Global Information Assurance Certification Paper

Firewalls, Perimeter Protection, and VPNs

SANS 2001 Baltimore MD
GCFW Practical
Version 1.5e

Chris Talianek

A Small Business Perimeter Security Model

Table of Contents

Assignment 1: Security Architecture
- Business Profile
- Business Requirements
- Architecture Overview
- Network Diagram
- Network Structure
- Security Component Descriptions
  - Border Router
  - Border Firewall
  - Internal Firewall
  - Ethernet Switches
  - Intrusion Detection Systems
  - Supplier and Customer VPN Access
  - Business Partner Access
  - Employee Remote Access
  - Public and VPN Service Net Hosts

Assignment 2: Security Policy
- Border Router
  - Comments
  - Border Router Policy Summary
  - Loading the Encryption Software Image
  - Border Router Configuration Tutorial
  - Border Router Configuration File
- Border Firewall
  - Comments
  - Border Firewall Policy Summary
  - Loading the Encryption Software Key
  - Border Firewall Configuration Tutorial
    - Comments
    - Caveats
    - Tutorial
  - Border Firewall Configuration File
- Internal Firewall
  - Comments
  - Internal Firewall Policy Summary
  - IPTables Overview
  - System Hardening
  - Internal Firewall Configuration File

Assignment 3: Audit Your Security Architecture
- Planning the Assessment

Implementing the Assessment
Testing Environment
Testing and Monitoring Tools
Assessment Test 1
Assessment Test 2
Assessment Test 3
Assessment Test 4
Assessment Test 5
Assessment Test 6
Perimeter Analysis
Assignment 4: Design Under Fire
Firewall Attack
Server Attack
Denial-of-Service Attack
References

Assignment 1: Security Architecture

Business Profile

GIAC Enterprises is a small, start-up business employing fewer than 50 employees. Its primary line of business is the online sales of fortune cookie sayings. The company is located in the United States, where it does the majority of its business. GIAC recently completed a merger with a business partner located in China. The companies freely share resources, but their sales territories are separate. The China branch covers Asian and European markets, and the U.S. branch covers the United States and Canadian markets. The customer base consists of a few companies who bulk purchase the fortunes. GIAC has several suppliers of fortune sayings who are located in Asia and the U.S.

Business Requirements

Since the majority of the business is transacted over the Internet, GIAC has a business need for a reliable and secure E-commerce network. The GIAC principals outlined the following high-level requirements and limitations:

- Reliable Internet connectivity
- Secure network communications at a reasonable cost
- Simple network design that can be maintained by an IT staff of five employees consisting of a supervisor and four administrators
- Public web site and e-mail services
- Private web site for suppliers and customers
- Remote access to internal resources for the business partner
- Remote access to internal resources for occasional employee's at home and for sales personnel on the road
- Implementation of technologies such as firewalls, VPNs, virus scanners, and intrusion detection to protect the company's assets
- Implementation of employee education as to the proper use of company resources including topics such as appropriate use of web sites and email, passwords, virus awareness, and ethical use of company resources
- Logging of network traffic to provide a high-level auditing of activities
- Offsite storage of computer systems backups

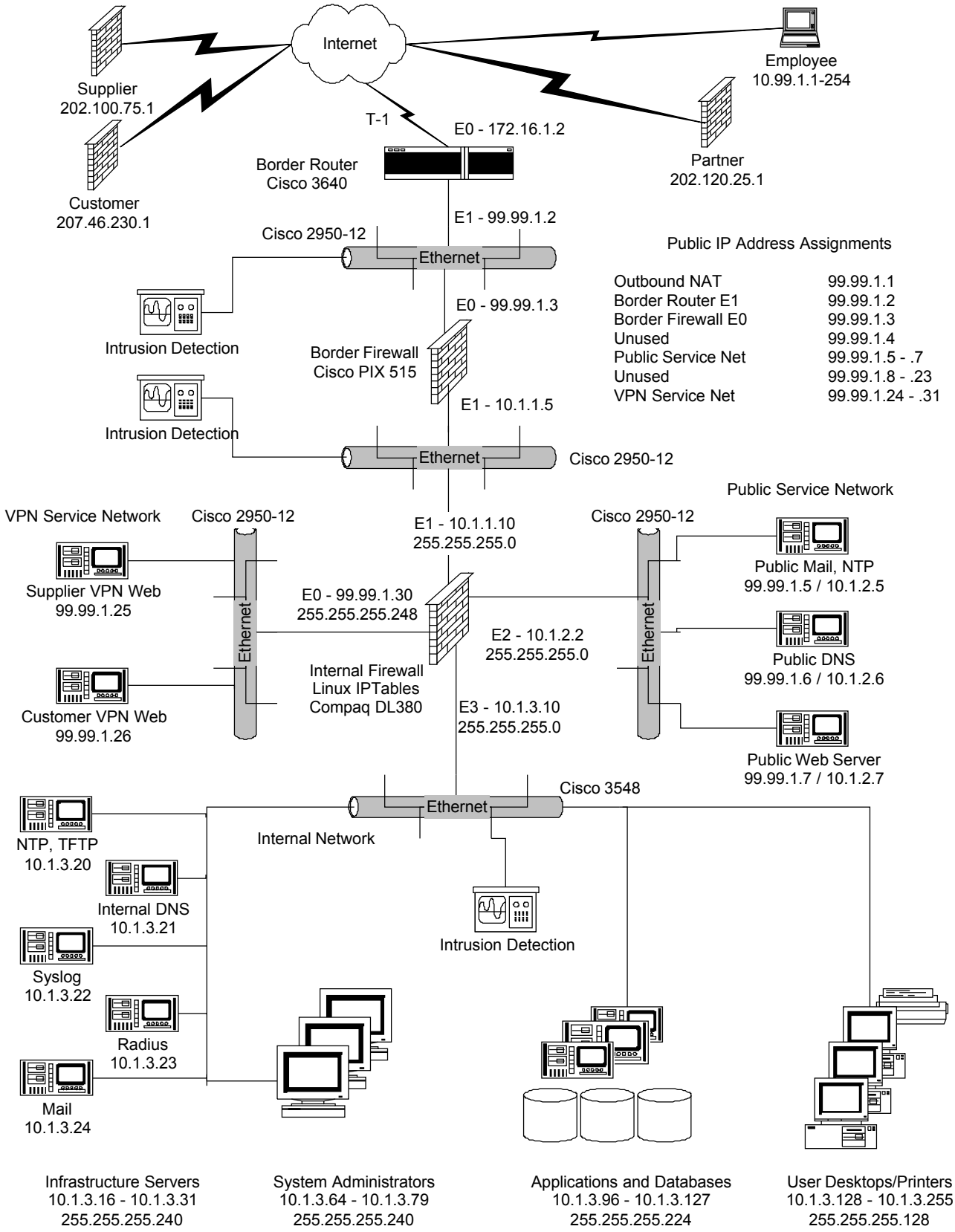
Architecture Overview

The following summarizes the architecture guidelines to satisfy the business communications needs. The budget allotted for the implementation is very tight, and the IT supervisor must carefully justify all expenditures to the business principals.

- A secure, limited access computer room with proper electrical and environmental controls
- A T-1 connection to the Internet
- Public IP addressing scheme of 32 addresses: Internet Service Providers are rather conservative lately when assigning address blocks.
- Cisco network equipment and Compaq rack mount servers running Linux for the general infrastructure: Standardization on reliable, high-quality, popular hardware and operating systems reduces the number of skill sets that a limited staff needs.
- Selection of reputable and popular equipment providers like Microsoft, Lotus, Oracle, Sun, and Compaq for the internal servers and desktops will be based primarily upon business needs. Standards will be developed and followed whenever possible.
- An external firewall to control traffic in and out of the Internet
- An internal firewall to control traffic within the internal networks
- Split DNS, Mail, and NTP services: Separate servers will provide these services for the public and for the private networks
- A public web server for corporate promotion, news, and events
- Encrypted VPN access for the business partner to access the GIAC network
- Authenticated VPN access for the customers and suppliers to dedicated private web servers
- Encrypted VPN access for remote employee access from anywhere
- Intrusion detection sensors to alert the administrators of suspicious activity or penetrations
- Centralized system logging to an internal server
- Secured SSH access from system administrator workstations to all servers, firewalls, and routing equipment
- Backup and system maintenance schedules including operating system upgrades and patches

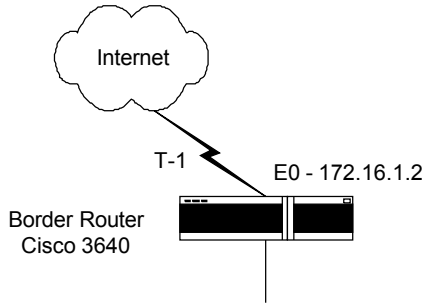
The GIAC network is diagrammed below:

tag:

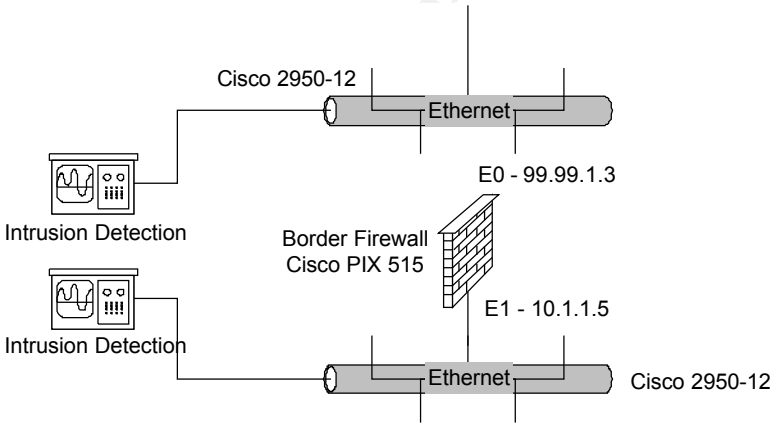


Network Structure

A T-1 communications line provides the connection to the Internet. At the border, a Cisco 3640 router will terminate the T-1 connection. This is the only connection to the Internet due to budget constraints.

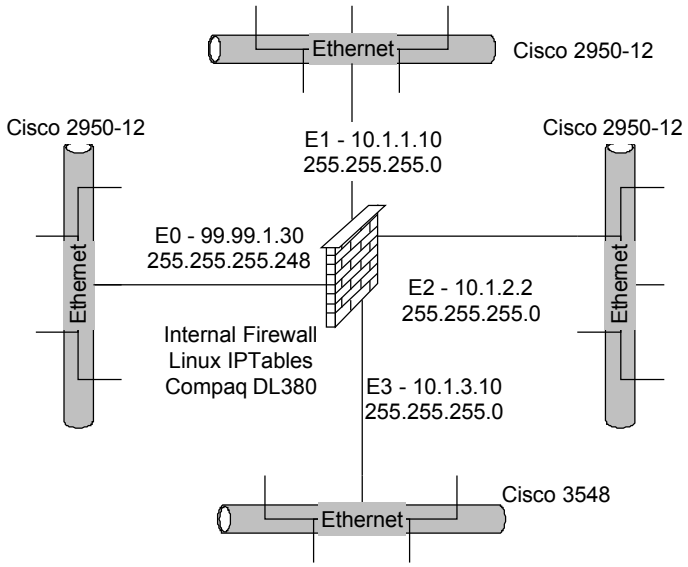


Inside of the router, a Cisco 2950-12 switch supports the Ethernet segment to the border firewall. The switch port that connects the border router will be spanned to allow an intrusion detection sensor to monitor all traffic inside the border router. The border firewall is a Cisco PIX 515 with two interfaces. On the inside interface, another Cisco 2950-12 switch provides the Ethernet segment to the internal firewall. There is another intrusion detection sensor monitoring the traffic on the inside of the PIX firewall. The PIX firewall will provide the primary traffic filtering into and out of the GAIC network. It also will be the termination point for VPN connections. The internal IP addresses of all outbound connections to the Internet will be NAT-ed to a single public address that is assigned to the external interface of the PIX.

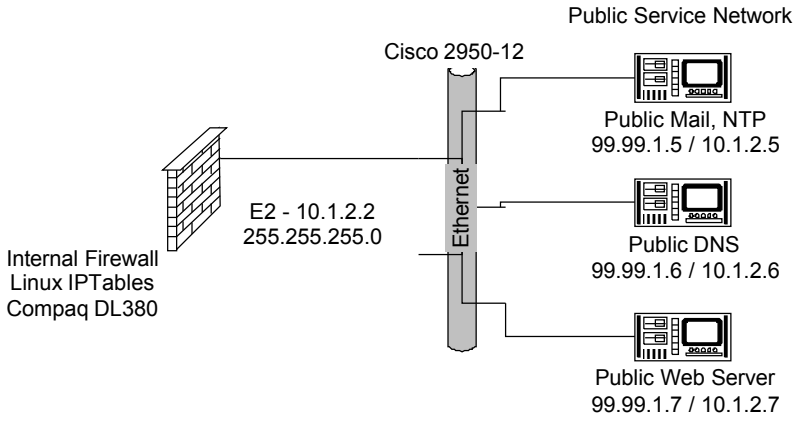


The internal firewall is a Compaq DL380 Linux server running IPTables. The Linux operating system will be installed with the bare minimum RPM packages to reduce the number of O/S

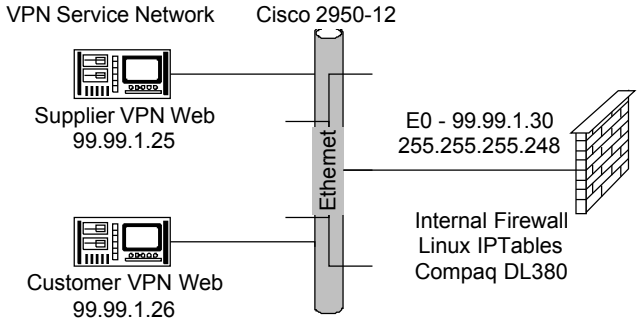
vulnerabilities that may arise in the future. The O/S will of course be hardened. The primary job of this firewall is to securely control the routing of all traffic to and from the appropriate internal or service networks. It provides defense-in-depth by replicating many of the traffic control policies that the border firewall enforces. If vulnerability in the PIX O/S is discovered, or if an administrative mistake in maintaining the PIX occurs, this firewall can provide additional protection. Attached to the internal firewall are an internal network and two service networks, which are detailed further below.



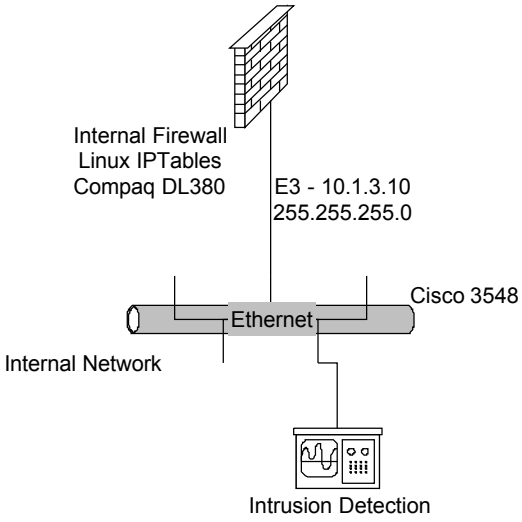
A public service network is attached to the internal firewall. A Cisco 2950-12 switch supports the Ethernet segment. This network provides external NTP, DNS, mail, and corporate web site services. Only external DNS addresses of the public service net computers are advertised to the outside world. The external outbound NAT address of the firewall is also assigned a name since some web-based applications will perform reverse lookups on the source address. Of course, the external name of "outbound.giac.com" does not disclose any useful information. Internal requests for external DNS, NTP, and mail service are forwarded to these public service net machines, which in turn forward the requests out to the Internet. This provides some additional protection for the inside servers. Sometimes, vulnerabilities are exploited when the requesting server asks a malicious server a question, and the malicious server replies with the exploit code as a response. This proxy technique limits this exposure to the public service net.



A VPN service network is attached to the internal firewall. A Cisco 2950-12 switch supports the Ethernet segment. This network provides private web services only to registered suppliers and customers. General public access is not permitted, so the threat level of the traffic on this service network is reduced. All traffic will be through an IPSEC authenticated connection so that we can ensure who is accessing these servers. The inbound supplier server is separated from the outbound customer server. Although we hope we are doing business with ethical customers, this separation should offer some protection from unscrupulous customers or suppliers from accessing the other's files. Web based applications will provide an interface for customers and suppliers to retrieve and deliver fortune cookie sayings, so the primary traffic on this net will be HTTP, HTTPS, and FTP based.

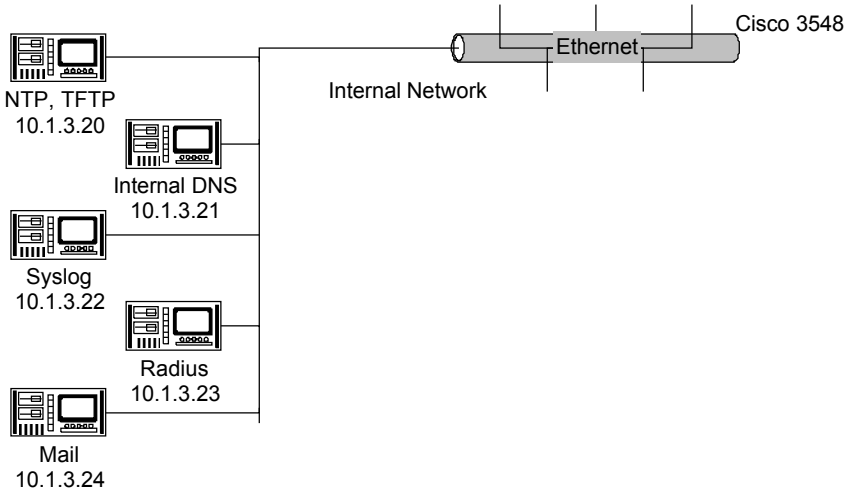


The last network connected to the internal firewall is the GIAC internal LAN. Stackable Cisco 3548 switches support the internal LAN segment. The GIAC internal network is not large enough to require multiple separate segments today, and budgetary constraints limit the IT department from initially segmenting the internal LAN. As the company and the LAN grow, the internal LAN may need segmentation to provide additional traffic controls and improved performance. An intrusion detection sensor is placed on the LAN to monitor inbound and outbound traffic by spanning the switch port that connects to the firewall.



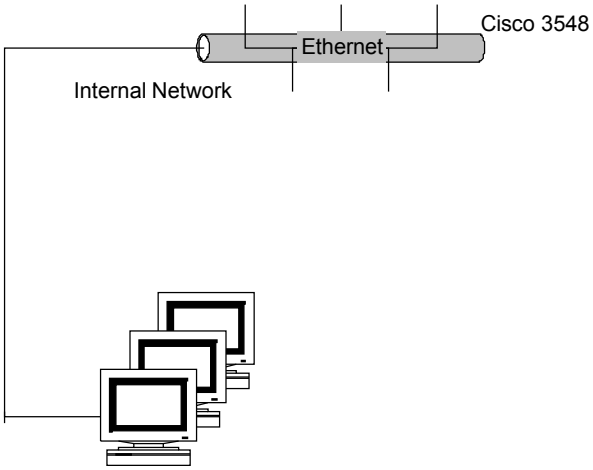
The internal LAN structure organizes the computers into subnets based upon their purpose. This will make future segmentation of the LAN easier. It also makes firewall rule sets smaller since subnetted ranges can allow several individual rules to be aggregated into one rule for the entire subnet.

The infrastructure servers are grouped together into a subnet range. These servers provide routine services to support general day-to-day operations. The internal services include NTP, DNS, and mail. All logging services for the internal LAN, service networks, and border control devices are logged to a central syslog server on this subnet. Radius services for remote employee VPN authentications are located here. A TFTP service is also provided for saving the border router and firewall configurations. The DNS, NTP, and mail servers do not connect directly to the Internet. They make requests to the public service net, which forwards their requests to the outside in a proxy-like fashion. Since these are mission critical services that have interactions with servers outside of the internal LAN, they are hardened and monitored like publicly accessible service net machines.



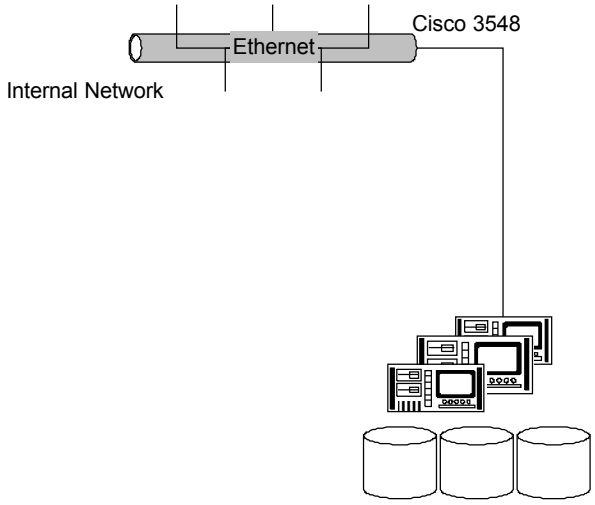
Infrastructure Servers
10.1.3.16 - 10.1.3.31
255.255.255.240

The system administrator workstations are grouped together into a small subnet range. These workstations will need privileged access to machines outside of the internal LAN. By segregating them into their own subnet, we can simplify any access control lists that need to restrict system administrator access to these individuals.

