them or question their reasons for doing things. Also, in a friendly atmosphere, some employees may know something about you personally, like your mother and father-in-law passed away last summer, so there is not a real likelihood of you being at your mother-in-laws house.

4.2 Distributed Denial of Service attack:

Since Will has decided to use Windows machines primarily on his network, the possibilities for internal Denial of Service are increased significantly. But, we are tasked with implementing a remote denial of service attack using compromised cable/dsl connected machines.

The first step in this process is deciding on a program to use and a course of action for implementing it. I have decided to use a recent DCOM exploit to get a shell on 50 or so DSL/Cable computers.

The first step in this process is to figure out which types of hosts I can easily compromise. Using the DCOM exploit written by H D Moore, I' ll be able to get a shell on boxes running the following versions of Windows:

- Windows 2000 SP0 (english)
- Windows 2000 SP1 (english)
- Windows 2000 SP2 (english)
- Windows 2000 SP3 (english)
- Windows 2000 SP4 (english)
- Windows XP SP0 (english)
- Windows XP SP1 (english)

Source code for this exploit can be found at:
http://packetstormsecurity.nl/0307-exploits/dcom.c

The next step in this process is to find some IP space to scan for vulnerable boxes. Road Runner is a large provider of cable modem service, so we' ll start there. This can be easily accomplished by:

1. doing a DNS lookup on www.rr.com, which returned an IP of: 24.30.203.14
2. plugging the IP we found in at ARIN: http://www.arin.net/whois/index.html
3. browsing through the results until we find the ' NetRange' section which will tell us what IP range is owned by Road Runner.

Now that we know where to scan, we need to know how to scan. We can use Nmap with TCP/IP fingerprinting turned on to scan remote hosts to make sure that DCOM is open and that the operating system in question is exploitable. The following command line from Nmap should do the trick:

nmap -sS -p 120-450 -O -oG - <ip-range>
We used the ' -oG' option to make the output more easily searchable for the appropriate operating systems.

Once we have a list of exploitable hosts, it is just a matter of running the exploit code provided in the URL above on each of them.

Now we have shell access to approximately 50 boxes. We need to use the command shell that we' ve got to down load a SYN flooder, which can be found at:
http://packetstormsecurity.org/DoS/Spastic.exe

Now we launch Spastic.exe on each of our hosts with a target of 7.1.40.177.

4.2.1 Countermeasures:
Since a group of 50 compromised DSL/Cable machines can generate plenty of traffic to overwhelm the external router which is a Cisco 3621XM. Having proper monitoring in place would help administrators determine the sources of the DOS and block them at the external router.

Also, coordinating with your upstream ISP to help block the malicious traffic could prove to be very useful as well.

One way to minimize the effect of a DOS attack such as this is to have an alternate internet connection available. Using an alternate internet connection would allow internal employees to continue to carry on business even though the main route to the internet is saturated with bogus traffic.

4.3 Attack plan to compromise an internal system:
We now need to formulate an attack to compromise an internal computer. Will has provided us with one obvious path into the internal network via the DMZ network. Both web servers are directly attached to a network that has two SQL servers on it. These two SQL servers are, in-turn, directly connected into the internal network.

So, if we are to take this path to get into the internal network, we' ll have to compromise at least one machine in the DMZ network (maybe two), one SQL server and one machine on the internal network. But really once we have accesss to the SQL server, we have un-restricted access to any of the internal machines.

Even if I did not know the internal network topology, one of the IIS servers would probably be the first target picked by most bad guys. Once the IIS server was exploited, it would only be a matter of time before it was discovered that the web server has two network connections, one of which leads right to a pair of SQL

servers, which have had their fair share of exploits in the last year.

We need to make sure that the machines in question are in fact IIS servers and to figure out what version of IIS that they are running. We can do this with a simple telnet probe:

l*ocalhost $ telnet 7.1.40.180 80*
Trying 7.1.40.180...
Connected to 7.1.40.180.
Escape character is '^]'.
HEAD / HTTP/1.1

HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: Tue, 24 Aug 2003 01:19:22 GMT
Connection: close
Content-Length: 4009
Content-Type: text/html

Connection closed by foreign host.