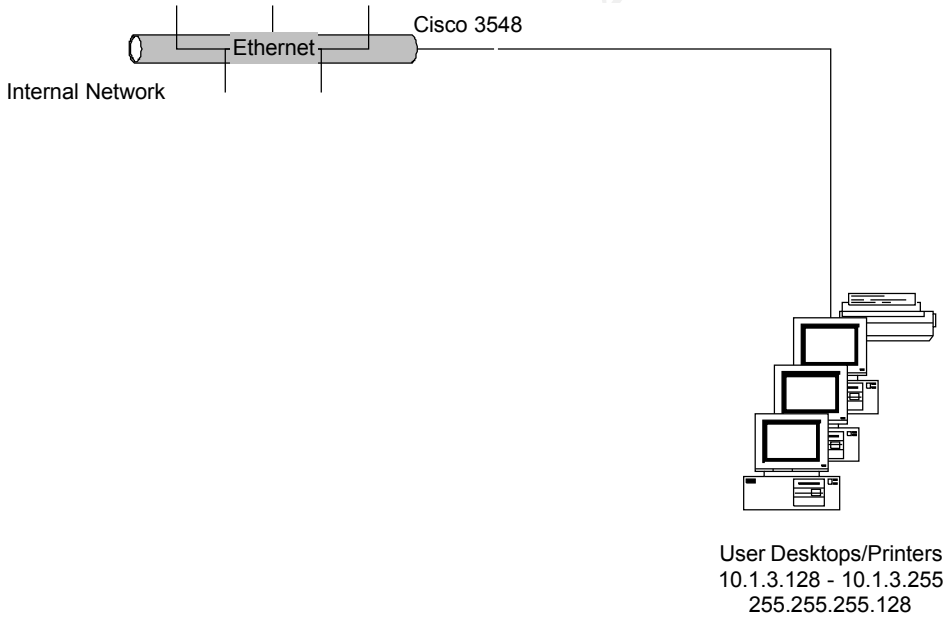


System Administrators
10.1.3.64 - 10.1.3.79
255.255.255.240

The internal application and database servers are grouped together into a subnet. GIAC users and the business partner access these servers to perform their daily duties. Applications provide interfaces to various databases as dictated by the business needs. These servers are segregated in order to simplify access control lists that limit traffic to and from these servers. The value of the servers makes them obvious candidates for further segmentation and traffic controls in the future.



The last internal subnet contains the general user desktop population. Other general user devices such as printers, scanners, and copiers are also located here. Netware file services are also located here to support desktop file sharing. As the business and network grows, this subnet may also be segmented onto its own network.



Security Component Descriptions

Border Router

The Cisco 3640 series router provides the connection to the Internet. This router is in the upper end of Cisco's midrange offerings. It supports up to four network modules. Each network module can support a variety of multi-port combinations so we will have room to expand in the future since we will initially configure it with only T1 and Ethernet connections. This router is capable of handling several WAN and LAN connections. It provides plenty of horsepower at a reasonable cost. Product literature and documentation at Cisco's web site can be found at: http://www.cisco.com/warp/public/cc/pd/rt/3600.

The router is running Cisco IOS Version 12.2(1b). The primary job of the border router will be to route packets. We will not be configuring extensive access control lists on the router since we have two firewalls that will handle the task of controlling specific server-level access. The router will be configured with some very basic traffic controls such as ingress/egress filtering, rejection of private IP addresses, and rejection of a few insane source addresses. It will also be hardened. Access controls will be implemented to restrict administrative access to the router.

Border Firewall

The Cisco PIX 515 provides the primary firewall services at the border. It is a stateful firewall that has a proven track record for reliability and performance. According to Cisco's web site http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/prodlit/pix51_ds.htm: "The PIX 515-UR (unrestricted) provides all the functionality of the PIX 515-R-BUN, as well as fail-over functionality, and up to six 10/100 Ethernet ports. These four additional ports allow for more robust traffic configurations as well as hosting a protected DMZ for hosting a Web site or performing URL filtering and virus detection. Designed for mid-sized organizations, the PIX 515 UR provides nearly 170 Mbps of throughput and over 100,000 concurrent connections for fast, reliable, and affordable security protection."

The firewall is running Secure PIX Firewall version 6.1(1). The PIX is configured to control all of the traffic into and out of the GIAC network. It also provides all of the VPN services that are required. Although the initial configuration uses only two Ethernet interfaces, future expansions could include utilizing additional outside interfaces to support multiple Internet connections. It is also capable of providing stateful failover when configured in pairs, if future business needs demand it.

Internal Firewall

A Compaq Proliant DL380 running Red Hat Linux 7.1 with IPTables provides the internal firewall services. The server is configured with a 1.0 GHz Intel P-III processor, 1.0GB of RAM, a pair of 18GB 10K rpm hard drives using RAID-1, and four Ethernet interfaces. The server is

expandable to allow another pair of hard drives, an additional 3GB of RAM, another processor, and additional disk controller cache. Faster and larger hard drives are also available. The initial configuration is already powerful enough, with plenty of room to grow. Further information can be found at Compaq's web site: http://www.compaq.com/products/servers/proliantdl380/index.html.

This firewall serves two basic purposes: a second layer of firewall screening for defense-in-depth, and traffic control between the various internal and service nets. Calling this machine an "internal" firewall is somewhat of a misnomer. It is within the primary border protection, but it is not controlling the traffic on the internal LAN. It controls traffic flows into and out of the various networks that are operating at differing levels of possible hostility. This firewall provides a convenient choke point to control necessary traffic flows between these different networks.

Ethernet Switches

Cisco series 2950-12 and 3548 series switches provide the Ethernet networks all throughout the organization. Switches are selected instead of hubs because they provide higher performance and they make sniffing network traffic difficult for attackers. The 2950-12 switches support up to 12 connections. These high-speed switches only need to support 3 or 4 connections each on the small networks in the perimeter, so they provide room to grow. The internal LAN will have greater requirements in terms of the number of connections and volume of traffic. The 3548 series switch can handle these increased requirements. Documentation at Cisco's web site can be found at: (2950-12) http://www.cisco.com/warp/public/cc/pd/si/casi/ca2950 and (3548) http://www.cisco.com/warp/public/cc/pd/si/casi/ca3500xl.

Intrusion Detection Systems

Three IDS sensors are installed on the network. Ideally, the public service net and the VPN service net should have sensors installed also, but the budget will not permit the extra expense. The three sensors are all Compaq Proliant servers. The server connected between the border router and PIX firewall needs to be configured with a higher disk storage performance than the inside servers since it captures and saves a portion of many of the packets traversing the switch. The two inside sensors are configured to alert and save particular traffic signatures, so we can scale down their hardware configuration somewhat. Further information at Compaq's web site can be found at: http://www.compaq.com/products/servers/proliantdl380/index.html

Outside sensor: Proliant DL380, 1.0 GHz Intel P-III, 512MB RAM,
(4) 18GB 15K rpm hard drives using RAID-1+0, 128MB raid cache
Inside sensors: Proliant DL380, 1.0 GHz Intel P-III, 256MB RAM,
(2) 18GB 10K rpm hard drives using RAID-1

All three sensors run Linux Red Hat 7.1. The operating systems are installed with a bare minimum of RPM packages since they only need to capture network traffic. The network interfaces that are connected to the switch for sniffing do not have IP addresses assigned and they are running in promiscuous mode.

The outside sensor runs tcpdump and it captures the first 134 bytes of particular traffic. This allows for a 14-byte Ethernet header, a maximum 60-byte IP header, and a maximum 60-byte TCP header. Any ICMP or UDP headers would be shorter than the 60-byte TCP header. Traffic excluded from the captures would be the high volume traffic that we permit through the firewall such as ports 80 and 443 to the web server, 25 to the mail server, etc. The purpose of the packet header collections is to be able to monitor reconnaissance, and to reconstruct exploit techniques if any are successful. Sensors on the inside of the firewall often cannot provide the information necessary to determine how someone crafted packets that succeeded in traversing your firewall. These log files can grow quickly, so an hourly process compresses a copy of the log and clears it. The logs will be retained on the disk for a few days and then taken off to backup tapes.

The sensor between the two firewalls will alert on suspicious traffic within the border, and capture the traffic to disk. This may be someone attempting to exploit a permitted service such as DNS, or it may be a denied service that slipped through the border. The sensor on the internal LAN is for critical alerts. This warns us about possible malicious traffic that made it past all of our perimeter protection. These sensors both run the Snort IDS, and the base rule sets will be customized to match our environment. The logs on both of these sensors are periodically reviewed, backed up, and removed. Although the initial design does not include it, they could be optionally configured with a phone line (dial-out only!) to send pager alerts to the administrators.

Supplier and Customer VPN Access

The suppliers and customers will use a VPN connection to access separate web servers on a dedicated service network. The fortune files do not require encrypted connections for transmission since it is not sensitive data, but we do want to ensure that the connections are authentic, so IPSEC AH will be used to establish the connection. The traffic will be primarily HTTP and FTP for navigating through the web site and transferring files. We will also allow HTTPS for transmission of any sensitive information such as a login credentials, and the web application will switch between the secure and non-secure modes as required.

The number of customers and suppliers is limited, so using access control lists and shared secret keys to permit the IPSEC AH connections will suffice. This does not scale well, so if the customer and supplier base grows too large, we may need to allow connections from anywhere and switch to a PKI based key exchange to control access.

Business Partner Access

The merger with a business partner in China creates a business need for the partner to have access to some of our internal network. The partner needs access to the internal DNS server, and to the subnet that contains the applications and databases. The applications run on web and Citrix based protocols (HTTP, HTTPS, and ICA), so these protocols are opened through the firewall from the partner's IPSEC gateway. The connection needs to be encrypted since sensitive information will be transmitted, so IPSEC ESP is used. Since the partner is located in China, I am going to assume that triple-DES encryption is not legal, and regular DES will be used.

Employee Remote Access

Occasionally, employees need to access the internal network from home or on the road. This access is not lengthy or frequent, but it does occur. System administrators may need to correct a problem, or sales persons may need to access resources while marketing GIAC services. The employees are permitted access to the entire internal network, just like they do when they are on site. Since sensitive information is accessed, IPSEC ESP using triple-DES encryption is used. The Cisco desktop client v3 is deployed on the users' machines that require access. Since the same Internet connection is used for both the inbound and outbound traffic, remote users would not be able to browse the Internet while connected. This would generate traffic that attempts to enter the PIX firewall outside interface and exit the same interface. This is a PIX policy violation and is not permitted. In order to permit Internet browsing while connected, a split tunnel configuration is used. This is unfortunate, and if GIAC decides to install another Internet connection in the future, it could be possible to separate the inbound and outbound traffic onto different interfaces or different firewalls and allow the browsing without the use of a split tunnel. Theoretically, the split tunnel opens the possibility for sensitive information to be captured locally on the user's PC and transmitted back to an attacker. To mitigate this exposure, the PCs will use locally installed firewall and virus scanning products. Although it is not an option for the traveling sales team, the home users will also use a hardware based home firewall product for additional protection.

Public and VPN Service Net Hosts

Although the primary focus of this practical is on the network design, border router, firewalls, and VPN technologies, some configuration notes on the service net hosts is warranted since they are frequently the focus of attacks.

These hosts are running Red Hat 7.1. The Unix administrator hardens the boxes and they are running the absolute minimum of required services. TCP wrappers is employed to control access to running services, and they all run syslog services to report system activity back to the internal centralized syslog server. The administrators remain aware of all security bulletins from CERT,

SANS, Security Focus, and Red Hat. The systems are patched quickly when vulnerabilities are published.

The public mail server is running Sendmail version 8.11.6. Its configuration is hardened to prevent spamming and to hide the internal mail server identity. The DNS server is running ISC BIND version 8.2.3. Its configuration is hardened to prevent misuse such as recursive queries for non-GIAC hosts and queries for the BIND version.

Assignment 2: Security Policy

Border Router

Comments

The configuration below uses a private address of 172.16.1.2 representing an address that the ISP would assign to our router's outside interface. The inside interface uses an address of 99.99.1.2 representing one of the public IP addresses assigned to GIAC Enterprises. Neither of these addresses is a legal public address on the Internet. They were used in my home lab for testing purposes. Since I used a private address on the outside interface and connected testing equipment in this outside private range, the ACL that would normally block this address space was commented (!) from my test configuration, but would be included in a production router configuration that uses legal public addresses.

Border Router Policy Summary

Action	Proto	Source	Destination	Service	Comments
		Inbound Policy			
Deny	ICMP	Any	Any		No ICMP
Deny	IP	99.99.1.0/255.255.255.224	Any		Ingress
Deny	IP	10.0.0.0/255.255.255.0	Any		RFC 1918
Deny	IP	192.168.0.0/255.255.0.0	Any		RFC 1918
Deny	IP	169.254.0.0/255.255.0.0	Any		Reserved
Deny	IP	127.0.0.0/255.255.255.0	Any		Loopback
Deny	IP	0.0.0.0	Any		Insane
Deny	IP	255.255.255.255	Any		Insane
Permit	TCP	Any	99.99.1.0/255.255.255.224		
Permit	UDP	Any	99.99.1.0/255.255.255.224		
Permit	ESP	Any	99.99.1.3		VPN
Permit	AH	Any	99.99.1.3		VPN
Deny	IP	Any	Any		Catch-all
		Oubound Policy			
Permit	Any	99.99.1.0/255.255.255.224	Any		Egress
Permit	Any	10.1.3.16/255.255.255.240	Any		Router access
Permit	Any	10.1.3.64/255.255.255.240	Any		Router access

Deny	Any	Any	Any		Catch-all
		Special Policy			
Permit	Any	10.1.3.64/255.255.255.240	Router Inside VTY	SSH	Admins
Deny	Any	Any	Router Inside VTY	Any	Catch-all

Loading the Encryption Software Image

This configuration enables SSH access for the administrators. In order for SSH to work, the IPSEC (DES or 3DES) encryption software image must be loaded. The following shows the commands and the screen messages from the O/S image loading process. I omitted some of the extraneous screen output with Cisco's address, and other irrelevant output that does not pertain to the technical aspects of the loading process.

```
carlton>enable
Password:
carlton# copy tftp://10.1.3.20/c2500-ik8s-l.122-1b.bin flash:
                  ****  NOTICE  ****
Flash load helper v1.0
This process will accept the copy options and then terminate
the current system image to use the ROM based image for the copy.
Routing functionality will not be available during that time.
If you are logged in via telnet, this connection will terminate.
Users with console access can see the results of the copy operation.
                  ---- ******** ----
Proceed? [confirm]
Destination filename [c2500-ik8s-l.122-1b.bin]?
Accessing tftp://10.1.3.20/c2500-ik8s-l.122-1b.bin...
Erase flash: before copying? [confirm]

00:33:31: %SYS-5-RELOAD: Reload requested
%SYS-4-CONFIG_NEWER: Configurations from version 12.2 may not be correctly understood.
%FLH: c2500-ik8s-l.122-1b.bin from 10.1.3.20 to flash ...

System flash directory:
File  Length   Name/status
  1   8833168  c2500-i-l.122-1b.bin
[8833232 bytes used, 7943984 available, 16777216 total]
Accessing file 'c2500-ik8s-l.122-1b.bin' on 10.1.3.20...
Loading c2500-ik8s-l.122-1b.bin from 10.1.3.20 (via Ethernet1): ! [OK]
```

```
Erasing device... eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
ee ...erased
Loading c2500-ik8s-l.122-1b.bin from 10.1.3.20 (via Ethernet1): !!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 13151248/16777216 bytes]

Verifying checksum...
Flash copy took 0:06:27 [hh:mm:ss]
%FLH: Re-booting system after download
F3: 12494360+656856+1114804 at 0x3000060

Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-IK8S-L), Version 12.2(1b), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Fri 15-Jun-01 06:53 by pwade
Image text-base: 0x030654A8, data-base: 0x00001000

Compliance with U.S. Export Laws and Regulations - Encryption

This product performs encryption and is regulated for export
by the U.S. Government.

Press RETURN to get started!
```

Border Router Configuration Tutorial

When you initially configure the router, ensure that it is not connected to the external communications line. The router must be completely configured before you allow any connections to it or malicious traffic may pass through before you are completed configuring it.

Connect a serial line to the console port on the router. Use a PC to connect to the router. I used Microsoft's HyperTerminal for an interface program. You will see the prompt "carlton" in the following commands since that is my router's name.

Enter privileged mode so that you are able to execute configuration commands.

```
carlton>enable
Password:
```


Enter configuration mode.

```
carlton# config t
Enter configuration commands, one per line.  End with CNTL/Z.
```

Set your passwords. The password is used to initially connect to the router. The secret is used to gain access to administrator (enable) mode.

```
carlton(config)# enable secret 0 xxxxxxxx
carlton(config)# enable password 0 xxxxxxxx
```

Set your passwords on the console lines for when you connect to the router via the console port.

```
carlton(config)#line con 0
carlton(config-line)#password 0 xxxxxxxx
carlton(config-line)#transport input none
carlton(config-line)#exit
carlton(config)#line aux 0
carlton(config-line)#password 0 xxxxxxxx
carlton(config-line)#transport input none
carlton(config-line)#exit
carlton(config)#exit
carlton(config)#
```

Name the router. This is required for SSH configuration.

```
carlton(config)# hostname carlton
carlton(config)# ip domain-name giac.com
```

Set the timezone.

```
carlton(config)# clock timezone edt -5
```

Configure the router to add timestamps to debugging and logging messages.

```
carlton(config)# service timestamps debug uptime
carlton(config)# service timestamps log uptime
```

Enable password encryption.

```
carlton(config)# service password-encryption
```

Define the access control list for the inbound traffic on the outside interface. The order is important: 1) all of the source address denies 2) all of the permits 3) final 'catch all' deny
Define the access control list as extended and name it.

```
carlton(config)# ip access-list extended e0_acl_inbound
```

Deny all ICMP traffic on the outside interface and log it.

```
carlton(config-ext-nacl)# deny   icmp any any log
```

Deny inbound spoofed source addresses (ingress filtering). This includes the GIAC public addresses, which should only originate on the inside of the router, private RFC 1918 addresses,

reserved 169.254.x space, and few common insane addresses such as the loopback, all zeroes, and all ones (255). Include the log option so that we have log entries for suspicious traffic.

```
carlton(config-ext-nacl)# deny   ip 99.99.1.0 0.0.0.31 any log
carlton(config-ext-nacl)# deny   ip 10.0.0.0 0.255.255.255 any log
carlton(config-ext-nacl)# deny   ip 172.16.0.0 0.15.255.255 any log
carlton(config-ext-nacl)# deny   ip 192.168.0.0 0.0.255.255 any log
carlton(config-ext-nacl)# deny   ip 169.254.0.0 0.0.255.255 any log
carlton(config-ext-nacl)# deny   ip 127.0.0.0 0.255.255.255 any log
carlton(config-ext-nacl)# deny   ip host 0.0.0.0 any log
carlton(config-ext-nacl)# deny   ip host 255.255.255.255 any log
```

Permit TCP and UDP traffic destined for our public addresses.

```
carlton(config-ext-nacl)# permit tcp any 99.99.1.0 0.0.0.31
carlton(config-ext-nacl)# permit udp any 99.99.1.0 0.0.0.31
```

Permit IPSEC protocols ESP and AH for VPN access to the PIX external interface.

```
carlton(config-ext-nacl)# permit esp any host 99.99.1.3
carlton(config-ext-nacl)# permit ahp any host 99.99.1.3
```

Deny the rest of the garbage and log it.

```
carlton(config-ext-nacl)# deny   ip any any log
```

Issue an "exit" to complete the list.

```
carlton(config-ext-nacl)# exit
carlton(config)#
```

Define a standard access control list for the traffic on the inside interface. Permit only our addresses from the inside to prevent us from possibly sending spoofed packets.

```
carlton(config)# access-list 25 permit 99.99.1.0 0.0.0.31
carlton(config)# access-list 25 permit 10.1.3.16 0.0.0.15
carlton(config)# access-list 25 permit 10.1.3.64 0.0.0.15
carlton(config)# access-list 25 deny   any log
```

Define a standard access control list for administrative access to the router from the system administrator's internal subnet range.

```
carlton(config)# access-list 10 permit 10.1.3.64 0.0.0.15 log
carlton(config)# access-list 10 deny   any log
```