OK, now we know that the web servers are indeed IIS servers and they are running IIS v5.0. If we are lucky, the administrators will not have patched these machines against the WebDAV vulnerability detailed in the following advisories::
http://www.cert.org/advisories/CA-2003-09.html
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109

On the other hand, this vulnerability was discovered in March of this year, so hopefully a patch has been put in place, or the WebDAV services turned off on these web servers.

A quick test using the script found at:
http://packetstormsecurity.nl/0103-exploits/vv5.pl
will let us know if the servers are vulnerable. Since we' re bad guys, we don' t really care that the above script will actually restart all IIS services. It' ll give us a quick and dirty check to see if the machines are even vulnerable.

After running the script it turns out that the administrators of this network are more pro-active about patching their machines than we had hoped. Either that, or they are not using the WebDAV functionality of IIS 5.0. The web servers do not appear to be vulnerable to the WebDAV vulnerability. Which is good for them, because this could have been the gateway into their internal network for me.

4.3.1 Countermeasures:
The best counter measure to mitigate this type of risk is to be pro-active about applying vendor released patches to your system. Also, turning off un-needed

services will help in a situation like this. Simply having the WebDAV service turned off would prevent an attack from working on this vulnerability as well.

References:

Cisco Systems. Cisco PIX Firewall Software Cisco PIX Firewall Command Reference, Version 6.2. 12 Dec, 2002. URL:http://www.cisco.com/en/US/products/sw/secursw/ps2120/products_command_reference_chapter09186a0080104237.html (22 Mar. 2003)

Cisco Systems. Cisco Secure PIX Firewall Fundamentals version 1.11. USA: Cisco Systems, Inc, 2000. 5-5 - 5-15

Garfinkel, Simson; Spafford, Gene. Practical Unix & Internet Security 2nd Edition. Sebastopol: O' Reilly & Associates, Inc, 1996. 605-610, 637-668

Hunt, Craig. TCP/IP Network Administration, 2nd Edition. Sebastopol: O' Reilly & Associates, Inc, 1998. 25-33, 384-385, 396-402

Osipov, Vitaly; Sweeney, Mike; Weaver, Woody. Cisco Security Specialist' s Guide to PIX Firewalls. Rockland: Syngress Publishing, Inc, 2002. 333-410

Stern, Hal; Eisler, Mike; Labiaga, Ricardo. Managing NFS and NIS, 2nd Edition. Sebastopol: O' Reilly & Associates, Inc, 2001. 86, 104, 120, 360

Tool and Software URLs:

Apache - http://httpd.apache.org
Barnyard - http://www.fidelissec.com/mudpit.html
BIND - http://www.isc.org
FreeS/WAN - http://www.freeswan.org
NetCat - http://www.atstake.com/research/tools/network_utilities/
Nmap - http://www.insecure.org/nmap
SendMail - http://www.sendmail.org
Snort - http://www.snort.org
SnortCenter - http://users.pandora.be/larc/
Squid - http://www.squid-cache.org/
STunnel - http:// www.stunnel.org