Configure the outside interface.
Enter the configuration mode for the interface.

```
carlton(config)# interface Ethernet0
```

Assign the IP address to the interface.

```
carlton(config-if)# ip address 172.16.1.2 255.255.255.0
```

Apply the access list that was defined for the outside interface.

```
carlton(config-if)# ip access-group e0_acl_inbound in
```

Disable undesirable ICMP traffic, Cisco Discovery Protocol (CDP), and broadcasts.

```
carlton(config-if)# no ip redirects
carlton(config-if)# no ip unreachables
carlton(config-if)# no ip proxy-arp
carlton(config-if)# no cdp enable
carlton(config-if)# no ip directed-broadcast
carlton(config-if)# no ip mask-reply
```

Issue an exit to complete the interface configuration.

```
carlton(config-if)# exit
carlton(config)#
```

Configure the inside interface.
Enter the configuration mode for the interface.

```
carlton(config)# interface Ethernet1
```

Assign the IP address to the interface.

```
carlton(config-if)# ip address 99.99.1.2 255.255.255.240
```

Apply the access list that was defined for the outside interface.

```
carlton(config-if)# ip access-group 25 in
```

Limit the telnet access to the inside interface.

```
carlton(config)# ip telnet source-interface Ethernet1
```

Disable undesirable ICMP traffic, Cisco Discovery Protocol (CDP), and broadcasts.

```
carlton(config-if)# no ip redirects
carlton(config-if)# no ip unreachables
carlton(config-if)# no ip proxy-arp
carlton(config-if)# no cdp enable
carlton(config-if)# no ip directed-broadcast
carlton(config-if)# no ip mask-reply
```

Issue an exit to complete the interface configuration.

```
carlton(config-if)# exit
carlton(config)#
```

Disable any interfaces that are not in use.

```
carlton(config)# interface Serial0
carlton(config-if)# no ip address
carlton(config-if)# shutdown
carlton(config-if)# no cdp enable
carlton(config-if)# exit
carlton(config)# interface Serial1
carlton(config-if)# no ip address
carlton(config-if)# shutdown
carlton(config-if)# no cdp enable
carlton(config-if)# exit
```

Configure default and static routes.

```
carlton(config)# ip default-gateway 172.16.1.1
carlton(config)# ip classless
carlton(config)# ip route 0.0.0.0 0.0.0.0 172.16.1.1
carlton(config)# ip route 10.1.3.0 255.255.255.0 99.99.1.3 permanent
carlton(config)# ip route 99.99.1.24 255.255.255.248 99.99.1.3 permanent
```

Disable source routing.

```
carlton(config)# no ip source-route
```

Configure logging options. Limit the logging rate to 10 messages per second. I like to limit the console messages to the "warning" level, and set a more verbose "informational" level for the syslog. Your preferences may vary. Send the logs out the inside interface to the syslog server IP address.

```
carlton(config)# logging rate-limit all 10
carlton(config)# logging console warnings
carlton(config)# logging trap informational
carlton(config)# logging facility syslog
carlton(config)# logging source-interface Ethernet1
carlton(config)# logging 10.1.3.22
```

Disable services that the router does not need. These services make the router vulnerable to attack.

```
carlton(config)# no service tcp-small-servers
carlton(config)# no service udp-small-servers
carlton(config)# no service config
carlton(config)# no ip finger
carlton(config)# no ip domain-lookup
carlton(config)# no ip bootp server
carlton(config)# no ip dhcp-client network-discovery
carlton(config)# no snmp
carlton(config)# no ip http server
carlton(config)# no ip rcmd rcp-enable
carlton(config)# no ip rcmd rsh-enable
```

```
carlton(config)# no cdp run
```

Optionally configure NTP services.

```
carlton(config)# ntp source Ethernet1
carlton(config)# ntp server 10.1.3.20
```

Configure a banner to discourage unauthorized access.

```
carlton(config)# banner motd /
Enter TEXT message.  End with the character '/'.
WARNING: Authorized access only.
Unauthorized persons will be prosecuted to the fullest extent of the law.
/
carlton(config)#
```

Generate the RSA key needed for SSH connections.

```
carlton(config)# crypto key generate rsa
The name for the keys will be: carlton.giac.com
% You already have RSA keys defined for carlton.giac.com.
% Do you really want to replace them? [yes/no]: yes
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.
How many bits in the modulus [512]:
1d03h: %SSH-5-DISABLED: SSH 1.5 has been disabled
Generating RSA keys ...
[OK]
carlton(config)#
1d03h: %SSH-5-ENABLED: SSH 1.5 has been enabled
carlton#
```

Configure SSH. Enable the AAA authentication model and set the login password to the line password. Set the SSH timeout and retry limit. Configure the virtual terminal lines to only permit access to the ACL that we defined for administrator access, and use SSH as the transport.

```
carlton(config)# aaa new-model
carlton(config)# aaa authentication login default line
carlton(config)# ip ssh time-out 120
carlton(config)# ip ssh authentication-retries 3
carlton(config)# line vty 0 4
carlton(config-line)# access-class 10 in
carlton(config-line)# password 0 xxxxxxxx
carlton(config-line)# transport input ssh
carlton(config-line)# exit
carlton(config)#
```

Save the configuration and exit.

```
carlton# write mem
Building configuration...
[OK]
carlton# copy run start
Destination filename [startup-config]?
Building configuration...
[OK]
carlton# exit
```

Border Router Configuration File

The completed border router configuration load file is shown below.

```
!
version 12.2
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
service password-encryption
no service tcp-small-servers
no service udp-small-servers
no service config
!
hostname carlton
!
boot system flash
logging rate-limit all 10
logging console warnings
aaa new-model
aaa authentication login default line
enable secret 5 xxxxxxxx
enable password 7 xxxxxxxx
!
clock timezone edt -5
ip subnet-zero
no ip source-route
no ip finger
ip telnet source-interface Ethernet1
no ip domain-lookup
ip domain-name giac.com
!
no ip bootp server
no ip dhcp-client network-discovery
ip ssh time-out 120
ip ssh authentication-retries 3
no snmp
!
interface Ethernet0
 ip address 172.16.1.2 255.255.255.0
 ip access-group e0_acl_inbound in
 no ip redirects
```

```
 no ip unreachables
 no ip proxy-arp
 no cdp enable
 no ip directed-broadcast
 no ip mask-reply
!
interface Ethernet1
 ip address 99.99.1.2 255.255.255.240
 ip access-group 25 in
 no ip redirects
 no ip unreachables
 no ip proxy-arp
 no cdp enable
 no ip directed-broadcast
 no ip mask-reply
!
interface Serial0
 no ip address
 shutdown
 no cdp enable
!
interface Serial1
 no ip address
 shutdown
 no cdp enable
!
ip default-gateway 172.16.1.1
ip classless
ip route 0.0.0.0 0.0.0.0 172.16.1.1
ip route 10.1.3.0 255.255.255.0 99.99.1.3 permanent
ip route 99.99.1.24 255.255.255.248 99.99.1.3 permanent
no ip http server
no ip rcmd rcp-enable
no ip rcmd rsh-enable
no ip access-list extended e0_acl_inbound
ip access-list extended e0_acl_inbound
 deny   icmp any any log
 deny   ip 99.99.1.0 0.0.0.31 any log
 deny   ip 10.0.0.0 0.255.255.255 any log
! deny   ip 172.16.0.0 0.15.255.255 any log
 deny   ip 192.168.0.0 0.0.255.255 any log
 deny   ip 169.254.0.0 0.0.255.255 any log
 deny   ip 127.0.0.0 0.255.255.255 any log
 deny   ip host 0.0.0.0 any log
 deny   ip host 255.255.255.255 any log
 permit tcp any 99.99.1.0 0.0.0.31
 permit udp any 99.99.1.0 0.0.0.31
 permit esp any host 99.99.1.3
 permit ahp any host 99.99.1.3
 deny   ip any any log
logging trap informational
logging facility syslog
logging source-interface Ethernet1
logging 10.1.3.22
no access-list 10
access-list 10 permit 10.1.3.64 0.0.0.15 log
access-list 10 deny   any log
no access-list 25
```

```
access-list 25 permit 99.99.1.0 0.0.0.31
access-list 25 permit 10.1.3.16 0.0.0.15
access-list 25 permit 10.1.3.64 0.0.0.15
access-list 25 deny   any log
no cdp run
banner motd ^C
WARNING: Authorized access only.
Unauthorized persons will be prosecuted to the fullest extent of the law.
^C
!
line con 0
 password 7 xxxxxxxx
 transport input none
line aux 0
 password 7 xxxxxxx
line vty 0 4
 access-class 10 in
 password 7 xxxxxxxx
 transport input ssh
!
ntp source Ethernet1
ntp server 10.1.3.20
end
```

Border Firewall

Comments

The outside interface uses addresses of 99.99.1.x representing the public IP addresses assigned to GIAC Enterprises. These are not legal public address on the Internet. They were used in my home lab for testing purposes. Since I used a private 172.16.1.x address on the outside router interface and connected testing equipment in this outside private range, the ACL that would normally block this address space was commented (!) from my test configuration, but would be included in a production firewall configuration that uses legal public addresses.

The firewall configuration below uses public addresses of 202.120.25.x, 202.100.75.x, and 207.46.230.x representing addresses for the partner, supplier, and customer. These are addresses that I chose as examples simply because the 202.x ranges were listed at APNIC as China registrations, and the 207.x was listed at ARIN as a U.S registration (Microsoft). They were used in my home lab for testing purposes and have no significance otherwise.

Border Firewall Policy Summary

Action	Proto	Source	Destination	Service	Comments
		Inbound Policy			

Permit	UDP	99.99.1.2	10.1.3.22	SYSLOG	Before ingress deny
Permit	UDP	99.99.1.2	10.1.3.20	NTP	Before ingress deny
Permit	UDP	99.99.1.2	10.1.3.20	TFTP	Before ingress deny
Deny	IP	Any	10.0.0.0/255.0.0.0		RFC 1918
Deny	IP	10.0.0.0/255.0.0.0	Any		RFC 1918
Deny	IP	Any	192.168.0.0/255.255.0.0		RFC 1918
Deny	IP	192.168.0.0/255.255.0.0	Any		RFC 1918
Deny	IP	Any	172.16.0.0/255.255.240.0		RFC 1918
Deny	IP	172.16.0.0/255.255.240.0	Any		RFC 1918
Deny	IP	127.0.0.0/255.0.0.0	Any		Loopback
Deny	IP	169.254.0.0/255.255.0.0	Any		Reserved
Deny	IP	0.0.0.0	Any		Insane
Deny	IP	255.255.255.255	Any		Insane
Deny	IP	224.0.0.0/224.0.0.0	Any		Multicast
Deny	IP	1.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	2.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	23.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	31.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	67.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	68.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	69.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	70.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	71.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	72.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	73.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	74.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	75.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	76.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	77.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	78.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	79.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	80.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	81.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	82.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	83.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	84.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	85.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	86.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	87.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	88.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	89.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	90.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	91.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	92.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	93.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	94.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	95.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	96.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	97.0.0.0/255.255.255.0	Any		Reserved

Deny	IP	98.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	99.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	100.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	101.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	102.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	103.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	104.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	105.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	106.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	107.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	108.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	109.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	110.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	111.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	112.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	113.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	114.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	115.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	126.0.0.0/255.255.255.0	Any		Reserved
Deny	IP	191.255.0.0/255.255.0.0	Any		Reserved
Deny	IP	192.0.2.0/255.255.255.0	Any		Reserved
Deny	IP	201.0.0.0/255.255.192.0	Any		Reserved
Deny	IP	223.255.255.0/255.255.255.0	Any		Reserved
Deny	IP	99.99.1.0/255.255.255.224	Any		Ingress
Permit	TCP	Any	99.99.1.5	SMTP	Mail
Permit	UDP	Any	99.99.1.6	DNS	Lookups
Permit	TCP	Any	99.99.1.7	HTTP	Web
Permit	TCP	Any	99.99.1.7	HTTPS	SSL
Deny	IP	Any Inbound	Any		Catch-All
		Outbound Policy			
Deny	IP	99.99.1.0/255.255.255.240	Any		Egress
Deny	IP	99.99.1.16/255.255.255.248	Any		Egress
Permit	TCP	10.1.2.5	Any	SMTP	Mail
Permit	UDP	10.1.2.6	Any	DNS	Lookups
Permit	UDP	10.1.2.5	128.118.25.3	NTP	Time
Permit	UDP	10.1.2.5	132.236.56.250	NTP	Time
Permit	UDP	10.1.2.5	198.82.162.213	NTP	Time
Permit	TCP	10.1.3.128/255.255.255.128	Any	HTTP	Web
Permit	TCP	10.1.3.128/255.255.255.128	Any	HTTPS	Web
Permit	TCP	10.1.3.128/255.255.255.128	Any	FTP	File transfer
Permit	TCP	10.1.3.64/255.255.255.248	Any	HTTP	Web
Permit	TCP	10.1.3.64/255.255.255.248	Any	HTTPS	SSL
Permit	TCP	10.1.3.64/255.255.255.248	Any	FTP	File transfer
Permit	TCP	10.1.3.64/255.255.255.248	99.99.1.2	SSH	Admin
Permit	TCP	10.1.1.5	10.3.2.23	RADIUS	Remote auth
Permit	UDP	10.1.1.5	10.3.2.23	RADIUS	Remote auth
Deny	IP	Any Inbound	Any		Catch-All
		NAT Policy			

		10.1.3.64/255.255.255.248	99.99.1.2	No NAT	Router Admn
		Any	10.99.1.0/255.255.255.248	No NAT	Emp VPN
		Any	202.120.25.64/255.255.255.224	No NAT	Partner VPN
		Any	202.100.75.32/255.255.255.224	No NAT	Supplier VPN
		Any	207.46.230.192/255.255.255.224	No NAT	Customer VPN
		10.1.3.64/255.255.255.248	Any	NAT	Outbound Users
		10.1.3.128/255.255.255.128	Any	NAT	Outbound Users
		IPSEC Policy			
Permit	IP	10.1.3.96/255.255.255.224	202.120.25.64/255.255.255.224	ESP des	Partner VPN
Permit	IP	10.1.3.16/255.255.255.240	202.120.25.64/255.255.255.224	ESP des	Partner VPN
Permit	IP	99.99.1.25	202.100.75.32/255.255.255.224	AH	Supp VPN
Permit	IP	99.99.1.26	207.46.230.192/255.255.255.224	AH	Cust VPN
Permit	IP	10.1.2.0/255.255.255.0	10.99.1.0/255.255.255.0	ESP 3des	Emp VPN
Permit	IP	10.1.3.0/255.255.255.0	10.99.1.0/255.255.255.0	ESP 3des	Emp VPN
Permit	IP	99.99.1.0/255.255.255.224	10.99.1.0/255.255.255.0	ESP 3des	Emp VPN

Loading the Encryption Software Key

This configuration enables SSH access for the administrators and IPSEC for VPN access. In order for these encryption technologies to work, the encryption software must be activated with a license key. This key is entered when you load the operating system. The following shows the commands and the screen messages from the O/S image loading process. I omitted some of the extraneous screen output with Cisco's address, and other irrelevant output that does not pertain to the technical aspects of the loading process.

I powered on the PIX, and then the following occurred.

Cisco Secure PIX Firewall BIOS (4.0) #0: Fri Mar 31 11:32:03 PST 2000
Platform PIX-506
Flash=i28F640J5 @ 0x300

Use BREAK or ESC to interrupt flash boot.
Use SPACE to begin flash boot immediately.
I hit the ESC key.
Flash boot interrupted.

```
0: i8255X @ PCI(bus:0 dev:13 irq:11)
1: i8255X @ PCI(bus:0 dev:14 irq:10)

Using 1: i82559 @ PCI(bus:0 dev:14 irq:10), MAC: 0005.3290.0e89
Use ? for help.
monitor> interface 1
0: i8255X @ PCI(bus:0 dev:13 irq:11)
1: i8255X @ PCI(bus:0 dev:14 irq:10)

Using 1: i82559 @ PCI(bus:0 dev:14 irq:10), MAC: 0005.3290.0e89
address 10.1.1.5
monitor> server 10.1.3.20
server 10.1.3.20
monitor> file pix611.bin
file pix611.bin
monitor> tftp
tftp pix611.bin@10.1.3.20.....
.....
.....
.....
.....
Received 2562048 bytes

Cisco Secure PIX Firewall admin loader (3.0) #0: Thu May 17 19:55:23 PDT 2001
Flash=i28F640J5 @ 0x300
BIOS Flash=AT29C257 @ 0xd8000
Flash version 6.1.1, Install version 6.1.1
Do you wish to copy the install image into flash? [n] y

Installing to flash

Serial Number: XXXXXXXXX (0xXXXXXX)
Activation Key: xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx

Do you want to enter a new activation key? [n]y
Enter Activation Key
   Part 1 of 4: xxxxxxxx
   Part 2 of 4: xxxxxxxx
   Part 3 of 4: xxxxxxxx
   Part 4 of 4: xxxxxxxx

Writing image into flash...
32MB RAM
Flash=i28F640J5 @ 0x300
BIOS Flash=AT29C257 @ 0xfffd8000
```

```
mcwa i82559 Ethernet at irq 11  MAC: 0005.3290.0e88
mcwa i82559 Ethernet at irq 10  MAC: 0005.3290.0e89

 ------------------------------------------------------------------------
                          ||        ||
                          ||        ||
                         ||||      ||||
                       ..:||||||:...:||||||:..
                       c i s c o S y s t e m s
                      Private Internet eXchange
 ------------------------------------------------------------------------
                      Cisco Secure PIX Firewall

Cisco Secure PIX Firewall Version 6.1(1)

Licensed Features:
Failover:       Disabled
VPN-DES:        Enabled
VPN-3DES:       Enabled
Maximum Interfaces:     2
Cut-through Proxy:      Enabled
Guards:         Enabled
Websense:       Enabled
Throughput:     Limited
ISAKMP peers:   Unlimited

 ********************************* Warning *********************************
 Compliance with U.S. Export Laws and Regulations - Encryption.

 This product performs encryption and is regulated for export
 by the U.S. Government.
…
…
Cryptochecksum(changed): xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
Type help or '?' for a list of available commands.
spock>
```

Border Firewall Configuration Tutorial

Comments

When you initially configure the firewall, ensure that it is not connected to the external

communications line. The firewall must be completely configured before you allow any connections to it or malicious traffic may pass through before you are completed configuring it.

Connect a serial line to the console port on the firewall. Use a PC to connect to the firewall. I used Microsoft's HyperTerminal for an interface program. You will see the prompt "spock" in the following commands since that is my firewall's name.

There are many ACLs that are configured on this firewall. For instructional purposes, I am not going to show every entry in the tutorial below. An explanation of the purpose of the ACL traffic controls is illustrated in a following section. I will also skip miscellaneous default values such the screen pager display length, or the arp timeout value. I will concentrate on the methods used to configure the firewall, particularly in the IPSEC configuration since the commands themselves are not clearly self-documenting.

Caveats

The IPSEC implementation on the PIX does not handle traffic filtering as well as one would hope. IPSEC traffic bypasses the regular access lists that are applied to the outside interface. Separate access lists are created to handle the IPSEC traffic, but they only affect the outbound half of the traffic flow. Furthermore, when you attempt to filter IPSEC traffic on particular TCP or UDP ports, you receive the following warning:

WARNING: access-list has port selectors may have performance impact

I found another interesting curiosity in VPN client implementation using the required dynamic crypto maps. If you use any access lists, the initial connection negotiation requires a rule allowing traffic from the firewall's interface address to the VPN client address. This is not documented, but that may be because the VPN client documentation does not show real-world access lists to control the traffic flow from the client.

I did a lot of experimenting with the access lists, and I could not get them to filter IPSEC inbound. Conveniently, the Cisco documentation examples do not show very much filtering. They show complete network-to-network access lists, and they do not show tight address/service level filtering. Considering the short time that effective VPNs have been offered by vendors, this may just be a growing pain that we need to tolerate until the implementations improve.

I was uncomfortable forcing TCP and UDP port level filtering considering the performance warning and the odd behaviors that I observed while experimenting with Cisco's IPSEC. Instead, I elected to configure the VPN connections on a simple level that is consistent with the documentation. The internal firewall is capable of providing tight address and service filtering. Since the VPN connections are from somewhat "trusted" sources, the single level firewall filtering should suffice until the IPSEC implementation improves.