Appendix A: Internal NetFilter firewall script:

```
#!/bin/sh

IFCONFIG="/sbin/ifconfig"
IPTABLES="/sbin/iptables"
OUTSIDE="eth0"
OUTSIDE_ADDR="192.168.100.10"
INSIDE="eth1"
INSIDE_ADDR="192.168.101.1"
CLASS_A_1918="10.0.0.0/8"
CLASS_B_1918="172.16.0.0/12"
CLASS_C_1918="192.168.0.0/16"
DMZ_NETWORK="192.168.10.0/24"
SUPPLIERS_VPN_NETWORK="192.168.31.0/24"
PARTNERS_VPN_NETWORK="192.168.33.0/24"
REMOTE_EMPL_VPN_NETWORK="192.168.35.0/24"
INSIDE_NETWORK="192.168.100.0/24"

# ------------------------------------------
# Load necessary modules
# ------------------------------------------
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe ipt_state
/sbin/modprobe iptable_filter

# Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
# Enable broadcast echo protection
echo 1> /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
# Disable source routed packets
for INT in /proc/sys/net/ipv4/conf/*/accept_source_route ; do
  echo 0 > ${INT}
done
# Enable TCP SYN Cookie Protection
echo 1 > /proc/sys/net/ipv4/tcp_syncookies


# ------------------------------------------
# Clear out all existing firewall rules
# ------------------------------------------
${IPTABLES} --flush
${IPTABLES} -t nat --flush
${IPTABLES} -t mangle --flush
${IPTABLES} --delete-chain
```

```
# ------------------------------------------------
# Default DROP policy
# ------------------------------------------------
${IPTABLES} --policy INPUT DROP
${IPTABLES} --policy FORWARD DROP
${IPTABLES} --policy OUTPUT DROP

# ------------------------------------------------
# Static IP translations
# ------------------------------------------------
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.11 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.11
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.11 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.11
${IFCONFIG} ${OUTSIDE}:0 192.168.100.11
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.12 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.12
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.12 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.12
${IFCONFIG} ${OUTSIDE}:1 192.168.100.12
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.13 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.13
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.13 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.13
${IFCONFIG} ${OUTSIDE}:2 192.168.100.13
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.14 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.14
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.14 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.14
${IFCONFIG} ${OUTSIDE}:3 192.168.100.14
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.15 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.15
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.15 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.15
${IFCONFIG} ${OUTSIDE}:4 192.168.100.15
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.16 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.16
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.16 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.16
${IFCONFIG} ${OUTSIDE}:5 192.168.100.16
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.17 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.17
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.17 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.17
```

```
${IFCONFIG} ${OUTSIDE}:6 192.168.100.17
${IPTABLES} -t nat -A PREROUTING -d 192.168.100.18 -i ${OUTSIDE} -j DNAT
--to-destination 192.168.101.18
${IPTABLES} -t nat -A POSTROUTING -s 192.168.101.18 -o ${OUTSIDE} -j
SNAT --to-source 192.168.100.18
${IFCONFIG} ${OUTSIDE}:7 192.168.100.18

# ------------------------------------------
# INPUT table
# ------------------------------------------
${IPTABLES} -A INPUT -i lo -j ACCEPT
${IPTABLES} -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
${IPTABLES} -A INPUT -p tcp --dport 22 -j ACCEPT
${IPTABLES} -A INPUT -p icmp -j ACCEPT
${IPTABLES} -A INPUT -s ${CLASS_A_1918} -j DROP
${IPTABLES} -A INPUT -s ${CLASS_B_1918} -j DROP
${IPTABLES} -A INPUT -m state --state INVALID -j DROP

# ------------------------------------------
# OUTPUT table
# ------------------------------------------
${IPTABLES} -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
${IPTABLES} -A OUTPUT -p udp --dport 53 -o ${INSIDE} -j ACCEPT
${IPTABLES} -A OUTPUT -p tcp --dport 53 -o ${INSIDE} -j ACCEPT
${IPTABLES} -A OUTPUT -p icmp -j ACCEPT
${IPTABLES} -A OUTPUT -m state --state INVALID -j DROP

# ------------------------------------------
# FORWARD table
# ------------------------------------------
${IPTABLES} -A FORWARD -m state --state ESTABLISHED,RELATED -j
ACCEPT
${IPTABLES} -A FORWARD -m state --state INVALID -j DROP
${IPTABLES} -A FORWARD -d 255.255.255.255 -j DROP
${IPTABLES} -A FORWARD -s ${CLASS_A_1918} -j DROP
${IPTABLES} -A FORWARD -s ${CLASS_B_1918} -j DROP
# Allow access from internal network
${IPTABLES} -A FORWARD -p tcp -s ${INSIDE_NETWORK} -d 192.168.101.14
--dport 636 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${INSIDE_NETWORK} -d 192.168.101.13
--dport 25 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${INSIDE_NETWORK} -d 192.168.101.13
--dport 993 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${INSIDE_NETWORK} -d 192.168.101.12
--dport 53 -j ACCEPT
```

```
${IPTABLES} -A FORWARD -p udp -s ${INSIDE_NETWORK} -d 192.168.101.12
--dport 53 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${INSIDE_NETWORK} -d 192.168.101.14
--dport 111 -j ACCEPT
${IPTABLES} -A FORWARD -p udp -s ${INSIDE_NETWORK} -d 192.168.101.14
--dport 111 -j ACCEPT
${IPTABLES} -A FORWARD -p udp -s ${INSIDE_NETWORK} -d 192.168.101.14
--dport 1039 -j ACCEPT
${IPTABLES} -A FORWARD -p udp -s ${INSIDE_NETWORK} -d 192.168.101.14
--dport 2049 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${INSIDE_NETWORK} -d 192.168.101.17
--dport 443 -j ACCEPT
# Allow access from the DMZ network
${IPTABLES} -A FORWARD -p tcp -s 192.168.10.10 -d 192.168.101.16 --dport
3306 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s 192.168.10.20 -d 192.168.101.16 --dport
3306 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s 192.168.10.1 -d 192.168.101.13 --dport 25
-j ACCEPT
# Allow access from the REMOTE EMPLOYEES VPN network
${IPTABLES} -A FORWARD -p tcp -s ${VPN_NETWORK} -d 192.168.101.13 --
dport 25 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${VPN_NETWORK} -d 192.168.101.14 --
dport 993 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${VPN_NETWORK} -d 192.168.101.12 --
dport 53 -j ACCEPT
${IPTABLES} -A FORWARD -p udp -s ${VPN_NETWORK} -d 192.168.101.12 --
dport 53 -j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s ${VPN_NETWORK} -d 192.168.101.17 --
dport 443 -j ACCEPT
# Allow access from the protected internal network to the other networks
${IPTABLES} -A FORWARD -p tcp -s 192.168.101.13 -d 192.168.10.5 --dport 25
-j ACCEPT
${IPTABLES} -A FORWARD -p tcp -s 192.168.101.12 -d 192.168.10.5 --dport 53
-j ACCEPT
${IPTABLES} -A FORWARD -p udp -s 192.168.101.12 -d 192.168.10.5 --dport
53 -j ACCEPT
```

Appendix B: Border Router Configuration:

```
!
service timestamps debug uptime
service timestamps log uptime
service password-encryption
no service finger
no service tcp-small-servers
no service udp-small-servers
!
hostname giac-router
!
enable secret 5 $1$QPZq$CiYKu9lmKxNLJrttMTuRf0
!
no ip name-server
no ip source-route
no ip finger
!
ip subnet-zero
no ip domain-lookup
ip routing
!
interface FastEthernet 0/0
 no shutdown
 description connected to Internet
 ip address 105.0.0.1 255.255.255.248
 ip access-group 3 in
 keepalive 10
!
interface FastEthernet 0/1
 no shutdown
 description connected to EthernetLAN
 ip address 104.0.0.1 255.255.255.0
 ip access-group 6 in
 keepalive 10
!
router rip
 version 2
 network 104.0.0.0
 passive-interface FastEthernet 0/0
 no auto-summary
!
!
ip classless
!
access-list 1 permit 127.0.0.1 0.0.0.0
```

```
access-list 3 deny 10.0.0.0 0.255.255.255
access-list 3 deny 172.16.0.0 0.15.255.255
access-list 3 deny 192.168.0.0 0.0.255.255
access-list 3 deny 224.0.0.0 31.255.255.255
access-list 3 deny 127.0.0.0 0.255.255.255
access-list 3 permit any
access-list 6 deny 10.0.0.0 0.255.255.255
access-list 6 deny 172.16.0.0 0.15.255.255
access-list 6 deny 192.168.0.0 0.0.255.255
access-list 6 deny 224.0.0.0 31.255.255.255
access-list 6 deny 127.0.0.0 0.255.255.255
access-list 6 permit any
!
! IP Static Routes
ip route 0.0.0.0 0.0.0.0 FastEthernet 0/0
no ip http server
no ip bootp
no ip direct-broadcast
no ip unreachables
no snmp
no snmp-server location
no snmp-server contact
banner motd #ATTENTION: Un-Authorized Access Prohibited#
!
line console 0
 exec-timeout 0 0
 password W5smdy5R
 login
!
line vty 0 4
 access-class 1 in
 password 9Jd9KWkY
 login
!
end
```