Tutorial

Enter privileged mode so that you are able to execute configuration commands.

```
spock> enable
Password: ********
spock#
```

Enter configuration mode.

```
spock# configure terminal
```

Set your passwords. The first password is to connect unprivileged, and the second is to enter privileged mode via the "enable" command.

```
spock(config)# passwd xxxxxxxx
spock(config)# enable password xxxxxxxx
```

Set the hostname and domain. This is needed to configure SSH administrative access.

```
spock(config)# hostname spock
spock(config)# domain-name giac.com
```

Name the interfaces. The security level number (0 to 100) of the interface determines its relative security level in relation to the other interfaces. A higher number indicates a more secure interface. In this example, we set the inside interface as high as possible, and the Internet connection as low as possible. If there were service nets attached to the PIX, they would be assigned a number somewhere between.

```
spock(config)# nameif ethernet0 outside security0
spock(config)# nameif ethernet1 inside security100
```

Assign IP addresses and netmasks to the interfaces.

```
spock(config)# ip address outside 99.99.1.3 255.255.255.240
spock(config)# ip address inside 10.1.1.5 255.255.255.0
```

Set the interface speed and transmission unit size.

```
spock(config)# interface ethernet0 10baset
spock(config)# interface ethernet1 10baset
spock(config)# mtu outside 1500
spock(config)# mtu inside 1500
```

Define static routes. The first one with all of the zeroes is the default route, which is outbound.

```
spock(config)# route outside 0.0.0.0 0.0.0.0 99.99.1.2 1
spock(config)# route inside 10.1.2.0 255.255.255.0 10.1.1.10 1
spock(config)# route inside 10.1.3.0 255.255.255.0 10.1.1.10 1
spock(config)# route inside 99.99.1.24 255.255.255.248 10.1.1.10 1
```

Define the outbound NAT address that will be assigned to all outbound connections, thereby hiding our internal addresses.

```
spock(config)# global (outside) 1 99.99.1.1
```

Define which internal addresses will be NATed to 99.99.1.1 when going outbound. These address ranges are the system administrators and general users workstations.

```
spock(config)# nat (inside) 1 10.1.3.64 255.255.255.248 0 0
spock(config)# nat (inside) 1 10.1.3.128 255.255.255.128 0 0
```

Disable outbound NAT from system administrator workstations to the border router. This allows the real internal address to be presented to the router so that it can decide with its ACLs if we are allowed access.

```
spock(config)# access-list acl_no_nat permit ip 10.1.3.64 255.255.255.248 host 99.99.1.2
spock(config)# nat (inside) 0 access-list acl_no_nat
```

Define allowable inbound connections and specify their NAT scheme. The first three map public 99.99.1.x to private 10.1.2.x addresses for the public service net (mail/dns/web). The next two map public 99.99.1.25 and .26 to the VPN service net without NAT. The last two disable NAT for inbound connections (router admin, employee VPN, partner VPN) to the internal infrastructure servers (syslog/ntp/etc) and applications subnets.

```
spock(config)# static (inside,outside) 99.99.1.5 10.1.2.5 netmask 255.255.255.255 0 0
spock(config)# static (inside,outside) 99.99.1.6 10.1.2.6 netmask 255.255.255.255 0 0
spock(config)# static (inside,outside) 99.99.1.7 10.1.2.7 netmask 255.255.255.255 0 0
spock(config)# static (inside,outside) 99.99.1.25 99.99.1.25 netmask 255.255.255.255 0 0
spock(config)# static (inside,outside) 99.99.1.26 99.99.1.26 netmask 255.255.255.255 0 0
spock(config)# static (inside,outside) 10.1.3.16 10.1.3.16 netmask 255.255.255.240 0 0
spock(config)# static (inside,outside) 10.1.3.96 10.1.3.96 netmask 255.255.255.224 0 0
```

Define the TFTP server for saving the configuration.

```
spock(config)# tftp-server inside 10.1.3.20 /
```

Enable protection against syn-flood attacks.

```
spock(config)# floodguard enable
```

Disable SNMP. I changed the community string from the default public to some value, just in case an administrator inadvertently enables SNMP.

```
spock(config)# no snmp-server location
spock(config)# no snmp-server contact
spock(config)# snmp-server community llpb0nes
spock(config)# no snmp-server enable traps
```

Configure the logging options and syslog server. I like to set PIX configurations to a very verbose logging level of debugging because this causes all of the browsing URLs to be included. In some very high volume installations, this is not reasonable, so you may have to adjust accordingly.

```
spock(config)# logging on
spock(config)# logging trap debugging
spock(config)# logging history debugging
```

```
spock(config)# logging host inside 10.1.3.22
```

Generate the RSA key needed for SSH access. I used a key size of 1024 bits. Use the special "ca save all" since the regular "write mem" will not save this key.

```
spock(config)# ca generate rsa key 1024
For <key_modulus_size) >= 1024, key generation could
  Take up to several minutes. Please wait.
spock(config)# ca save all
```

Configure SSH administrative access and limit the internal sources to proper subnet.

```
spock(config)# ssh 10.1.3.64 255.255.255.248 inside
spock(config)# ssh timeout 10
```

Define the RADIUS server "bouncer" for authenticating remote employee access. Notice the shared key defined for accessing "bouncer". This key should be changed periodically.

```
spock(config)# aaa-server radius-authport 1812
spock(config)# aaa-server radius-acctport 1813
spock(config)# aaa-server RADIUS protocol radius
spock(config)# aaa-server bouncer protocol radius
spock(config)# aaa-server bouncer (inside) host 10.1.3.23 6^Xa+KlN[2 timeout 10
```

Define the access control lists for connections on the outside interface.

```
spock(config)# access-list acl_out permit udp host 99.99.1.2 host 10.1.3.22 eq syslog
spock(config)# access-list acl_out permit udp host 99.99.1.2 host 10.1.3.20 eq ntp
spock(config)# access-list acl_out permit udp host 99.99.1.2 host 10.1.3.20 eq tftp
… complete list is shown in a following section …
spock(config)# access-list acl_out deny ip any any
```

Define the access control lists for connections on the inside interface.

```
spock(config)# access-list acl_in deny ip 99.99.1.0 255.255.255.248 any
spock(config)# access-list acl_in deny ip 99.99.1.8 255.255.255.252 any
spock(config)# access-list acl_in permit tcp host 10.1.2.5 any eq smtp
… complete list is shown in a following section …
spock(config)# access-list acl_in deny ip any any
```

Assign the access control lists to their respective interfaces.

```
spock(config)# access-group acl_out in interface outside
spock(config)# access-group acl_in in interface inside
```

Define a pool of addresses that will be assigned to employee's PCs when they are connecting remotely via their VPN client.

```
spock(config)# ip local pool vpn_emp_addrs 10.99.1.1-10.99.1.254
```

Enable key management protocol on the outside interface.

```
spock(config)# isakmp enable outside
```

Define the shared secret key for each of the IPSEC connection sources. Note that the employee access uses 0.0.0.0 since the employees can connect from anywhere. The partner, supplier, and customer definitions include the addresses of their security gateways. The shared keys should be changed periodically.

```
spock(config)# isakmp key 47(jFew#87?jn90% address 0.0.0.0 netmask 0.0.0.0
spock(config)# isakmp key 9$r&3890F:Tf-J6@ address 202.120.25.1 netmask 255.255.255.255
spock(config)# isakmp key 23kdf9+34^4ge3S_ address 202.100.75.1 netmask 255.255.255.255
spock(config)# isakmp key dsk38^52md-)7$#H address 207.46.230.1 netmask 255.255.255.255
```

Define the key management policy for triple-DES connections. Our employees will use this. The key lifetime is shorted from the default of 24 hours to 4 hours (14400 seconds). This is much shorter than the amount of time it would take for someone to decrypt the traffic.

```
spock(config)# isakmp policy 10 authentication pre-share
spock(config)# isakmp policy 10 encryption 3des
spock(config)# isakmp policy 10 hash md5
spock(config)# isakmp policy 10 group 2
spock(config)# isakmp policy 10 lifetime 14400
```

Define the key management policy for triple-DES connections. Our partner will use this.

```
spock(config)# isakmp policy 20 authentication pre-share
spock(config)# isakmp policy 20 encryption des
spock(config)# isakmp policy 20 hash md5
spock(config)# isakmp policy 20 group 1
spock(config)# isakmp policy 20 lifetime 14400
```

Enable IPSEC.

```
spock(config)# sysopt connection permit-ipsec
```

Define the IPSEC protocols and hash algorithms that VPN connections will use. Here we are setting up triple-DES, DES, and Authentication Header. We could possibly run into incompatibility issues with some suppliers or customers with respect to specific hash algorithms, so it is possible that we may need to define a few more in the future to accommodate these situations.

```
spock(config)# crypto ipsec transform-set strong-des esp-3des esp-md5-hmac
spock(config)# crypto ipsec transform-set regular-des esp-des esp-md5-hmac
spock(config)# crypto ipsec transform-set authhead ah-md5-hmac
```

Define the split tunnel access list for the employee VPN connections. When employees access the 10.1.2.x, 10.1.3.x, or 99.99.1.x address ranges, IPSEC will apply. Otherwise, their traffic will be routed without IPSEC to the Internet from their PC.

```
spock(config)# access-list acl_vpn_employee_split permit ip 10.1.2.0 255.255.255.0 10.99.1.0
255.255.255.0
spock(config)# access-list acl_vpn_employee_split permit ip 10.1.3.0 255.255.255.0 10.99.1.0
255.255.255.0
```

```
spock(config)# access-list acl_vpn_employee_split permit ip 99.99.1.0 255.255.255.224 10.99.1.0
255.255.255.0
```

Define a dynamic map for the employee VPN connections. The PIX requires dynamic maps since we cannot fully define the security association for the employees since we do not know where they are coming from. The PIX will dynamically create the SA during the connection negotiation. This dynamic map will be joined to regular static map. Notice the reference to use "strong-des", which we defined earlier.

```
spock(config)# crypto dynamic-map dynmap_outside 100 match address acl_vpn_employee
spock(config)# crypto dynamic-map dynmap_outside 100 set transform-set strong-des
spock(config)# crypto dynamic-map dynmap_outside 100 set security-association lifetime
seconds 14400 kilobytes 4608000
```

Define a static map for the employee VPN connections and reference the dynamic map from above. We specify a RADIUS server to ensure that the employee must provide login credentials, in case a malicious user gains physical access to the employee's computer. The map is assigned to the outside interface.

```
spock(config)# crypto map staticmap_outside 90 ipsec-isakmp dynamic dynmap_outside
spock(config)# crypto map staticmap_outside client authentication bouncer
spock(config)# crypto map staticmap_outside interface outside
```

Define a policy that is pushed to the employee VPN client. This selects the public address pool, the internal DNS server, and domain name. Split tunnel is enabled. After an idle time of 30 minutes (1800 seconds), the connection will terminate.

```
spock(config)# vpngroup vpngrp_employee address-pool vpn_emp_addrs
spock(config)# vpngroup vpngrp_employee dns-server 10.1.3.21
spock(config)# vpngroup vpngrp_employee default-domain giac.com
spock(config)# vpngroup vpngrp_employee split-tunnel acl_vpn_employee_split
spock(config)# vpngroup vpngrp_employee idle-time 1800
```

Define the access control list for the partner IPSEC connection to access our internal applications and infrastructure subnets.

```
spock(config)# access-list acl_vpn_partner1 permit ip 10.1.3.96 255.255.255.224 202.120.25.64
255.255.255.224
spock(config)# access-list acl_vpn_partner1 permit ip 10.1.3.16 255.255.255.240 202.120.25.64
255.255.255.224
```

Define the IPSEC connection for our partner. We use ISAKMP for key negotiation. The access list above is linked. The partner security gateway address is set. The regular DES encryption method is chosen and the lifetime of 4 hours is set.

```
spock(config)# crypto map staticmap_outside 10 ipsec-isakmp
spock(config)# crypto map staticmap_outside 10 match address acl_vpn_partner1
spock(config)# crypto map staticmap_outside 10 set peer 202.120.25.1
spock(config)# crypto map staticmap_outside 10 set transform-set regular-des
spock(config)# crypto map staticmap_outside 10 set security-association lifetime seconds 14400
```

```
kilobytes 4608000
```

Define the access control list for the supplier IPSEC connection to access the private web server.

```
spock(config)# access-list acl_vpn_supplier1 permit ip host 99.99.1.25 202.100.75.32
255.255.255.224
```

Define the IPSEC connection for the supplier. We use ISAKMP for key negotiation. The access list above is linked. The supplier security gateway address is set. The authentication header protocol is chosen and the lifetime of 4 hours is set.

```
spock(config)# crypto map staticmap_outside 20 ipsec-isakmp
spock(config)# crypto map staticmap_outside 20 match address acl_vpn_supplier1
spock(config)# crypto map staticmap_outside 20 set peer 202.100.75.1
spock(config)# crypto map staticmap_outside 20 set transform-set authhead
spock(config)# crypto map staticmap_outside 20 set security-association lifetime seconds 14400
kilobytes 4608000
```

Define the access control list for the customer IPSEC connection to access the private web.

```
spock(config)# access-list acl_vpn_customer1 permit ip host 99.99.1.26 207.46.230.192
255.255.255.224
```

Define the IPSEC connection for the customer. We use ISAKMP for key negotiation. The access list above is linked. The customer security gateway address is set. The authentication header protocol is chosen and the lifetime of 4 hours is set.

```
spock(config)# crypto map staticmap_outside 30 ipsec-isakmp
spock(config)# crypto map staticmap_outside 30 match address acl_vpn_customer1
spock(config)# crypto map staticmap_outside 30 set peer 207.46.230.1
spock(config)# crypto map staticmap_outside 30 set transform-set authhead
spock(config)# crypto map staticmap_outside 30 set security-association lifetime seconds 14400
kilobytes 4608000
```

Save the configuration to memory and to the TFTP server.

```
spock(config)# ca save all
spock(config)# exit
spock# write mem
Building configuration...
Cryptochecksum: xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
[OK]
spock# write net :spock.cfg
Building configuration...
TFTP write '//spock.cfg' at 10.1.3.20 on interface 1
[OK]
spock# exit
```

Border Firewall Configuration File

The completed border firewall configuration file is shown below. Please note that I added comment lines as denoted by the use of a colon at the beginning of the line.

```
: Saved
:
PIX Version 6.1(1)
:
:-------------- set interface names, passwords, and host names --------------
nameif ethernet0 outside security0
nameif ethernet1 inside security100
enable password zs87yRwjvqz.rF0Z encrypted
passwd zs87yRwjvqz.rF0Z encrypted
hostname spock
domain-name giac.com
:
:--------- higher level stack handling of some protocols is available -------
fixup protocol http 80
fixup protocol h323 1720
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol sip 5060
fixup protocol skinny 2000
fixup protocol ftp strict 21
names

:**************************************************************************
**
:--------------------------------------------------------------------------
:---------------------- outside interface rules ---------------------------
:--------------------------------------------------------------------------
:**************************************************************************
**

:---- allow local router traffic inside before ingress & rfc 1918 drops -----
: syslog to the inside
access-list acl_out permit udp host 99.99.1.2 host 10.1.3.22 eq syslog
: ntp queries to public service net
access-list acl_out permit udp host 99.99.1.2 host 10.1.3.20 eq ntp
: tftp loads to/from the inside tftp server
access-list acl_out permit udp host 99.99.1.2 host 10.1.3.20 eq tftp

:---------- drop all rfc 1918 private traffic on outside network ------------
access-list acl_out deny ip any 10.0.0.0 255.0.0.0
access-list acl_out deny ip 10.0.0.0 255.0.0.0 any
access-list acl_out deny ip any 192.168.0.0 255.255.0.0
access-list acl_out deny ip 192.168.0.0 255.255.0.0 any
: access-list acl_out deny ip any 172.16.0.0 255.255.240.0
: access-list acl_out deny ip 172.16.0.0 255.255.240.0 any

:----------- drop other unwanted or insane sources on outside network -------
access-list acl_out deny ip 127.0.0.0 255.0.0.0 any
access-list acl_out deny ip 169.254.0.0 255.255.0.0 any
```

```
access-list acl_out deny ip host 0.0.0.0 any
access-list acl_out deny ip host 255.255.255.255 any
access-list acl_out deny ip 224.0.0.0 224.0.0.0 any

:------------ drop all iana reserved sources on outside network -------------
: these arguably may not be worth the maintenance or cpu-resources.
: packets from these sources are seldom good-natured.
: iana could start assigning these, so we need to periodically check the
: reserved list at iana to validate these.
access-list acl_out deny ip 1.0.0.0 255.0.0.0 any
access-list acl_out deny ip 2.0.0.0 255.0.0.0 any
access-list acl_out deny ip 23.0.0.0 255.0.0.0 any
access-list acl_out deny ip 31.0.0.0 255.0.0.0 any
access-list acl_out deny ip 67.0.0.0 255.0.0.0 any
access-list acl_out deny ip 68.0.0.0 255.0.0.0 any
access-list acl_out deny ip 69.0.0.0 255.0.0.0 any
access-list acl_out deny ip 70.0.0.0 255.0.0.0 any
access-list acl_out deny ip 71.0.0.0 255.0.0.0 any
access-list acl_out deny ip 72.0.0.0 255.0.0.0 any
access-list acl_out deny ip 73.0.0.0 255.0.0.0 any
access-list acl_out deny ip 74.0.0.0 255.0.0.0 any
access-list acl_out deny ip 75.0.0.0 255.0.0.0 any
access-list acl_out deny ip 76.0.0.0 255.0.0.0 any
access-list acl_out deny ip 77.0.0.0 255.0.0.0 any
access-list acl_out deny ip 78.0.0.0 255.0.0.0 any
access-list acl_out deny ip 79.0.0.0 255.0.0.0 any
access-list acl_out deny ip 80.0.0.0 255.0.0.0 any
access-list acl_out deny ip 81.0.0.0 255.0.0.0 any
access-list acl_out deny ip 82.0.0.0 255.0.0.0 any
access-list acl_out deny ip 83.0.0.0 255.0.0.0 any
access-list acl_out deny ip 84.0.0.0 255.0.0.0 any
access-list acl_out deny ip 85.0.0.0 255.0.0.0 any
access-list acl_out deny ip 86.0.0.0 255.0.0.0 any
access-list acl_out deny ip 87.0.0.0 255.0.0.0 any
access-list acl_out deny ip 88.0.0.0 255.0.0.0 any
access-list acl_out deny ip 89.0.0.0 255.0.0.0 any
access-list acl_out deny ip 90.0.0.0 255.0.0.0 any
access-list acl_out deny ip 91.0.0.0 255.0.0.0 any
access-list acl_out deny ip 92.0.0.0 255.0.0.0 any
access-list acl_out deny ip 93.0.0.0 255.0.0.0 any
access-list acl_out deny ip 94.0.0.0 255.0.0.0 any
access-list acl_out deny ip 95.0.0.0 255.0.0.0 any
access-list acl_out deny ip 96.0.0.0 255.0.0.0 any
access-list acl_out deny ip 97.0.0.0 255.0.0.0 any
access-list acl_out deny ip 98.0.0.0 255.0.0.0 any
access-list acl_out deny ip 99.0.0.0 255.0.0.0 any
access-list acl_out deny ip 100.0.0.0 255.0.0.0 any
access-list acl_out deny ip 101.0.0.0 255.0.0.0 any
access-list acl_out deny ip 102.0.0.0 255.0.0.0 any
access-list acl_out deny ip 103.0.0.0 255.0.0.0 any
access-list acl_out deny ip 104.0.0.0 255.0.0.0 any
access-list acl_out deny ip 105.0.0.0 255.0.0.0 any
access-list acl_out deny ip 106.0.0.0 255.0.0.0 any
access-list acl_out deny ip 107.0.0.0 255.0.0.0 any
access-list acl_out deny ip 108.0.0.0 255.0.0.0 any
access-list acl_out deny ip 109.0.0.0 255.0.0.0 any
access-list acl_out deny ip 110.0.0.0 255.0.0.0 any
access-list acl_out deny ip 111.0.0.0 255.0.0.0 any
```

```
access-list acl_out deny ip 112.0.0.0 255.0.0.0 any
access-list acl_out deny ip 113.0.0.0 255.0.0.0 any
access-list acl_out deny ip 114.0.0.0 255.0.0.0 any
access-list acl_out deny ip 115.0.0.0 255.0.0.0 any
access-list acl_out deny ip 126.0.0.0 255.0.0.0 any
access-list acl_out deny ip 191.255.0.0 255.255.0.0 any
access-list acl_out deny ip 192.0.2.0 255.255.255.0 any
access-list acl_out deny ip 201.0.0.0 255.255.192.0 any
access-list acl_out deny ip 223.255.255.0 255.255.255.0 any

:--- drop all inbound traffic on outside interface with our public addrs ----
: ingress filtering. only block x.0 thru x.11, and x.16 thru x.31
: the public x.12 thru x.15 are assigned outside to vpn clients
access-list acl_out deny ip 99.99.1.0 255.255.255.224 any

:--- allow outside world to public service net via permissible protocols ----
: public addresses: mail, dns, http, and https
access-list acl_out permit tcp any host 99.99.1.5 eq smtp
access-list acl_out permit udp any host 99.99.1.6 eq domain
access-list acl_out permit tcp any host 99.99.1.7 eq www
access-list acl_out permit tcp any host 99.99.1.7 eq 443

:----------- catch-all rules to drop the rest of the garbage ----------------
access-list acl_out deny ip any any


:*************************************************************************
**
:-------------------------------------------------------------------------
:-------------------- inside interface rules -----------------------------
:-------------------------------------------------------------------------
:*************************************************************************
**

:----- drop all outbound traffic on inside interface with outside source ----
: egress filtering. skip the vpn service net since it uses public addrs.
access-list acl_in deny ip 99.99.1.0 255.255.255.240 any
access-list acl_in deny ip 99.99.1.16 255.255.255.248 any

:---- allow public service net to outside world via permissible protocols ---
: private addresses: mail, dns, and time
access-list acl_in permit tcp host 10.1.2.5 any eq smtp
access-list acl_in permit udp host 10.1.2.6 any eq domain
access-list acl_in permit udp host 10.1.2.5 host 128.118.25.3 eq ntp
access-list acl_in permit udp host 10.1.2.5 host 132.236.56.250 eq ntp
access-list acl_in permit udp host 10.1.2.5 host 198.82.162.213 eq ntp

:------- allow inside users to outside world via permissible protocols ------
: general user desktops: http, https, and ftp
access-list acl_in permit tcp 10.1.3.128 255.255.255.128 any eq www
access-list acl_in permit tcp 10.1.3.128 255.255.255.128 any eq 443
access-list acl_in permit tcp 10.1.3.128 255.255.255.128 any eq ftp
: admin user desktops: http, https, and ftp
access-list acl_in permit tcp 10.1.3.64 255.255.255.248 any eq www
access-list acl_in permit tcp 10.1.3.64 255.255.255.248 any eq 443
access-list acl_in permit tcp 10.1.3.64 255.255.255.248 any eq ftp

:----------------- allow administrative router protocols --------------------
```

```
access-list acl_in permit tcp 10.1.3.64 255.255.255.248 host 99.99.1.2 eq
22

:-------------- allow VPN radius authentication & accounting ----------------
access-list acl_in permit tcp host 10.1.1.5 host 10.3.2.23 range 1812 1813
access-list acl_in permit udp host 10.1.1.5 host 10.3.2.23 range 1812 1813

:------------ catch-all rules to drop the rest of the garbage ---------------
access-list acl_in deny ip any any

:**************************************************************************
**
:--------------------------------------------------------------------------
:------------ disable nat for router admin traffic and vpn traffic ----------
:--------------------------------------------------------------------------
:**************************************************************************
**
access-list acl_no_nat permit ip 10.1.3.64 255.255.255.248 host 99.99.1.2
access-list acl_no_nat permit ip any 10.99.1.0 255.255.255.0
access-list acl_no_nat permit ip any 202.120.25.64 255.255.255.224
access-list acl_no_nat permit ip any 202.100.75.32 255.255.255.224
access-list acl_no_nat permit ip any 207.46.230.192 255.255.255.224

:**************************************************************************
**
:--------------------------------------------------------------------------
:--------------------------- vpn access rules -----------------------------
:--------------------------------------------------------------------------
:**************************************************************************
**

:-------- specify return traffic to be protected to partner via ipsec -------
access-list acl_vpn_partner1 permit ip 10.1.3.96 255.255.255.224
202.120.25.64 255.255.255.224
access-list acl_vpn_partner1 permit ip 10.1.3.16 255.255.255.240
202.120.25.64 255.255.255.224

:------- specify return traffic to be protected to suppliers via ipsec ------
access-list acl_vpn_supplier1 permit ip host 99.99.1.25 202.100.75.32
255.255.255.224

:------- specify return traffic to be protected to customers via ipsec ------
access-list acl_vpn_customer1 permit ip host 99.99.1.26 207.46.230.192
255.255.255.224

:---- specify inbound employee traffic that is protected by split tunnel ----
access-list acl_vpn_employee_split permit ip 10.1.2.0 255.255.255.0
10.99.1.0 255.255.255.0
access-list acl_vpn_employee_split permit ip 10.1.3.0 255.255.255.0
10.99.1.0 255.255.255.0
access-list acl_vpn_employee_split permit ip 99.99.1.0 255.255.255.224
10.99.1.0 255.255.255.0

:------ specify return traffic to be protected to employees via ipsec -------
access-list acl_vpn_employee permit ip host 99.99.1.3 10.99.1.0
255.255.255.0
access-list acl_vpn_employee permit ip 10.1.2.0 255.255.255.0 10.99.1.0
255.255.255.0
```

```
access-list acl_vpn_employee permit ip 10.1.3.0 255.255.255.0 10.99.1.0
255.255.255.0

:------------------------ set logging options -------------------------------
pager lines 22
logging on
logging trap debugging
logging history debugging
logging host inside 10.1.3.22
:
:--------------------- configure the interface cards -----------------------
interface ethernet0 10baset
interface ethernet1 10baset
mtu outside 1500
mtu inside 1500
ip address outside 99.99.1.3 255.255.255.240
ip address inside 10.1.1.5 255.255.255.0
ip audit info action drop
ip audit attack action drop
:
:--- define pool of public addresses to assign to employee vpn connections --
ip local pool vpn_emp_addrs 10.99.1.1-10.99.1.254
pdm history enable
arp timeout 14400
:
:------------------- nat a public addr to all outbound connections -----------
global (outside) 1 99.99.1.1
:
:--------- define when outbound nat is applied to a connection --------------
: skip nat for sysadmin to router
nat (inside) 0 access-list acl_no_nat
: do nat for outside world connections for sysadmin and general user
subnets
nat (inside) 1 10.1.3.64 255.255.255.248 0 0
nat (inside) 1 10.1.3.128 255.255.255.128 0 0
:
:------ nat public addrs to public service net for inbound connections ------
static (inside,outside) 99.99.1.5 10.1.2.5 netmask 255.255.255.255 0 0
static (inside,outside) 99.99.1.6 10.1.2.6 netmask 255.255.255.255 0 0
static (inside,outside) 99.99.1.7 10.1.2.7 netmask 255.255.255.255 0 0
:----------- permit non-nat connections to private web servers --------------
static (inside,outside) 99.99.1.25 99.99.1.25 netmask 255.255.255.255 0 0
static (inside,outside) 99.99.1.26 99.99.1.26 netmask 255.255.255.255 0 0
:
:----------- permit non-nat connections to internal network -----------------
static (inside,outside) 10.1.3.16 10.1.3.16 netmask 255.255.255.240 0 0
static (inside,outside) 10.1.3.96 10.1.3.96 netmask 255.255.255.224 0 0
:
:------------ bind the acls as a group to the proper interface --------------
access-group acl_out in interface outside
access-group acl_in in interface inside
:
:---------------------- define static routing ------------------------------
route outside 0.0.0.0 0.0.0.0 99.99.1.2 1
route inside 10.1.2.0 255.255.255.0 10.1.1.10 1
route inside 10.1.3.0 255.255.255.0 10.1.1.10 1
route inside 99.99.1.24 255.255.255.248 10.1.1.10 1
:
```

```
:------------------------- set timeout values -------------------------------
timeout xlate 3:00:00
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h323
0:05:00 sip 0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
:
:------------ define radius authentication for employee remote access -------
aaa-server radius-authport 1812
aaa-server radius-acctport 1813
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
aaa-server bouncer protocol radius
aaa-server bouncer (inside) host 10.1.3.23 6^Xa+KlN[2 timeout 10
:
:------------------------------ disable snmp --------------------------------
no snmp-server location
no snmp-server contact
snmp-server community llpb0nes
no snmp-server enable traps
:
:---------------- define tftp server for saving config file -----------------
tftp-server inside 10.1.3.20 /
:
:---------------------- enable syn-flood protection ------------------------
floodguard enable
:
:
:**************************************************************************
**
:--------------------------------------------------------------------------
:------------------- ipsec security association definitions  ---------------
:--------------------------------------------------------------------------
:**************************************************************************
**
: enable ipsec
sysopt connection permit-ipsec
no sysopt route dnat
:
: define permissible protocols and hash algorithms
crypto ipsec transform-set strong-des esp-3des esp-md5-hmac
crypto ipsec transform-set regular-des esp-des esp-md5-hmac
crypto ipsec transform-set authhead ah-md5-hmac
:
: define dynamic security association for employee ipsec access
crypto dynamic-map dynmap_outside 100 match address acl_vpn_employee
crypto dynamic-map dynmap_outside 100 set transform-set strong-des
crypto dynamic-map dynmap_outside 100 set security-association lifetime
seconds 14400 kilobytes 4608000
:
: define security association for partner ipsec access
crypto map staticmap_outside 10 ipsec-isakmp
crypto map staticmap_outside 10 match address acl_vpn_partner1
crypto map staticmap_outside 10 set peer 202.120.25.1
crypto map staticmap_outside 10 set transform-set regular-des
crypto map staticmap_outside 10 set security-association lifetime seconds
14400 kilobytes 4608000
:
: define security association for supplier ipsec access
```

```
crypto map staticmap_outside 20 ipsec-isakmp
crypto map staticmap_outside 20 match address acl_vpn_supplier1
crypto map staticmap_outside 20 set peer 202.100.75.1
crypto map staticmap_outside 20 set transform-set authhead
crypto map staticmap_outside 20 set security-association lifetime seconds
14400 kilobytes 4608000
:
: define security association for customer ipsec access
crypto map staticmap_outside 30 ipsec-isakmp
crypto map staticmap_outside 30 match address acl_vpn_customer1
crypto map staticmap_outside 30 set peer 207.46.230.1
crypto map staticmap_outside 30 set transform-set authhead
crypto map staticmap_outside 30 set security-association lifetime seconds
14400 kilobytes 4608000
:
: define security association for employee ipsec access
crypto map staticmap_outside 90 ipsec-isakmp dynamic dynmap_outside
crypto map staticmap_outside client authentication bouncer
crypto map staticmap_outside interface outside
:
: enable key management for ipsec connections
isakmp enable outside
: define shared secret keys for emps, partners, suppliers, custs
isakmp key 47(jFew#87?jn90% address 0.0.0.0 netmask 0.0.0.0
isakmp key 9$r&3890F:Tf-J6@ address 202.120.25.1 netmask 255.255.255.255
isakmp key 23kdf9+34^4ge3S_ address 202.100.75.1 netmask 255.255.255.255
isakmp key dsk38^52md-)7$#H address 207.46.230.1 netmask 255.255.255.255
:
: define valid encryption combinations
isakmp policy 10 authentication pre-share
isakmp policy 10 encryption 3des
isakmp policy 10 hash md5
isakmp policy 10 group 2
isakmp policy 10 lifetime 14400
isakmp policy 20 authentication pre-share
isakmp policy 20 encryption des
isakmp policy 20 hash md5
isakmp policy 20 group 1
isakmp policy 20 lifetime 14400
:
: define ipsec connection parameters to push to employee vpn clients
vpngroup vpngrp_employee address-pool vpn_emp_addrs
vpngroup vpngrp_employee dns-server 10.1.3.21
vpngroup vpngrp_employee default-domain giac.com
vpngroup vpngrp_employee split-tunnel acl_vpn_employee_split
vpngroup vpngrp_employee idle-time 1800
:
: ------------------ define system admin connections via ssh ----------------
telnet timeout 5
ssh 10.1.3.64 255.255.255.248 inside
ssh timeout 10
terminal width 80
:
Cryptochecksum:xxxxxxxxxxxxxxxxxxxx
: end
```

Internal Firewall

Comments

The border firewall uses addresses of 99.99.1.x representing the public IP addresses assigned to GIAC Enterprises. These are not legal public address on the Internet. They were used in my home lab for testing purposes. Since I used a private 172.16.1.x address on the outside router interface and connected testing equipment in this outside private range, the ACL's that would normally block this address space were commented (#) from my test configuration, but would be included in a production firewall configuration that uses legal public addresses.

The firewall configuration below uses public addresses of 202.120.25.x, 202.100.75.x, and 207.46.230.x representing addresses for the partner, supplier, and customer. These are addresses that I chose as examples simply because the 202.x ranges were listed at APNIC as China registrations, and the 207.x was listed at ARIN as a U.S registration (Microsoft). They were used in my home lab for testing purposes and have no significance otherwise.

Internal Firewall Policy Summary

Action	Proto	Source	Destination	Service	Comments
		General Defenses			
Drop	TCP	Any	Any	Any	Out of Spec
Drop	Any	RFC 1918 private addresses	Any	Any	
Drop	Any	Experimental, reserved, loopback	Any	Any	
Drop	Any	Multicast and broadcast	Any	Any	
		Local Firewall to Inside Net			
Permit	UDP	10.1.3.10	10.1.3.22	SYSLOG	
Permit	UDP	10.1.3.10	10.1.3.21	DNS	
Permit	TCP	10.1.3.64/255.255.255.248	10.1.3.10	SSH	Sysadmin
Permit	UDP	10.1.3.10	10.1.3.20	TIME	
		Border Devices to/from Inside & Public Service Nets			
Permit	UDP	10.1.1.5	10.1.3.22	SYSLOG	
Permit	UDP	99.99.1.2	10.1.3.22	SYSLOG	
Permit	TCP	10.1.3.64/255.255.255.248	99.99.1.2	SSH	Sysadmin
Permit	TCP	10.1.3.64/255.255.255.248	10.1.1.5	SSH	Sysadmin
Permit	UDP	99.99.1.2	10.1.3.20	TIME	
Permit	UDP	10.1.1.5	10.1.3.23	RADIUS	Emp auth
Permit	UDP	10.1.1.5	10.1.3.20	TFTP	PIX config

Permit	UDP	99.99.1.2	10.1.3.20	TFTP	Router cfg
		Outside to Public Svc Net			
Permit	TCP	Any	10.1.2.5	SMTP	Mail inbound
Permit	TCP	Any	10.1.2.6	DNS	External DNS
Permit	TCP	Any	10.1.2.7	HTTP	Public web
Permit	TCP	Any	10.1.2.7	HTTPS	Public web
		Remote VPN Employee to Inside			
Permit	TCP	10.99.1.0/255.255.255.0	All 10.1.3.0	SSH	Server logins
Permit	UDP	10.99.1.0/255.255.255.0	10.1.3.21	DNS	Internal DNS
Permit	TCP	10.99.1.0/255.255.255.0	10.1.3.24	POP3	Internal mail
Permit	TCP	10.99.1.0/255.255.255.0	10.1.3.96/255.255.255.224	HTTP	Internal apps
Permit	TCP	10.99.1.0/255.255.255.0	10.1.3.96/255.255.255.224	HTTPS	Internal apps
Permit	TCP	10.99.1.0/255.255.255.0	10.1.3.96/255.255.255.224	Citrix	Internal apps
		Partner to Inside & Public Service Nets			
Permit	TCP	202.120.25.64/255.255.255.224	10.1.3.21	DNS	Internal DNS
Permit	TCP	202.120.25.64/255.255.255.224	10.1.3.96/255.255.255.224	HTTP	Internal apps
Permit	TCP	202.120.25.64/255.255.255.224	10.1.3.96/255.255.255.224	HTTPS	Internal apps
Permit	TCP	202.120.25.64/255.255.255.224	10.1.3.96/255.255.255.224	Citrix	Internal apps
		Supplier to VPN Svc Net			
Permit	TCP	202.100.75.32/255.255.255.224	99.99.1.25	HTTP	Supplier web
Permit	TCP	202.100.75.32/255.255.255.224	99.99.1.25	HTTPS	Supplier web
Permit	TCP	202.100.75.32/255.255.255.224	99.99.1.25	SFTP	Supplier web
Permit	TCP	202.100.75.32/255.255.255.224	99.99.1.25	FTP	Supplier web
		Customer to VPN Svc Net			
Permit	TCP	207.46.230.192/255.255.255.224	99.99.1.26	HTTP	Supplier web
Permit	TCP	207.46.230.192/255.255.255.224	99.99.1.26	HTTPS	Supplier web
Permit	TCP	207.46.230.192/255.255.255.224	99.99.1.26	SFTP	Supplier web
Permit	TCP	207.46.230.192/255.255.255.224	99.99.1.26	FTP	Supplier web
		VPN Service Net to Inside			
Permit	UDP	99.99.1.24/255.255.255.248	10.1.3.22	SYSLOG	
Permit	UDP	99.99.1.24/255.255.255.248	10.1.3.20	TIME	
		Public Service Net to Outside			
Permit	UDP	10.1.2.5	128.118.25.3	TIME	NTP provider
Permit	UDP	10.1.2.5	132.236.56.250	TIME	NTP provider
Permit	UDP	10.1.2.5	198.82.162.213	TIME	NTP provider
Permit	UDP	10.1.2.6	Any	DNS	Forwarding
Permit	TCP	10.1.2.5	Any	SMTP	Mail out
		Public Service Net to Inside			

Permit	UDP	All 10.1.2.0	10.1.3.22	SYSLOG	
Permit	TCP	10.1.2.7	10.1.3.24	SMTP	Mail inbound
		Inside to Public Service Net			
Permit	TCP	10.1.3.64/255.255.255.248	All 10.1.2.0	SSH	Sysadmin
Permit	TCP	10.1.3.64/255.255.255.248	10.1.2.7	HTTP	Public web
Permit	TCP	10.1.3.128/255.255.255.128	10.1.2.7	HTTP	Public web
Permit	TCP	10.1.3.64/255.255.255.248	10.1.2.7	HTTPS	Public web
Permit	TCP	10.1.3.128/255.255.255.128	10.1.2.7	HTTPS	Public web
Permit	TCP	10.1.3.24	10.1.2.5	SMTP	Mail out
Permit	UDP	10.1.3.20	10.1.2.5	TIME	NTP forward
Permit	UDP	10.1.3.21	10.1.2.6	DNS	Forwarding
		Inside to VPN Service Net			
Permit	TCP	10.1.3.64/255.255.255.248	99.99.1.24/255.255.255.248	SSH	Sysadmin
Permit	TCP	10.1.3.64/255.255.255.248	99.99.1.24/255.255.255.248	HTTP	VPN webs
Permit	TCP	10.1.3.64/255.255.255.248	99.99.1.24/255.255.255.248	HTTPS	VPN webs
Permit	TCP	10.1.3.64/255.255.255.248	99.99.1.24/255.255.255.248	SFTP	VPN webs
Permit	TCP	10.1.3.64/255.255.255.248	99.99.1.24/255.255.255.248	FTP	VPN webs
Permit	TCP	10.1.3.128/255.255.255.128	99.99.1.24/255.255.255.248	SSH	User logins
Permit	TCP	10.1.3.128/255.255.255.128	99.99.1.24/255.255.255.248	HTTP	VPN webs
Permit	TCP	10.1.3.128/255.255.255.128	99.99.1.24/255.255.255.248	HTTPS	VPN webs
Permit	TCP	10.1.3.128/255.255.255.128	99.99.1.24/255.255.255.248	SFTP	VPN webs
Permit	TCP	10.1.3.128/255.255.255.128	99.99.1.24/255.255.255.248	FTP	VPN webs
		Inside to Outside World			
Permit	TCP	10.1.3.128/255.255.255.128	Any	HTTP	Browsing
Permit	TCP	10.1.3.128/255.255.255.128	Any	HTTPS	Browsing
Permit	TCP	10.1.3.128/255.255.255.128	Any	FTP	Downloads
Permit	TCP	10.1.3.64/255.255.255.248	Any	HTTP	Browsing
Permit	TCP	10.1.3.64/255.255.255.248	Any	HTTPS	Browsing
Permit	TCP	10.1.3.64/255.255.255.248	Any	FTP	Downloads
		Anything Else			
Deny	Any	Any	Any	Any	Log it too

IPTables Overview

Unlike the Cisco border products that require an extensive IOS command set, the Linux IPTables firewall commands are fairly brief. There are a lot of resources on the Internet and in bookstores that describe how to write rules that permit or deny particular kinds of traffic. Robert Ziegler published a particularly good SANS course book that shows how to construct rules, including many examples of common traffic flows.

The firewall rule set can be put into a bash shell script that is run at system startup from the initialization directories. I placed my rule set into /usr/local/bin/fw_init.sh and I execute it from the /etc/rc.local startup file.

The rules can be classified into three basic types: INPUT, OUTPUT, and FORWARD. The input rules control traffic that originates outside the firewall and is destined for a process running on the firewall itself. The output rules control traffic that originates from a process on the firewall and is destined outside the firewall. The forward rules control traffic that originate outside the firewall and are to be passed through the firewall to another interface to an outside destination.

Each rule allows you to specify the protocol such as UDP or TCP, the source address and source ports, and the destination address and destination ports. Specific network interfaces can also be specified so that you can control on which interface traffic arrives or leaves. Since IPTables is stateful, connections are added to an internal table that tracks connections as they are built (syn/syn-ack/ack), as they are used, and as they are torn down (fin-ack). Rules can be constructed to allow or deny traffic based upon their state, as well as based upon the specific state flags. For example, you can accept traffic that is already in an established state without having to code rules to examine the flag settings or the port reversals. You can also reject crafted packets that have TCP flag combinations that should not naturally occur.