Appendix C: Primary Firewall Configuration:

```
nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 dmz security10
nameif ethernet3 vpn security30
nameif ethernet4 intf4 security25
nameif ethernet5 sfail security15
enable password CBqmWlVHnqcjHxGZ encrypted
passwd cItR6hpAokBWJLtJ encrypted
hostname giac-fw
domain-name giac.com
fixup protocol ftp 21
fixup protocol http 80
fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
fixup protocol ils 389
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
fixup protocol smtp 25
names
access-list outside_in permit icmp any any echo-reply
access-list outside_in permit icmp any any unreachable
access-list outside_in deny ip 10.0.0.0 255.0.0.0 any
access-list outside_in deny ip 172.16.0.0 255.240.0.0 any
access-list outside_in deny ip 192.168.0.0 255.255.0.0 any
access-list outside_in permit tcp any host 104.0.0.10 eq www
access-list outside_in permit tcp any host 104.0.0.15 eq 25
access-list outside_in permit tcp any host 104.0.0.15 eq 53
access-list outside_in permit udp any host 104.0.0.15 eq 53
access-list outside_in permit tcp any host 104.0.0.20 eq https
access-list dmz_in permit icmp any any echo-reply
access-list dmz_in permit icmp any any unreachable
access-list dmz_in permit tcp host 192.168.10.20 host 192.168.100.16 eq 3306
access-list dmz_in permit tcp host 192.168.10.18 host 192.168.100.16 eq 3306
access-list dmz_in permit tcp host 192.168.10.5 host 192.168.100.13 eq 25
access-list dmz_in permit tcp host 192.168.10.5 any eq 53
access-list dmz_in permit udp host 192.168.10.5 any eq 53
access-list vpn_in permit icmp any any echo-reply
access-list vpn_in permit icmp any any unreachable
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.10
eq www
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.10
```

```
eq https
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.13
eq 25
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.13
eq 993
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.12
eq 53
access-list vpn_in permit udp 192.168.30.0 255.255.255.0 host 192.168.100.12
eq 53
access-list vpn_in permit tcp 192.168.30.0 255.255.255.0 host 192.168.100.17
eq https
access-list inside_in permit icmp any any echo-request
access-list inside_in permit tcp host 192.168.100.10 any eq 80
access-list inside_in permit tcp host 192.168.100.10 any eq 443
access-list inside_in permit tcp host 192.168.100.12 any eq 53
access-list inside_in permit udp host 192.168.100.12 any eq 53
pager lines 24
logging off
logging trap warnings
logging history warnings
interface ethernet0 auto
interface ethernet1 auto
interface ethernet2 auto
interface ethernet3 auto
interface ethernet4 auto shutdown
interface ethernet5 auto
mtu outside 1500
mtu inside 1500
mtu dmz 1500
mtu vpn 1500
mtu intf4 1500
mtu sfail 1500
ip address outside 104.0.0.2 255.255.255.0
ip address inside 192.168.100.1 255.255.255.0
ip address dmz 192.168.10.1 255.255.255.0
ip address vpn 192.168.30.1 255.255.255.0
ip address intf4 127.0.0.1 255.255.255.255
ip address sfail 10.0.0.1 255.255.255.252
ip audit info action alarm
ip audit attack action alarm
failover
failover timeout 0:00:00
failover poll 15
failover ip address outside 104.0.0.3
failover ip address inside 192.168.100.2
```

```
failover ip address dmz 192.168.10.2
failover ip address vpn 192.168.30.2
failover ip address intf4 0.0.0.0
failover ip address sfail 10.0.0.2
failover link sfail
pdm history enable
arp timeout 14400
global (outside) 1 104.0.0.4 netmask 255.255.255.0
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (dmz,outside) 104.0.0.10 192.168.10.10 netmask 255.255.255.255 0 0
static (dmz,outside) 104.0.0.15 192.168.10.5 netmask 255.255.255.255 0 0
static (dmz,outside) 104.0.0.18 192.168.10.18 netmask 255.255.255.255 0 0
static (dmz,outside) 104.0.0.20 192.168.10.20 netmask 255.255.255.255 0 0
access-group outside_in in interface outside
access-group dmz_in in interface dmz
access-group vpn_in in interface vpn
access-group inside_in in interface inside
route outside 0.0.0.0 0.0.0.0 104.0.0.1 1
timeout xlate 3:00:00
timeout conn 0:00:00 half-closed 0:10:00 udp 0:01:00 rpc 0:10:00 h323 0:05:00
sip 0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
aaa-server LOCAL protocol local
no snmp-server enable traps
floodguard enable
no sysopt connection permit-ipsec
no sysopt route dnat
telnet timeout 10
ssh timeout 10
terminal width 80
```

Appendix D: VPN Server Configuration /etc/ipsec.conf and /usr/local/lib/ipsec/remote-employees:

<start /etc/ipsec.conf>
```
version 2.0     # conforms to second version of ipsec.conf specification

# basic configuration
config setup

# Add connections here.
conn remote-employee
    left=104.0.0.100
    leftsubnet=192.168.100.0/24
    leftid=@vpn1.giac.com
    leftupdown="ipsec remote-employees"
    leftrsasigkey=0sAQOUx/OT+/f82EEpMZp4QQ87gfpcAwx3HotSZr0dLwyPu0qhAYGdHTRhhbDQUwJ41
ptEXmCxIDbac/8EMj5ruHjGHBrRe1FjbcgU0TGbKzyc4fxxkfKEHgwc8BkvYpcF8KXTM0GABaFL8FSTklLN
sGWYbFrnLNQ7TfZPUV4lPxBxm17iEFlmAd3vEn3DsY7OrmBn5Wv0fg5RK3eQDtDRpDjeDSj/cZRQAdTB
t8cmj5VDAEADUFK72QKVCnPNPZZcH01jYEiCTw8I56rcc4bEN7rVv/25NwCUjGkgGvtXL0cqIg4bl/mIWaH
7Ho5Fvi0MdfKddVHzffTTIGlRv0CHQIpBDk8OgEjnfWNsiRZH91wfP321
    rightnexthop=%defaultroute
    right=%any
    rightid=@empl-laptop.giac.com
    rightrsasigkey=0sAQOJDu6+oDlSrj1KryqQSJ0W3rCrqzrXVfnqbslmkWdbDmvWnyvLCF9RE1arDu8PIsP
3RF9rBoPJn5gePU1ZgiEXsicPBIGepX7sg1CJm1+Q+IPvKtFXr+bNalhi4imtSYbvBlZcTTFmEyRGB06fOy1
zAVRbmfdDPNmk3/trj75vypeYS0ILnFtmV5X38MlrpCZiQSA58mpORo8y2UdGT/LtqVoUJIlaqeS0faZ6xkh
MRdDwjDGGcmAtsYBPKVi9pW8AKwgFrZL1Aij4fPIBaom5YCrZLNqci6pYVvAFLWqV9cK7nkNekJ5rP3Uf
ob0S8mBHxQDc47M9o79VxCSbOlCt+xBgetAroNuOSGj82SOg7m/3
    auto=start
<end /etc/ipsec.conf>

<start /usr/local/lib/ipsec/remote-employees>
#! /bin/sh
# default updown script
# Copyright (C) 2000, 2001  D. Hugh Redelmeier, Henry Spencer
#
# This program is free software; you can redistribute it and/or modify it
# under the terms of the GNU General Public License as published by the
# Free Software Foundation; either version 2 of the License, or (at your
# option) any later version.  See <http://www.fsf.org/copyleft/gpl.txt>.
#
# This program is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
# or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
# for more details.
#
# RCSID $Id: _updown.in,v 1.20 2002/04/24 07:36:06 mcr Exp $



# CAUTION: Installing a new version of FreeS/WAN will install a new
# copy of this script, wiping out any custom changes you make.  If
# you need changes, make a copy of this under another name, and customize
# that, and use the (left/right)updown parameters in ipsec.conf to make
# FreeS/WAN use yours instead of this default one.



# check interface version
case "$PLUTO_VERSION" in
```

```
1.[0])          # Older Pluto?!?  Play it safe, script may be using new features.
                echo "$0: obsolete interface version \`$PLUTO_VERSION'  ," >&2
                echo "$0:       called by obsolete Pluto?" >&2
                exit 2
                ;;
1.*)            ;;
*)              echo "$0: unknown interface version \`$PLUTO_VERSION'  " >&2
                exit 2
                ;;
esac

# check parameter(s)
case "$1:$*" in
'  :'  )                                    # no parameters
                ;;
ipfwadm:ipfwadm)        # due to (left/right)firewall; for default script only
                ;;
custom:*)               # custom parameters (see above CAUTION comment)
                ;;
*)              echo "$0: unknown parameters \`$*'  " >&2
                exit 2
                ;;
esac

# utility functions for route manipulation
# Meddling with this stuff should not be necessary and requires great care.
uproute() {
                doroute add
}
downroute() {
                doroute del
}
doroute() {
                parms="-net $PLUTO_PEER_CLIENT_NET netmask $PLUTO_PEER_CLIENT_MASK"
                parms2="dev $PLUTO_INTERFACE gw $PLUTO_NEXT_HOP"
                case "$PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK" in
                "0.0.0.0/0.0.0.0")
                                # horrible kludge for obscure routing bug with opportunistic
                                it="route $1 -net 0.0.0.0 netmask 128.0.0.0 $parms2 &&
                                                route $1 -net 128.0.0.0 netmask 128.0.0.0 $parms2"
                                ;;
                *)              it="route $1 $parms $parms2"
                                ;;
                esac
                eval $it
                st=$?
                if test $st -ne 0
                then
                                # route has already given its own cryptic message
                                echo "$0: \`$it'   failed" >&2
                                if test " $1 $st" = " add 7"
                                then
                                                # another totally undocumented interface -- 7 and
                                                # "SIOCADDRT: Network is unreachable" means that
                                                # the gateway isn'  t reachable.
                                                echo "$0: (incorrect or missing nexthop setting??)" >&2
                                fi
                fi
                return $st
}
```

```
# the big choice
case "$PLUTO_VERB:$1" in
prepare-host:*|prepare-client:*)
        # delete possibly-existing route (preliminary to adding a route)
        case "$PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK" in
        "0.0.0.0/0.0.0.0")
                # horrible kludge for obscure routing bug with opportunistic
                it="route del -net 0.0.0.0 netmask 128.0.0.0 2>&1 ;
                        route del -net 128.0.0.0 netmask 128.0.0.0 2>&1"
                ;;
        *)
                it="route del -net $PLUTO_PEER_CLIENT_NET \
                                                        netmask
$PLUTO_PEER_CLIENT_MASK 2>&1"
                ;;
        esac
        oops="` eval $it` "
        status="$?"
        if test " $oops" = " " -a " $status" != " 0"
        then
                oops="silent error, exit status $status"
        fi
        case "$oops" in
        '  SIOCDELRT: No such process'  *)
                # This is what route (currently -- not documented!) gives
                # for "could not find such a route".
                oops=
                status=0
                ;;
        esac
        if test " $oops" != " " -o " $status" != " 0"
        then
                echo "$0: \` $it'   failed ($oops)" >&2
        fi
        exit $status
        ;;
route-host:*|route-client:*)
        # connection to me or my client subnet being routed
        uproute
        ;;
unroute-host:*|unroute-client:*)
        # connection to me or my client subnet being unrouted
        downroute
        ;;
up-host:*)
        # connection to me coming up
        # If you are doing a custom version, firewall commands go here.
        ;;
down-host:*)
        # connection to me going down
        # If you are doing a custom version, firewall commands go here.
        ;;
up-client:)
        # connection to my client subnet coming up
        # If you are doing a custom version, firewall commands go here.
    /sbin/iptables -t nat -A POSTROUTING -s $PLUTO_PEER -o eth1 -j SNAT --to-source 192.168.30.30
        ;;
down-client:)
        # connection to my client subnet going down
```

```
                # If you are doing a custom version, firewall commands go here.
        /sbin/iptables -t nat -D POSTROUTING -s $PLUTO_PEER -o eth1 -j SNAT --to-source 192.168.30.30
                ;;
up-client:ipfwadm)
                # connection to client subnet, with (left/right)firewall=yes, coming up
                # This is used only by the default updown script, not by your custom
                # ones, so do not mess with it; see CAUTION comment up at top.
                ipfwadm -F -i accept -b -S $PLUTO_MY_CLIENT_NET/$PLUTO_MY_CLIENT_MASK \
                                -D $PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK
                ;;
down-client:ipfwadm)
                # connection to client subnet, with (left/right)firewall=yes, going down
                # This is used only by the default updown script, not by your custom
                # ones, so do not mess with it; see CAUTION comment up at top.
                ipfwadm -F -d accept -b -S $PLUTO_MY_CLIENT_NET/$PLUTO_MY_CLIENT_MASK \
                                -D $PLUTO_PEER_CLIENT_NET/$PLUTO_PEER_CLIENT_MASK
                ;;
*)              echo "$0: unknown verb \` $PLUTO_VERB'   or parameter \` $1'  " >&2
                exit 1
                ;;
esac
<end /usr/local/lib/ipsec/remote-employees>
```

Appendix E: VPN Server Firewall Script:

```
#!/bin/sh

IFCONFIG="/sbin/ifconfig"
IPTABLES="/sbin/iptables"
OUTSIDE="eth0"
OUTSIDE_ADDR="104.0.0.100"
INSIDE="eth1"
INSIDE_ADDR="192.168.30.20"
CLASS_A_1918="10.0.0.0/8"
CLASS_B_1918="172.16.0.0/12"
CLASS_C_1918="192.168.0.0/16"
INSIDE_NETWORK="192.168.30.0/24"

# ----------------------------------------------
# Load necessary modules
# ----------------------------------------------
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe ipt_state
/sbin/modprobe iptable_filter

# Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
# Enable broadcast echo protection
echo 1> /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
# Disable source routed packets
for INT in /proc/sys/net/ipv4/conf/*/accept_source_route ; do
  echo 0 > ${INT}
done
# Enable TCP SYN Cookie Protection
echo 1 > /proc/sys/net/ipv4/tcp_syncookies


# ----------------------------------------------
# Clear out all existing firewall rules
# ----------------------------------------------
${IPTABLES} --flush
${IPTABLES} -t nat --flush
${IPTABLES} -t mangle --flush
${IPTABLES} --delete-chain

# ----------------------------------------------
# Default DROP policy for INPUT table
# ----------------------------------------------
```

```
${IPTABLES} --policy INPUT DROP

# ------------------------------------------
# INPUT table
# ------------------------------------------
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p icmp --icmp-type fragmentation-needed -j ACCEPT
iptables -A INPUT -p 50 -d ${OUTSIDE_ADDR} -j ACCEPT
iptables -A INPUT -p 51 -d ${OUTSIDE_ADDR} -j ACCEPT
iptables -A INPUT -p udp -d ${OUTSIDE_ADDR} --dport 500 -j ACCEPT
```