The following shows an example of rules that allow traffic flow for a browser to communicate with a web server. The browser is anywhere on the Internet, and the web server is at 192.168.1.1. The outside interface is “eth0” and the web server is located on a service network on interface “eth1”.

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A FORWARD -i eth0 -o $eth1 -p tcp -s any/0 --sport 1024:65535 -d 192.168.1.1 --dport 80 -j ACCEPT
```

The first rule allows established connections. After the TCP 3-way handshake completes, this rule will allow the remaining traffic flows to pass through. The second rule allows any source address via ports greater than 1023 to connect to port 80 on the web server. This rule will be used during the connection establishment.

IPTables permits the logging of packets. Unfortunately, each packet that you want logged will need its own rule. For example, if you wanted to log the matched HTTP rule in the above

example, you would need a separate rule since the log option cannot be appended onto the forward rule.

```
iptables -A FORWARD -i eth0 -o $eth1 -p tcp -s any/0 --sport 1024:65535 -d 192.168.1.1 --dport 80 -j LOG
```

I found that the default logging level logs not only to the log files in /var/log, but it also sends the log entries to the console. This is not only annoying, but if you are under attack, the console could become useless. I elected to lower the logging level to "info (6)" which disables the console screen messages. An example of a "catch-all" log command for "input" packets follows:

```
iptables -A INPUT -j LOG --log-level info
```

System Hardening

Since the firewall product is running on a general-purpose Linux operating system, it is hardened, and only the bare minimum packages are installed. The only network services running are NTP for clock synchronization from inside, and SSH for administrative access from inside.

In addition to standard hardening processes, the general network behavior must be restricted. This includes disabling source routing, disabling ICMP broadcasts, enabling syn-flood protection, and spoofed source protection. The Linux 2.4 kernel has several of these abilities built-in, and the IPTables configuration script needs to ensure that they are enabled.

Internal Firewall Configuration File

The complete internal firewall configuration file is shown below. Please note that I added comment lines as denoted by the use of a pound-sign (#) at the beginning of the line.

```
#!/bin/bash
# fw_init.sh
# ---------------------- modification history ------------------------------
#
# cxt 06-23-2001 Initial Setup
#
#
# -------------------------- comments ----------------------------------------
#
# Baseline techniques from book "Advanced Linux Firewall Configuration"
# by Robert Ziegler, SANS 2001 in Baltimore MD.
#

# ------------------------------------------------------------------------
# -------------------- kernel monitoring support -------------------------
# ------------------------------------------------------------------------

echo "$0: initializing traffic rules"

DIR="/proc/sys/net/ipv4"
cd $DIR
if (( $? != 0 )); then
```

```
   echo "$0: abort: unable to cd to $DIR"
   exit 1
fi

# enable ip forwarding
echo 1 >ip_forward
for FILE in conf/*/proxy_arp; do
    echo 1 >$FILE
done

# enable broadcast echo protection
echo 1 >icmp_echo_ignore_broadcasts

# enable all echo protection
echo 1 >icmp_echo_ignore_all

# disable source routing
for FILE in conf/*/accept_source_route; do
    echo 0 >$FILE
done

# enable tcp syn cookie protection
echo 1 >tcp_syncookies

# disable icmp redirects
for FILE in  conf/*/accept_redirects  conf/*/send_redirects; do
    echo 0 >$FILE
done

# drop spoofed packets
for FILE in  conf/*/rp_filter; do
    echo 1 >$FILE
done

# log impossible martian packets
for FILE in  conf/*/log_martians; do
    echo 1 >$FILE
done

# ------------------------------------------------------------------------
# ------------------------ load kernel modules ----------------------------
# ------------------------------------------------------------------------

modprobe ip_tables
modprobe ip_conntrack
modprobe ip_conntrack_ftp
modprobe ipt_LOG
modprobe ipt_REJECT
modprobe ipt_state
modprobe ipt_multiport
modprobe ipt_limit

# ------------------------------------------------------------------------
# -------------------- setup default firewall policies -------------------
# ------------------------------------------------------------------------

# variables for my local addresses and interfaces
IF_OUTSIDE="eth1"
```

```
IF_SERV_PUBLIC="eth2"
IF_SERV_VPN="eth0"
IF_INSIDE="eth3"
ADDR_OUTSIDE="10.1.1.10"
ADDR_SERV_PUBLIC="10.1.2.2"
ADDR_SERV_VPN="99.99.1.30"
ADDR_INSIDE="10.1.3.10"

# whack em all
iptables --flush

# allow loopback traffic
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# setup default drop-all rules
iptables --policy INPUT   DROP
iptables --policy OUTPUT  DROP
iptables --policy FORWARD DROP

# -------------------- block out-of-spec packets ----------------------------

iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A INPUT -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
iptables -A INPUT -p tcp --tcp-flags ACK,FIN FIN -j DROP
iptables -A INPUT -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -A INPUT -p tcp --tcp-flags ACK,PSH PSH -j DROP
# block fragmented icmp
iptables -A INPUT --fragment -p icmp -j DROP

iptables -A FORWARD -p tcp --tcp-flags ALL NONE -j DROP
iptables -A FORWARD -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A FORWARD -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
iptables -A FORWARD -p tcp --tcp-flags ACK,FIN FIN -j DROP
iptables -A FORWARD -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -A FORWARD -p tcp --tcp-flags ACK,PSH PSH -j DROP
# block fragmented icmp
iptables -A FORWARD --fragment -p icmp -j DROP

# -------- allow established connections already in the state table ---------

iptables -A INPUT   -m state --state INVALID -j DROP
iptables -A OUTPUT  -m state --state INVALID -j DROP
iptables -A FORWARD -m state --state INVALID -j DROP
iptables -A INPUT   -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# --------------- block spoofing and impossible addresses -------------------

# block land attack
iptables -A INPUT -i $IF_OUTSIDE  -s $ADDR_OUTSIDE  -j DROP
iptables -A INPUT -i $IF_SERV_PUBLIC -s $ADDR_SERV_PUBLIC -j DROP
iptables -A INPUT -i $IF_SERV_VPN -s $ADDR_SERV_VPN -j DROP
iptables -A INPUT -i $IF_INSIDE   -s $ADDR_INSIDE   -j DROP
```

```
# block standard rfc 1918 private addresses
# be careful to not block privates that we are using !
iptables -A INPUT   -i $IF_OUTSIDE     -s 10.0.0.0/8     -j DROP
iptables -A INPUT   -i $IF_OUTSIDE     -s 172.16.0.0/12  -j DROP
iptables -A INPUT   -i $IF_OUTSIDE     -s 192.168.0.0/16 -j DROP
iptables -A INPUT   -i $IF_SERV_PUBLIC -s 10.0.0.0/8     -j DROP
iptables -A INPUT   -i $IF_SERV_PUBLIC -s 172.16.0.0/12  -j DROP
iptables -A INPUT   -i $IF_SERV_PUBLIC -s 192.168.0.0/16 -j DROP
iptables -A INPUT   -i $IF_SERV_VPN    -s 10.0.0.0/8     -j DROP
iptables -A INPUT   -i $IF_SERV_VPN    -s 172.16.0.0/12  -j DROP
iptables -A INPUT   -i $IF_SERV_VPN    -s 192.168.0.0/16 -j DROP
# iptables -A INPUT   -i $IF_INSIDE      -s 10.0.0.0/8     -j DROP
iptables -A INPUT   -i $IF_INSIDE      -s 172.16.0.0/12  -j DROP
iptables -A INPUT   -i $IF_INSIDE      -s 192.168.0.0/16 -j DROP
# iptables -A FORWARD -i $IF_OUTSIDE     -s 10.0.0.0/8     -j DROP
# iptables -A FORWARD -i $IF_OUTSIDE     -s 172.16.0.0/12  -j DROP
iptables -A FORWARD -i $IF_OUTSIDE     -s 192.168.0.0/16 -j DROP
# iptables -A FORWARD -i $IF_SERV_PUBLIC -s 10.0.0.0/8     -j DROP
iptables -A FORWARD -i $IF_SERV_PUBLIC -s 172.16.0.0/12  -j DROP
iptables -A FORWARD -i $IF_SERV_PUBLIC -s 192.168.0.0/16 -j DROP
iptables -A FORWARD -i $IF_SERV_VPN    -s 10.0.0.0/8     -j DROP
iptables -A FORWARD -i $IF_SERV_VPN    -s 172.16.0.0/12  -j DROP
iptables -A FORWARD -i $IF_SERV_VPN    -s 192.168.0.0/16 -j DROP
# iptables -A FORWARD -i $IF_INSIDE      -s 10.0.0.0/8     -j DROP
iptables -A FORWARD -i $IF_INSIDE      -s 172.16.0.0/12  -j DROP
iptables -A FORWARD -i $IF_INSIDE      -s 192.168.0.0/16 -j DROP

# block experimental and reserved addresses
iptables -A INPUT   -s 240.0.0.0/4   -j DROP
iptables -A INPUT   -s 169.254.0.0/16 -j DROP
iptables -A INPUT   -s 192.0.2.0/24   -j DROP
iptables -A FORWARD -s 240.0.0.0/4    -j DROP
iptables -A FORWARD -s 169.254.0.0/16 -j DROP
iptables -A FORWARD -s 192.0.2.0/24   -j DROP

# block traffic on physical interfaces claiming to be from loopback
iptables -A INPUT   -i $IF_OUTSIDE     -s 127.0.0/8 -j DROP
iptables -A INPUT   -i $IF_SERV_PUBLIC -s 127.0.0/8 -j DROP
iptables -A INPUT   -i $IF_SERV_VPN    -s 127.0.0/8 -j DROP
iptables -A INPUT   -i $IF_INSIDE      -s 127.0.0/8 -j DROP
iptables -A FORWARD -i $IF_OUTSIDE     -s 127.0.0/8 -j DROP
iptables -A FORWARD -i $IF_SERV_PUBLIC -s 127.0.0/8 -j DROP
iptables -A FORWARD -i $IF_SERV_VPN    -s 127.0.0/8 -j DROP
iptables -A FORWARD -i $IF_INSIDE      -s 127.0.0/8 -j DROP

# block multicast source, and non-udp multicast destinations
iptables -A INPUT   -s 224.0.0.0/4 -j DROP
iptables -A INPUT   -d 224.0.0.0/4 -p ! udp -j DROP
iptables -A FORWARD -s 224.0.0.0/4 -j DROP
iptables -A FORWARD -d 224.0.0.0/4 -p ! udp -j DROP

# block broadcast garbage
iptables -A INPUT   -s 255.0.0.0/8
iptables -A INPUT   -d 0.0.0.0/8
iptables -A FORWARD -s 255.0.0.0/8
iptables -A FORWARD -d 0.0.0.0/8
```

```
# block directed broadcasts
iptables -A INPUT   -i $IF_OUTSIDE     -d 10.1.1.0    -j DROP
iptables -A INPUT   -i $IF_OUTSIDE     -d 10.1.1.255  -j DROP
iptables -A INPUT   -i $IF_SERV_PUBLIC -d 10.1.2.0    -j DROP
iptables -A INPUT   -i $IF_SERV_PUBLIC -d 10.1.2.255  -j DROP
iptables -A INPUT   -i $IF_SERV_VPN    -d 99.99.1.24  -j DROP
iptables -A INPUT   -i $IF_SERV_VPN    -d 99.99.1.31  -j DROP
iptables -A INPUT   -i $IF_INSIDE      -d 10.1.3.0    -j DROP
iptables -A INPUT   -i $IF_INSIDE      -d 10.1.3.255  -j DROP
iptables -A FORWARD -i $IF_OUTSIDE     -d 10.1.1.0    -j DROP
iptables -A FORWARD -i $IF_OUTSIDE     -d 10.1.1.255  -j DROP
iptables -A FORWARD -i $IF_SERV_PUBLIC -d 10.1.2.0    -j DROP
iptables -A FORWARD -i $IF_SERV_PUBLIC -d 10.1.2.255  -j DROP
iptables -A FORWARD -i $IF_SERV_VPN    -d 99.99.1.24  -j DROP
iptables -A FORWARD -i $IF_SERV_VPN    -d 99.99.1.31  -j DROP
iptables -A FORWARD -i $IF_INSIDE      -d 10.1.3.0    -j DROP
iptables -A FORWARD -i $IF_INSIDE      -d 10.1.3.255  -j DROP

# ------------------------------------------------------------------------
# ------------ allowable traffic between local firewall and inside ----------
# ------------------------------------------------------------------------

# syslog from local firewall to inside syslog server
iptables -A OUTPUT -o $IF_INSIDE -p udp \
        -s 10.1.3.10 --sport 514 -d 10.1.3.22 --dport 514 -j ACCEPT

# dns from local firewall to inside dns server
iptables -A OUTPUT -o $IF_INSIDE -p udp \
        -s 10.1.3.10 --sport 1024:65535 -d 10.1.3.21 --dport 53 -j ACCEPT
iptables -A OUTPUT -o $IF_INSIDE -p udp \
        -s 10.1.3.10 --sport 53 -d 10.1.3.21 --dport 53 -j ACCEPT

# ssh from sysadmin workstations to local firewall
iptables -A INPUT -i $IF_INSIDE -p tcp -s 10.1.3.64/255.255.255.248 \
        --sport 1024:65535 -d 10.1.3.10 --dport 22 -j ACCEPT

# time request from firewall to inside ntp server
iptables -A OUTPUT -o $IF_INSIDE -p udp \
        -s 10.1.3.10 --sport 123 -d 10.1.3.20 --dport 123 -j ACCEPT

# ------------------------------------------------------------------------
# --------------- allowable traffic between border devices to ---------------
# --------------------- inside and public service nets ----------------------
# ------------------------------------------------------------------------

# syslog from border firewall to inside syslog-server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
        -s 10.1.1.5  --sport 514 -d 10.1.3.22 --dport 514 -j ACCEPT
# syslog from border router to inside syslog-server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
        -s 99.99.1.2 --sport 1024:65535 -d 10.1.3.22 --dport 514 -j ACCEPT
# ssh from sysadmin workstations to border router
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
        -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
        -d 99.99.1.2 --dport 22 -j ACCEPT
# ssh from sysadmin workstations to border firewall
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
        -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
```

```
        -d 10.1.1.5 --dport 22 -j ACCEPT
# ntp from border router to inside ntp-server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
        -s 99.99.1.2 --sport 123 -d 10.1.3.20 --dport 123 -j ACCEPT
# radius from border firewall to inside radius-server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
      -s 10.1.1.5  --sport 1812:1813 -d 10.1.3.23 --dport 1812:1813 -j
ACCEPT
# tftp from border firewall to inside tftp-server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
        -s 10.1.1.5 --sport 1024:65535 -d 10.1.3.20 --dport 69 -j ACCEPT
# tftp from border router to inside tftp-server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
        -s 99.99.1.2 --sport 1024:65535 -d 10.1.3.20 --dport 69 -j ACCEPT
######
###### the following two rules accomodate an xinetd behavior that sends
###### highport-highport udp traffic.  the tftp/xinetd daemons need to be
###### either fixed or replaced so that this is not necessary. the external
###### firewall lessens the exposure here since it handles this gracefully,
###### but it still needs improvement.
######
# tftp transport port negotiation between border router and inside tftp-
server
# try to find a tftp configuration that does not use high-high ports!
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p udp \
    -s 10.1.3.20 --sport 1024:65535 -d 99.99.1.2 --dport 1024:65535 -j
ACCEPT
# tftp transport port negotiation between border firewall and tftp-server
# try to find a tftp configuration that does not use high-high ports!
# the firewall seems to always negotiate port 12345.
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p udp \
        -s 10.1.3.20 --sport 1024:65535 -d 10.1.1.5 --dport 12345 -j
ACCEPT
######
######

# ------------------------------------------------------------------------
# ---------- allowable traffic from outside to public service net -----------
# ------------------------------------------------------------------------

# smtp from anywhere outside to public service net smtp server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_PUBLIC -p tcp \
        -s any/0 --sport 1024:65535 -d 10.1.2.5 --dport 25 -j ACCEPT

# dns from anywhere outside to public service net dns server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_PUBLIC -p udp \
        -s any/0 --sport 1024:65535 -d 10.1.2.6 --dport 53 -j ACCEPT
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_PUBLIC -p udp \
        -s any/0 --sport 53 -d 10.1.2.6 --dport 53 -j ACCEPT

# http from anywhere outside to public service net web server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_PUBLIC -p tcp \
        -s any/0 --sport 1024:65535 -d 10.1.2.7 --dport 80 -j ACCEPT

# https from anywhere outside to public service net web server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_PUBLIC -p tcp \
        -s any/0 --sport 1024:65535 -d 10.1.2.7 --dport 443 -j ACCEPT
```

```
# ------------------------------------------------------------------------
# -- allowable vpn employee access from outside to inside and public nets ---
# ------------------------------------------------------------------------

# ssh from employee vpn clients to anywhere inside
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 10.99.1.0/255.255.255.0 --sport 1024:65535 \
         -d 10.1.3.0/255.255.255.0 --dport 22 -j ACCEPT

# dns from employee vpn clients to inside dns server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
         -s 10.99.1.0/255.255.255.0 --sport 1024:65535 \
         -d 10.1.3.21 --dport 53 -j ACCEPT
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
         -s 10.99.1.0/255.255.255.0 --sport 53 \
         -d 10.1.3.21 --dport 53 -j ACCEPT

# pop3 mail from employee vpn clients to inside mail server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 10.99.1.0/255.255.255.0 --sport 1024:65535 \
         -d 10.1.3.24 --dport 110 -j ACCEPT

# http from employee vpn clients to inside web-based applications
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 10.99.1.0/255.255.255.0 --sport 1024:65535 \
         -d 10.1.3.96/255.255.255.224  --dport 80 -j ACCEPT

# https from employee vpn clients to inside web-based applications
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 10.99.1.0/255.255.255.0 --sport 1024:65535 \
         -d 10.1.3.96/255.255.255.224  --dport 443 -j ACCEPT

# citrix-ica from employee vpn clients to inside web-based applications
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 10.99.1.0/255.255.255.0 --sport 1024:65535 \
         -d 10.1.3.96/255.255.255.224  --dport 1494 -j ACCEPT

# ------------------------------------------------------------------------
# --- allowable vpn partner access from outside to inside and public nets ---
# ------------------------------------------------------------------------

# dns from partner vpn clients to inside dns server
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
         -s 202.120.25.64/255.255.255.224 --sport 1024:65535 \
         -d 10.1.3.21 --dport 53 -j ACCEPT
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p udp \
         -s 202.120.25.64/255.255.255.224 --sport 53 \
         -d 10.1.3.21 --dport 53 -j ACCEPT

# http from partner vpn clients to inside web-based applications
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 202.120.25.64/255.255.255.224 --sport 1024:65535 \
         -d 10.1.3.96/255.255.255.224  --dport 80 -j ACCEPT

# https from partner vpn clients to inside web-based applications
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 202.120.25.64/255.255.255.224 --sport 1024:65535 \
         -d 10.1.3.96/255.255.255.224  --dport 443 -j ACCEPT
```

```
# citrix-ica from partner vpn clients to inside web-based applications
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_INSIDE -p tcp \
         -s 202.120.25.64/255.255.255.224 --sport 1024:65535 \
         -d 10.1.3.96/255.255.255.224  --dport 1494 -j ACCEPT

# ------------------------------------------------------------------------
# ------ allowable vpn supplier access from outside to vpn service net ------
# ------------------------------------------------------------------------

# http from supplier vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 202.100.75.32/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.25 --dport 80 -j ACCEPT

# https from supplier vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 202.100.75.32/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.25 --dport 443 -j ACCEPT

# sftp from supplier vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 202.100.75.32/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.25 --dport 115 -j ACCEPT

# ftp from supplier vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 202.100.75.32/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.25 --dport 20:21 -j ACCEPT

# ------------------------------------------------------------------------
# ------ allowable vpn customer access from outside to vpn service net ------
# ------------------------------------------------------------------------

# http from customer vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 207.46.230.192/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.26 --dport 80 -j ACCEPT

# https from customer vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 207.46.230.192/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.26 --dport 443 -j ACCEPT

# sftp from customer vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 207.46.230.192/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.26 --dport 115 -j ACCEPT

# ftp from customer vpn clients to web-based application
iptables -A FORWARD -i $IF_OUTSIDE -o $IF_SERV_VPN -p tcp \
         -s 207.46.230.192/255.255.255.224 --sport 1024:65535 \
         -d 99.99.1.26 --dport 20:21 -j ACCEPT

# ------------------------------------------------------------------------
# -------- allowable traffic from vpn service net to inside ----------------
# ------------------------------------------------------------------------
```

```
# syslog from all vpn access servers to inside syslog server
iptables -A FORWARD -i $IF_SERV_VPN -o $IF_INSIDE -p udp \
         -s 99.99.1.24/255.255.255.248 --sport 514 \
         -d 10.1.3.22 --dport 514 -j ACCEPT
# ntp from all vpn access servers to inside ntp server
iptables -A FORWARD -i $IF_SERV_VPN -o $IF_INSIDE -p udp \
         -s 99.99.1.24/255.255.255.248 --sport 123 \
         -d 10.1.3.20 --dport 123 -j ACCEPT

# ------------------------------------------------------------------------
# --------- allowable traffic from public service net to outside ---------
# ------------------------------------------------------------------------

# ntp requests from public service net ntp server to outside ntp providers
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_OUTSIDE -p udp \
         -s 10.1.2.5 --sport 123 -d 128.118.25.3   --dport 123 -j ACCEPT
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_OUTSIDE -p udp \
         -s 10.1.2.5 --sport 123 -d 132.236.56.250 --dport 123 -j ACCEPT
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_OUTSIDE -p udp \
         -s 10.1.2.5 --sport 123 -d 198.82.162.213 --dport 123 -j ACCEPT

# dns requests from public service net dns server to outside world
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_OUTSIDE -p udp \
         -s 10.1.2.6 --sport 1024:65535 -d any/0 --dport 53 -j ACCEPT
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_OUTSIDE -p udp \
         -s 10.1.2.6 --sport 53 -d any/0 --dport 53 -j ACCEPT

# smtp from public mail server to outside world
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_OUTSIDE -p tcp \
         -s 10.1.2.5 --sport 1024:65535 -d any/0 --dport 25 -j ACCEPT

# ------------------------------------------------------------------------
# ---------- allowable traffic from public service net to inside ---------
# ------------------------------------------------------------------------

# syslog from all public service net hosts to inside server
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_INSIDE -p udp \
        -s 10.1.2.0/255.255.255.0 --sport 514 -d 10.1.3.22 --dport 514 -j
ACCEPT

# smtp from public mail server to inside mail server
iptables -A FORWARD -i $IF_SERV_PUBLIC -o $IF_INSIDE -p tcp \
         -s 10.1.2.7 --sport 1024:65535 -d 10.1.3.24 --dport 25 -j ACCEPT

# ------------------------------------------------------------------------
# ----------- allowable traffic from inside to public service net --------
# ------------------------------------------------------------------------

# ssh from inside admin workstations to all public service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 10.1.2.0/255.255.255.0    --dport 22 -j ACCEPT

# http from inside admin workstations to public net web server
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 10.1.2.7 --dport 80 -j ACCEPT
```

```
# http from inside general users to public net web server
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d 10.1.2.7 --dport 80 -j ACCEPT

# https from inside admin workstatons to public net web server
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 10.1.2.7 --dport 443 -j ACCEPT

# https from inside general users to public net web server
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d 10.1.2.7 --dport 443 -j ACCEPT

# smtp from inside mail server to public net mail server
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p tcp \
         -s 10.1.3.24 --sport 1024:65535 -d 10.1.2.5 --dport 25 -j ACCEPT

# ntp requests from internal ntp server to public service net ntp server
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p udp \
         -s 10.1.3.20 --sport 123 -d 10.1.2.5 --dport 123 -j ACCEPT

# dns forwarding from internal dns server to public service net dns server
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p udp \
         -s 10.1.3.21 --sport 1024:65535 -d 10.1.2.6 --dport 53 -j ACCEPT
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_PUBLIC -p udp \
         -s 10.1.3.21 --sport 53 -d 10.1.2.6 --dport 53 -j ACCEPT

# ------------------------------------------------------------------------
# ------------- allowable traffic from inside to vpn service net ---------
# ------------------------------------------------------------------------

# ssh from inside admin workstations to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 22 -j ACCEPT

# http from inside admin workstations to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 80 -j ACCEPT

# https from inside admin workstations to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 443 -j ACCEPT

# sftp from inside admin workstations to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 115 -j ACCEPT

# ftp from inside admin workstations to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 20:21 -j ACCEPT
```

```
# ssh from inside general users to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 22 -j ACCEPT

# http from inside general users to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 80 -j ACCEPT

# https from inside general users to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 443 -j ACCEPT

# sftp from inside general users to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 115 -j ACCEPT

# ftp from inside general users to all vpn service net
iptables -A FORWARD -i $IF_INSIDE -o $IF_SERV_VPN -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d 99.99.1.24/255.255.255.0  --dport 20:21 -j ACCEPT

# ------------------------------------------------------------------------
# --------------- allowable traffic from inside to outside ---------------
# ------------------------------------------------------------------------

# http from general users to outside world
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d any/0 --dport 80 -j ACCEPT

# https from general users to outside world
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d any/0 --dport 443 -j ACCEPT

# ftp from general users to outside world
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
         -s 10.1.3.128/255.255.255.128 --sport 1024:65535 \
         -d any/0 --dport 20:21 -j ACCEPT

# http from admin users to outside world
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d any/0 --dport 80 -j ACCEPT

# https from admin users to outside world
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d any/0 --dport 443 -j ACCEPT

# ftp from admin users to outside world
iptables -A FORWARD -i $IF_INSIDE -o $IF_OUTSIDE -p tcp \
         -s 10.1.3.64/255.255.255.248 --sport 1024:65535 \
         -d any/0 --dport 20:21 -j ACCEPT
```

```
# ------------------------------------------------------------------------
# ----------- log and drop the packets that do not match a rule ----------
# ------------------------------------------------------------------------

iptables -A INPUT   -j LOG  --log-level info
iptables -A OUTPUT  -j LOG  --log-level info
iptables -A FORWARD -j LOG  --log-level info
iptables -A INPUT   -j DROP
iptables -A OUTPUT  -j DROP
iptables -A FORWARD -j DROP
```

Assignment 3: Audit Your Security Architecture

Planning the Assessment

1. Scope of the assessment

   This assessment is intended to evaluate the effectiveness of the external firewall. Although hosts behind the firewall may be targets of network probing, the individual host level security is not the focus of the testing. The objective is to determine if the firewall is allowing only the services as outlined in the security policy. Testing will be limited to non-destructive probing so that servers are not likely to need restoration.

2. Define expectations

   The firewall will be tested to insure that it permits the desired services and denies everything else, according to the security policy. It needs to conceal the network structure behind it as much as possible. Some of the servers behind the firewall are expected to permit outside connections, so their presence cannot be hidden by the firewall. The firewall should prevent the detection of any servers that are not intended to be accessible from outside locations. The firewall itself must prevent unauthorized access, from outside or within. Firewall logs will be examined to ensure that proper audit trails of permitted and denied services is recorded.

3. Impact upon business operations

   Comprehensive scanning of perimeter security components can place a substantial burden upon the overall system. Furthermore, individual components may become overloaded and fail to function correctly. Such disruptions cannot occur during normal business hours. Since GIAC Enterprises' revenues are primarily derived from a business-to-business model, a suitable weekend will be selected to run the assessment. Business partners, customers, and suppliers will be notified one month in advance.

4. Required resources

   a. Physical access to the building and all computer equipment
   b. Contact names and phone numbers of key business and technical personnel
   c. Diagrams indicating all networks and hosts, including their addresses and purposes
   d. Onsite administrators that are knowledgeable about both the network and the hosts
   e. Backup tapes, backup procedures, and restore procedures
   f. Network scanning tools and network sniffers

5. Scheduling and Resource issues

a. Since GIAC Enterprises has relationships with businesses in Asia, the network assessment cannot occur during business hours in Asia. This means that the testing cannot start before 7:00AM on a Saturday morning.
b. The system must be fully operational by 9:00AM on the Monday morning following the tests so that GIAC Enterprises can resume normal business operations.
c. The tests should conclude early enough so as to allow for system reboots and possible system restorations if any damage should occur. The system administrators estimate no more than 8 hours for these activities, in a worst-case scenario. This places the required completion time for testing at no later than 1:00AM Monday morning.
d. Holiday weekends are to be avoided since key personnel might be harder to contact.

6. Cost estimates

The following cost estimates use 8 hours per person as a "day"; therefore, 2 persons working for 4 hours constitutes 1 billable day. Since GIAC Enterprises is performing the scans utilizing in-house personnel, an average of the IT staff's hourly rate will be used.

Planning	2.0 days	this includes backups and reviewing recovery procedures
Testing	1.5 day	3 administrators X 4 hours
Review	2.0 days	this includes IT review, and presentation to management
Total	5.5 days	$40.00 per hour X 8 hours per day X 5.5 days
Cost	$1760	

7. Preparations

a. The IT staff designs tests that satisfy the expectations described in Item 2
b. All required system backup and restoration documentation is gathered and reviewed
c. Key personnel contact information is collected in case of unexpected problems
d. Scanning and sniffing equipment is checked for readiness
e. Physical access to the building is checked including keys, alarm procedures, etc.
f. Backups are run before the testing starts

8. Implementation

a. Backups are run.
b. Testing starts.
c. Results are monitored as the testing runs. If either the testing process or servers on the network fail, the IT supervisor will determine the recovery process. This might conclude the weekend's testing if a key server is down. If the problem is easily recovered, testing resumes.
d. System logs are briefly reviewed and collected for further analysis

9. Disaster recovery

The IT supervisor has the right to terminate all testing at any time. Management decided that system availability on Monday morning has precedence over system testing. The tests may be rescheduled.

If the testing process halts due to unforeseen problems, it may be restarted from the beginning if time permits, or it may be resumed from the point of failure if possible.

If a critical system component fails, testing stops. The supervisor must be immediately notified. The supervisor will work with the proper administrator of the failed component to bring that item back to operational status. The administrators must get the supervisor's approval to perform a system restore if necessary. System reboots should be anticipated.

10. Review the results

    The IT staff reviews the log files from the system components and from the testing tools. The supervisor prepares a report that is reviewed by the IT staff before presented to management. Unexpected results are noted along with any recommendations. If any system changes are required, they will be scheduled accordingly.

Implementing the Assessment

Testing Environment

Two sets of tests were performed.

In the first test, the scanning device was placed on the Ethernet segment outside the firewall using IP address 99.99.1.14. This test was intended to test the ingress filtering on the firewall since the firewall should not accept packets that have the GIAC public addresses as their source.

In the second test, the scanning device was placed on the Ethernet segment outside the border router using IP address 172.16.1.10. I purposely selected the placement of the scanning device outside of the 99.99.1.x range so that the firewall ingress filtering did not block all of the scans, thereby rendering them useless. Multiple tests were run with the purpose of exercising the firewall rules that control traffic flow into the GIAC network. In my home test lab, the typical access controls that block inbound RFC-1918 addresses in the 172.16.1.x range have been purposely omitted, and noted as such in the policies in the sections above. Given that a production network would not have the 172.16.1.x address range on the outside of its router, this ingress block would be in place.

Since I did not want the border router security policies to aid the firewall, I installed an ACL on the router that allowed 172.16.1.10 to send any IP or ICMP packets into the firewall, and I temporarily enabled ICMP-unreachable. These controls are still active on the two internal

firewalls, so the opening of this small slice of traffic to this one specific IP address for a few hours is an acceptable risk in order to get better results from the scan. The outside router interface configuration during the test is shown below, with the 2 additional ACL's in bold.

Border router configuration with permissive ACL's for 172.16.1.10

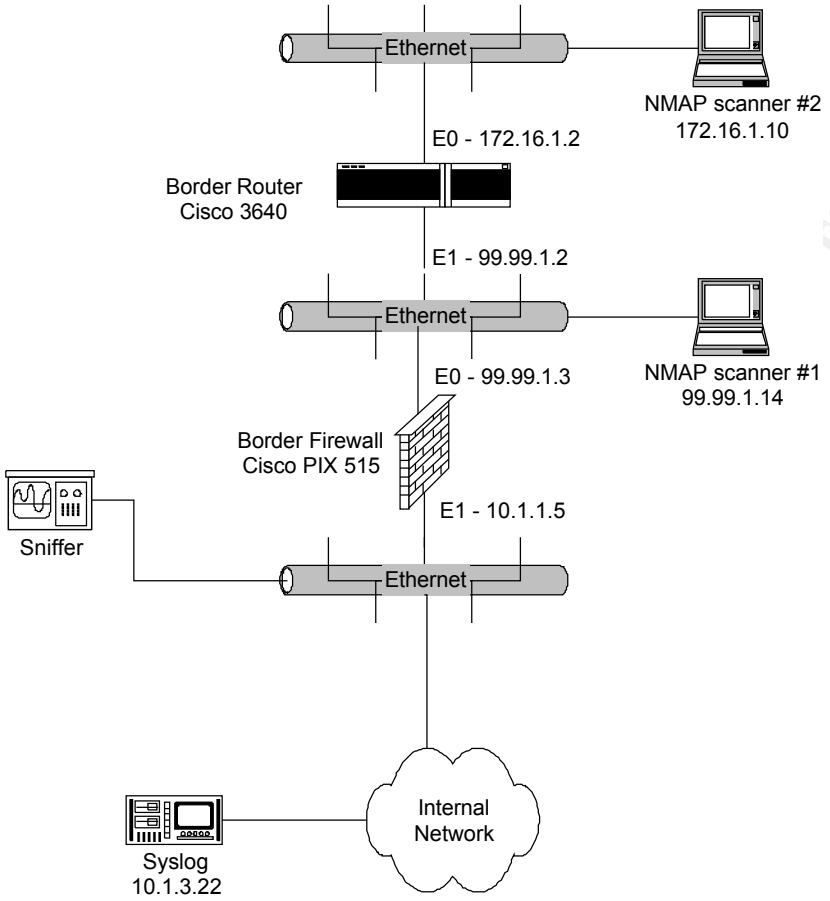
```
interface Ethernet0
 ip address 172.16.1.2 255.255.255.0
 ip access-group e0_acl_probes in
 no ip redirects
 no ip proxy-arp
 no cdp enable

ip access-list extended e0_acl_probes
 permit ip host 172.16.1.10 any
 permit icmp host 172.16.1.10 any
 deny   icmp any any log
 deny   ip 99.99.1.0 0.0.0.31 any log
 deny   ip 10.0.0.0 0.255.255.255 any log
 deny   ip 192.168.0.0 0.0.255.255 any log
 deny   ip 169.254.0.0 0.0.255.255 any log
 deny   ip 127.0.0.0 0.255.255.255 any log
 deny   ip host 0.0.0.0 any log
 deny   ip host 255.255.255.255 any log
permit tcp any 99.99.1.0 0.0.0.31
permit udp any 99.99.1.0 0.0.0.31
permit esp any host 99.99.1.3
permit ahp any host 99.99.1.3
deny   ip any any log
```

Testing and Monitoring Tools

The NMAP 2.53 scanning tool was used to probe the network. Packet flows were recorded on the inside of the firewall using TCPDUMP 3.4. Both of these tools are available for free on the Internet at http://www.insecure.org and http://www.tcpdump.org. In addition to the network traces, the border firewall reported denied attempts via syslog to the internal logging machine at 10.1.3.22. For each of the tests, a separate output file was collected from nmap, tcpdump, and syslog.

The diagram below shows the placement of the probing machine and the sniffer.



Assessment Test 1

This test verifies that the firewall ingress filters prevent the GIAC public addresses from entering the network. This guards against attacks that use spoofed GIAC addresses as their source. It also shows failures to connect directly to the firewall as a result of the overall architecture. A SYN can is used during the connection attempts. None of the connections succeed.

```
# Nmap (V. nmap) scan initiated 2.53 as: nmap -sS -P0 -vv -n -p 1-128 -oN
nmap-log00.txt 99.99.1.1 99.99.1.3 99.99.1.5
All 128 scanned ports on  (99.99.1.1) are: filtered
All 128 scanned ports on  (99.99.1.3) are: filtered
All 128 scanned ports on  (99.99.1.5) are: filtered
# Nmap run completed at Wed Oct  3 10:21:37 2001 -- 3 IP addresses (3 hosts
up) scanned in 1007 seconds
```

The 99.99.1.1 address is the outbound NAT address that is assigned to connections from the inside to the Internet. The 99.99.1.3 address is the external address of the firewall. The 99.99.1.5 address is the public address of the mail server on the public service net, which is translated to 10.1.2.5 via inbound NAT. These three addresses are representative of the pool of addresses that GIAC uses, and should show how the firewall treats each type of public address.

The attempts to connect to 99.99.1.5 fail since the ACL's against spoofing prevent it.

```
access-list acl_out deny ip 99.99.1.0 255.255.255.224 any
```

The denial is verified in the syslog records.

```
Oct  3 10:15:52 spock %PIX-4-106023: Deny tcp src outside:99.99.1.14/45134
dst inside:99.99.1.5/126 by access-group "acl_out"
Oct  3 10:15:52 spock %PIX-4-106023: Deny tcp src outside:99.99.1.14/45134
dst inside:99.99.1.5/58 by access-group "acl_out"
Oct  3 10:15:52 spock %PIX-4-106023: Deny tcp src outside:99.99.1.14/45134
dst inside:99.99.1.5/96 by access-group "acl_out"
```

Since the 99.99.1.1 address does not have static mapping in the PIX, we never even get to the point of checking the ACL's and it is rejected immediately, as is shown in the sample syslog entries below. Notice that the denial contains a "No xlate" message.

```
Oct  3 10:04:43 spock %PIX-3-106011: Deny inbound (No xlate) tcp src
outside:99.99.1.14/45134 dst outside:99.99.1.1/126
Oct  3 10:04:43 spock %PIX-3-106011: Deny inbound (No xlate) tcp src
outside:99.99.1.14/45134 dst outside:99.99.1.1/58
Oct  3 10:04:43 spock %PIX-3-106011: Deny inbound (No xlate) tcp src
outside:99.99.1.14/45134 dst outside:99.99.1.1/96
```

The PIX rejects the 99.99.1.3 connections since they are IPSEC packets. This is an interesting behavior that is inherent in the PIX. When an outside source attempts an unauthorized connection directly to the outside interface, the PIX examines it to determine if it is a valid IPSEC connection, since IPSEC connections are configured using this address. If the packet is not valid, it is rejected. The sample syslog messages are below.

```
Oct  3 10:11:15 spock %PIX-4-402106: Rec'd packet not an IPSEC packet. (ip)
dest_addr= 99.99.1.3, src_addr= 99.99.1.14, prot= tcp
```

Assessment Test 2

This test verifies that outside sources are not able to connect directly to the firewall. This is similar to the first test, except that our public addresses are not used as the source. A SYN can is used during the connection attempts. None of the connections succeed.

```
# Nmap (V. nmap) scan initiated 2.53 as: nmap -sS -O -P0 -F -vv -n -oN nmap-
log01.txt 99.99.1.1 99.99.1.3
Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
All 1062 scanned ports on  (99.99.1.1) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
All 1062 scanned ports on  (99.99.1.3) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

# Nmap run completed at Wed Oct  3 13:43:28 2001 -- 2 IP addresses (2 hosts
up) scanned in 4120 seconds
```

The 99.99.1.1 address is the outbound NAT address that is assigned to connections from the inside to the Internet. The 99.99.1.3 address is the external address of the firewall. Neither of the addresses allowed a connection. The nmap option "-F" scans for all service ports that are listed in its services file, which includes over 2000 ports. The PIX did not offer any services as is shown in the scanner output above.

Since the 99.99.1.1 address does not have static mapping in the PIX, we never even get to the point of checking the ACL's and it is rejected immediately, as is shown in the sample syslog entries below. Notice that the denial contains a "No xlate" message.

```
Oct  3 12:34:41 spock %PIX-3-106011: Deny inbound (No xlate) tcp src
outside:172.16.1.10/44815 dst outside:99.99.1.1/1409
Oct  3 12:34:41 spock %PIX-3-106011: Deny inbound (No xlate) tcp src
outside:172.16.1.10/44815 dst outside:99.99.1.1/775
Oct  3 12:34:41 spock %PIX-3-106011: Deny inbound (No xlate) tcp src
outside:172.16.1.10/44815 dst outside:99.99.1.1/135
```

The PIX rejects the 99.99.1.3 connections since they are IPSEC packets. This is an interesting behavior that is inherent in the PIX. When an outside source attempts an unauthorized connection directly to the outside interface, the PIX examines it to determine if it is a valid IPSEC connection, since IPSEC connections are configured using this address. If the packet is not valid, it is rejected. The sample syslog messages are below.

```
Oct  3 13:09:33 spock %PIX-4-402106: Rec'd packet not an IPSEC packet. (ip)
dest_addr= 99.99.1.3, src_addr= 172.16.1.10, prot= tcp
Oct  3 13:09:39 spock %PIX-4-402106: Rec'd packet not an IPSEC packet. (ip)
dest_addr= 99.99.1.3, src_addr= 172.16.1.10, prot= tcp
Oct  3 13:09:46 spock %PIX-4-402106: Rec'd packet not an IPSEC packet. (ip)
dest_addr= 99.99.1.3, src_addr= 172.16.1.10, prot= tcp
```

Assessment Test 3

This test verifies that outside sources are not able to connect to the internal infrastructure subnet at 10.1.3.20 through 10.1.3.24. The Unix "netstat –an" command reports all open ports on a machine, and this command shows that the following ports are open collectively on the machines on this subnet: 22, 25, 53, 69, 123, 514, 1812, and 1813. The scan specified all of these ports, and included telnet-23, which is not running, just to compare its behavior. A SYN can is used during the connection attempts. None of the connections succeed.