Appendix F: VPN Client Configuration /etc/ipsec.conf:

```
version     2.0         # conforms to second version of ipsec.conf specification

# basic configuration
config setup

# Add connections here.
conn giac-vpn
   left=%defaultroute
   leftid=@empl-laptop.giac.com
   leftrsasigkey=0sAQOJDu6+oDlSrj1KryqQSJ0W3
   rCrqzrXVfnqbslmkWdbDmvWnyvLCF9RE1arDu8P
   IsP3RF9rBoPJn5gePU1ZgiEXsicPBIGepX7sg1CJ
   m1+Q+lPvKtFXr+bNalhi4imtSYbvBlZcTTFmEyRG
   B06fOy1zAVRbmfdDPNmk3/trj75vypeYS0ILnFtmV
   5X38MlrpCZiQSA58mpORo8y2UdGT/LtqVoUJlIaq
   eS0faZ6xkhMRdDwjDGGcmAtsYBPKVi9pW8AKw
   gFrZL1Aij4fPIBaom5YCrZLNqci6pYVvAFLWqV9cK
   7nkNekJ5rP3Ufob0S8mBHxQDc47M9o79VxCSbOl
   Ct+xBgetAroNuOSGj82SOg7m/3
   right=104.0.0.100
   rightsubnet=192.168.100.0/24
   rightid=@vpn1.giac.com
   rightrsasigkey=0sAQOUx/OT+/f82EEpMZp4QQ87
   gfpcAwx3HotSZr0dLwyPu0qhAYGdHTRhhbDQU
   wJ41ptEXmCxIDbac/8EMj5ruHjGHBrRe1FjbcgU
   0TGbKzyc4fxxkfKEHgwc8BkvYpcF8KXTM0GABa
   FL8FSTklLNsGWYbFrnLNQ7TfZPUV4lPxBxm17i
   EFlmAd3vEn3DsY7OrmBn5Wv0fg5RK3eQDtDRp
   DjeDSj/cZRQAdTBt8cmj5VDAEADUFK72QKVCn
   PNPZZcH01jYEiCTw8I56rcc4bEN7rVv/25NwCUj
   GkgGvtXL0cqIg4bl/mIWaH7Ho5Fvi0MdfKddVHzff
   TTIGlRv0CHQlpBDk8OgEjnfWNsiRZH91wfP321
   auto=add
```