```
# Nmap (V. nmap) scan initiated 2.53 as: nmap -sS -O -P0 -p
23,22,25,53,69,123,514,1812,1813 -vv -n -oN nmap-log02.txt 10.1.3.20-24
Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (10.1.3.20):
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain
69/tcp     filtered    tftp
123/tcp    filtered    ntp
514/tcp    filtered    shell
1812/tcp   filtered    unknown
1813/tcp   filtered    unknown

Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (10.1.3.21):
Port       State       Service
```

```
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain
69/tcp     filtered    tftp
123/tcp    filtered    ntp
514/tcp    filtered    shell
1812/tcp   filtered    unknown
1813/tcp   filtered    unknown

Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (10.1.3.22):
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain
69/tcp     filtered    tftp
123/tcp    filtered    ntp
514/tcp    filtered    shell
1812/tcp   filtered    unknown
1813/tcp   filtered    unknown

Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (10.1.3.23):
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain
69/tcp     filtered    tftp
123/tcp    filtered    ntp
514/tcp    filtered    shell
1812/tcp   filtered    unknown
1813/tcp   filtered    unknown

Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
```

```
Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (10.1.3.24):
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain
69/tcp     filtered    tftp
123/tcp    filtered    ntp
514/tcp    filtered    shell
1812/tcp   filtered    unknown
1813/tcp   filtered    unknown

Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

# Nmap run completed at Wed Oct  3 14:23:39 2001 -- 5 IP addresses (5 hosts
up) scanned in 1278 seconds
```

In all of these scans, the access control list "acl_out" bound to the outside interface denies these attempts. There is no rule that allows them, so the default "deny all" rule stops them.

```
access-list acl_out deny ip any any
```

A sampling of the syslog entries from this scan is below.

```
Oct  3 14:02:13 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61061
dst inside:10.1.3.20/23 by access-group "acl_out"
Oct  3 14:02:13 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61061
dst inside:10.1.3.20/22 by access-group "acl_out"
Oct  3 14:06:32 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61061
dst inside:10.1.3.21/123 by access-group "acl_out"
Oct  3 14:06:32 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61061
dst inside:10.1.3.21/1812 by access-group "acl_out"
Oct  3 14:10:51 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61062
dst inside:10.1.3.22/25 by access-group "acl_out"
Oct  3 14:10:51 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61062
dst inside:10.1.3.22/69 by access-group "acl_out"
Oct  3 14:15:10 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61063
dst inside:10.1.3.23/53 by access-group "acl_out"
Oct  3 14:15:10 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61063
dst inside:10.1.3.23/514 by access-group "acl_out"
Oct  3 14:19:42 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61066
dst inside:10.1.3.24/1812 by access-group "acl_out"
Oct  3 14:19:42 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/61066
dst inside:10.1.3.24/123 by access-group "acl_out"
```

Assessment Test 4

This test verifies that outside sources are not able to connect to the internal subnets for the system administrators at 10.1.3.64/255.255.255.240, applications and databases at 10.1.3.96/255.255.255.224, and user desktops at 10.1.3.128/255.255.255.128. My test lab does not have any machines in the applications or desktop subnets. The only administrator machine was reconnected as the scanner on the outside. As a result, there were no targets in these ranges, but we can still tell if the firewall lets the packets through. In this scan, I picked a few common ports as examples, and I selected a random address with each of the target ranges. A SYN can is used during the connection attempts. None of the connections pass through the firewall.

```
# Nmap (V. nmap) scan initiated 2.53 as: nmap -sS -P0 -p 22,23,25,53 -vv -n -
oN nmap-log03.txt 10.1.3.70 10.1.3.100 10.1.3.150
Interesting ports on  (10.1.3.70):
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain

Interesting ports on  (10.1.3.100):
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain

Interesting ports on  (10.1.3.150):
Port       State       Service
22/tcp     filtered    ssh
23/tcp     filtered    telnet
25/tcp     filtered    smtp
53/tcp     filtered    domain

# Nmap run completed at Wed Oct  3 14:29:31 2001 -- 3 IP addresses (3 hosts
up) scanned in 108 seconds
```

The 10.1.3.64 and 10.1.3.128 ranges have mappings from the inside to the outside interface via NAT and GLOBAL statements. These are outbound mappings. Since these are inbound connections, they are rejected before they even get to the ACL checking.

```
global (outside) 1 99.99.1.1
nat (inside) 1 10.1.3.64 255.255.255.248 0 0
nat (inside) 1 10.1.3.128 255.255.255.128 0 0
```

Sample syslog messages are below:

```
Oct  3 14:27:35 spock %PIX-3-106010: Deny inbound tcp src
outside:172.16.1.10/55973 dst inside:10.1.3.70/53
Oct  3 14:27:35 spock %PIX-3-106010: Deny inbound tcp src
outside:172.16.1.10/55973 dst inside:10.1.3.70/22
Oct  3 14:28:53 spock %PIX-3-106010: Deny inbound tcp src
outside:172.16.1.10/55974 dst inside:10.1.3.150/25
Oct  3 14:28:59 spock %PIX-3-106010: Deny inbound tcp src
outside:172.16.1.10/55975 dst inside:10.1.3.150/53
```

Since the 10.1.3.96 range has a static mapping to allow partners to reach this net via a VPN connection, the access lists are examined to see if this traffic should be allowed. No rule exists, so the "catch-all" deny rule applies.

```
access-list acl_out deny ip any any
```

Sample syslog entries for this activity:

```
Oct  3 14:28:11 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/55973
dst inside:10.1.3.100/53 by access-group "acl_out"
Oct  3 14:28:11 spock %PIX-4-106023: Deny tcp src outside:172.16.1.10/55973
dst inside:10.1.3.100/22 by access-group "acl_out"
```

Assessment Test 5

This test verifies that outside sources are able to connect to the available services on the public service net. It also verifies that the outside cannot connect to the private service net for customers and suppliers. The Unix "netstat –an" command reports all open ports on a machine, and this command shows that the following ports are open collectively on the machines on these subnets: 22, 25, 53, 80, 123, and 443. The scan specified all of these ports, and included FTP-21, which is not running, just to compare its behavior. A SYN can is used during the connection attempts. None of the connections succeed on the private net. All of the connections succeed on the public net, except for DNS, which is only available via UDP. Since we should see successful connections, I included the O/S fingerprinting option "-O". None of the machines revealed a fingerprint. For brevity, I included the fingerprint report from only the first machine.

```
# Nmap (V. nmap) scan initiated 2.53 as: nmap -sS -O -P0 -p
21,22,25,53,80,123,443 -vv -n -oN nmap-log04.txt 99.99.1.5-7 99.99.1.25-26
Interesting ports on  (99.99.1.5):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     open        smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)

Sequence numbers: 4418E99B 47395754 4B78FA6B 4F4CE192 51D48D55 3AF3FC2A
No OS matches for host (If you know what OS is running on it, see
http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
TSeq(Class=TR)
T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=N)
T4(Resp=N)
T5(Resp=N)
T6(Resp=N)
```

```
T7(Resp=N)
PU(Resp=N)

Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (99.99.1.6):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

Interesting ports on  (99.99.1.7):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     open        http
123/tcp    filtered    ntp
443/tcp    open        https

Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (99.99.1.25):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

Warning:  No TCP ports found open on this machine, OS detection will be
MUCH less reliable
Interesting ports on  (99.99.1.26):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

# Nmap run completed at Wed Oct  3 14:45:06 2001 -- 5 IP addresses (5 hosts
up) scanned in 828 seconds
```

To test the DNS services, I temporarily connected my Win2K desktop to the network on the outside of the router, pointed my DNS to the GIAC name server, and issued a lookup.

```
C:\>ipconfig /all

Windows 2000 IP Configuration
        Host Name . . . . . . . . . . . . : alice
        Primary DNS Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Broadcast
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
Ethernet adapter cable:
        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : Winbond W89C940 PCI Ethernet Adapter
        Physical Address. . . . . . . . . : 00-20-78-18-1E-75
        DHCP Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 172.16.1.4
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 172.16.1.1
        DNS Servers . . . . . . . . . . . : 99.99.1.6
        NetBIOS over Tcpip. . . . . . . . : Disabled

C:\>nslookup www.giac.com
Server:  dns.giac.com
Address:  99.99.1.6

Name:    www.giac.com
Address:  99.99.1.7
```

Assessment Test 6

This test verifies that outside sources are not able to detect available services on the public and private services nets using fragmented ACK scans. This exercises the built-in PIX sanity checks in terms of connection state and fragmentation. This test uses the same targets and ports as the SYN scans in test 5. None of the scans succeed.

```
# Nmap (V. nmap) scan initiated 2.53 as: nmap -sA -f -P0 -p
21,22,25,53,80,123,443 -vv -n -oN nmap-log05.txt 99.99.1.5-7 99.99.1.25-26
Interesting ports on  (99.99.1.5):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
```

```
443/tcp    filtered    https

Interesting ports on  (99.99.1.6):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

Interesting ports on  (99.99.1.7):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

Interesting ports on  (99.99.1.25):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

Interesting ports on  (99.99.1.26):
Port       State       Service
21/tcp     filtered    ftp
22/tcp     filtered    ssh
25/tcp     filtered    smtp
53/tcp     filtered    domain
80/tcp     filtered    http
123/tcp    filtered    ntp
443/tcp    filtered    https

# Nmap run completed at Wed Oct  3 14:51:41 2001 -- 5 IP addresses (5 hosts
up) scanned in 181 seconds
```

The packets are all rejected due to their odd fragmentation. Note the sample syslog entries below and the PIX complaint about the header and packet length.

```
Oct  3 14:48:32 spock %PIX-5-500003: Bad TCP hdr length (hdrlen=20,
pktlen=16) from 172.16.1.10/43004 to 99.99.1.5/21, flags: ACK , on
interface outside
Oct  3 14:48:32 spock %PIX-5-500003: Bad TCP hdr length (hdrlen=20,
pktlen=16) from 172.16.1.10/43004 to 99.99.1.5/22, flags: ACK , on
interface outside
Oct  3 14:49:38 spock %PIX-5-500003: Bad TCP hdr length (hdrlen=20,
pktlen=16) from 172.16.1.10/43009 to 99.99.1.6/21, flags: ACK , on
interface outside
Oct  3 14:49:44 spock %PIX-5-500003: Bad TCP hdr length (hdrlen=20,
```

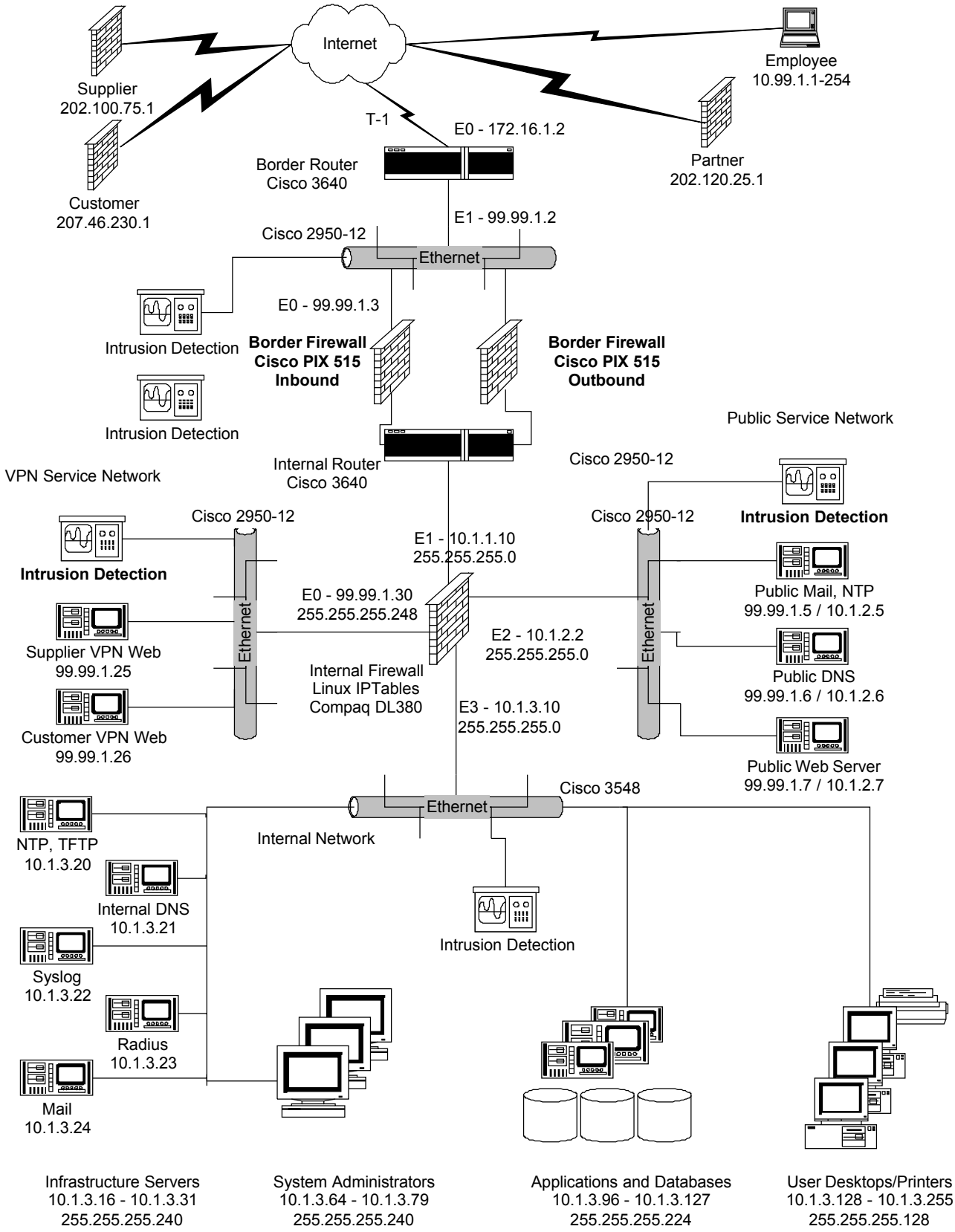
```
pktlen=16) from 172.16.1.10/43004 to 99.99.1.7/21, flags: ACK , on
interface outside
```

Perimeter Analysis

Although the system passed the scanning tests, this only shows that the perimeter is correctly configured. This is not a demonstration that the architecture is "hack-proof". Due to budget constraints, the IT staff was unable to get approval for two additional servers that would serve as intrusion detection devices for the public net and for the private net. These devices should be added when profits allow.

Another shortcoming is the use of one firewall for both inbound and outbound connections. By separating this traffic across two firewalls, the remote VPN clients would not need to use the split-tunnel architecture. The split-tunnel could conceivably allow a compromised remote user's PC to transmit information back to an attacker over the unencrypted half of the connection, while the user is connected via IPSEC to the GIAC network. By implementing inbound and outbound firewalls, the split-tunnel could be eliminated. Remote employee traffic destined for the Internet while connected via IPSEC could enter via the inbound firewall, route to the outbound firewall, and travel to the outside target. The user's PC would not be directly exposed to the Internet while connected to GIAC. As an interim measure, desktop firewall software and home-based hardware firewalls such as the Linksys BEFSR41 can help to mitigate the exposure.

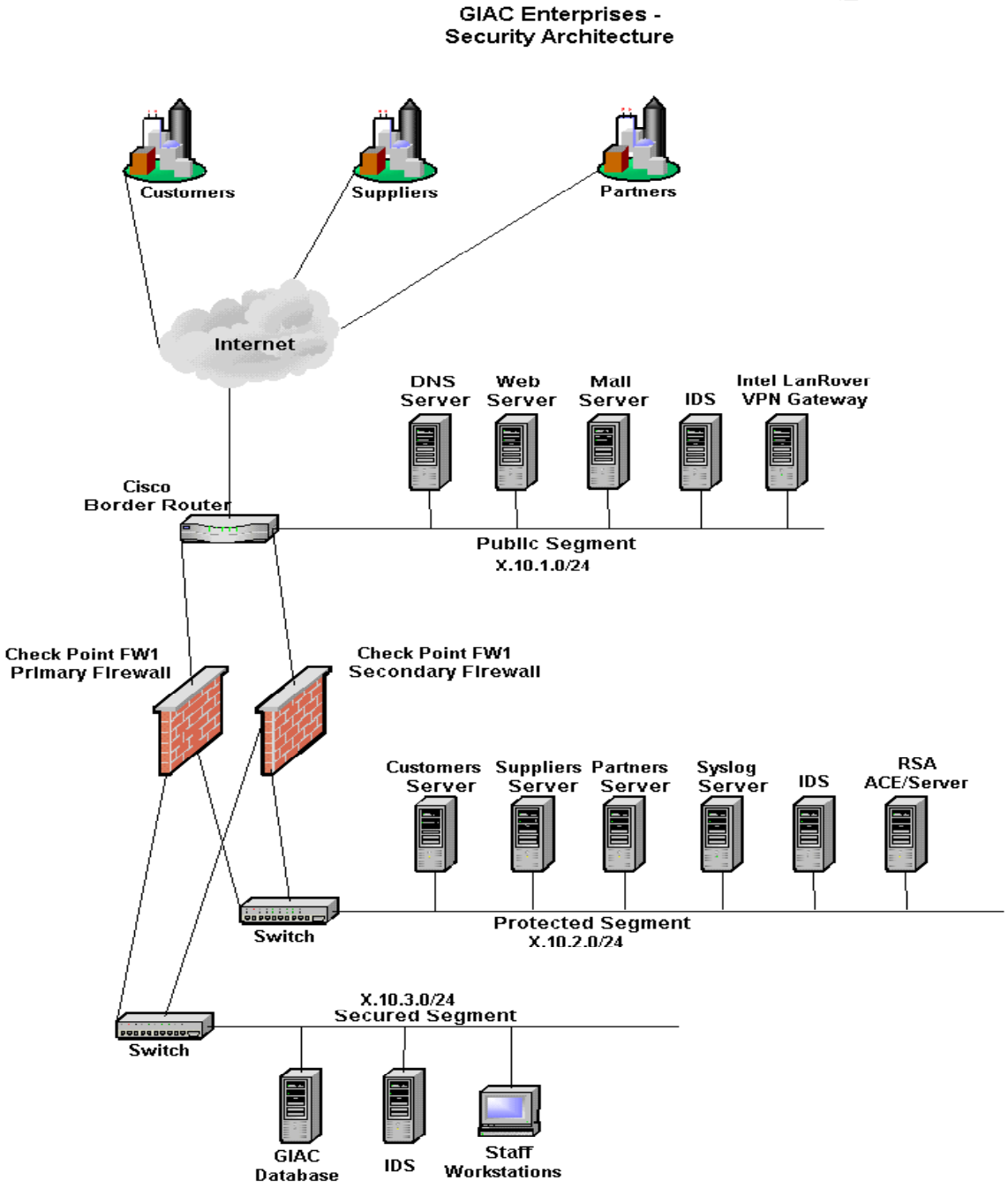
The diagram below shows these improvements.



Assignment 4: Design Under Fire

MeauHuat Tan wrote the GCFW practical that I chose to review. The practical can be found at: http://www.sans.org/y2k/practical/MeauHuat_Tan_GCFW.doc

The following diagram depicts the network:



Firewall Attack

According to page 4 of the practical, the external firewall is running Check Point FW1 (version 4.1 SP3). According to the CERT description of this vulnerability "Firewall-1 and VPN-1 include support for RDP, but do not provide adequate security controls for RDP data. By adding a faked RDP header to typical UDP traffic, any content can be passed to port 259 on any host on either side of the device. An attacker who exploits this vulnerability can build a tunnel to bypass the firewall and pass traffic to and from arbitrary hosts on either side of the firewall on port 259." This vulnerability advisory can be found at http://www.kb.cert.org/vuls/id/310295. The initial discovery was published at http://www.inside-security.de/advisories/fw1_rdp.html. The proof-of-concept code is at http://www.inside-security.de/advisories/fw1_rdp_poc.c

I have not done any serious programming for a while, but with the help of a few friends who are great programmers, a possible attack could be constructed. The vulnerability basically lets an attacker create a tunnel over port UDP-259 across the firewall. To take advantage of this vulnerability, an inside host would need to be compromised with a Trojan that operates on source and destination port UDP-259. Another requirement is that the border router will let this traffic through, which according to the router configuration in Assignment 2 of the practical, this traffic is permitted. The toughest part is getting the Trojan installed onto an internal machine. The best probability is to install the Trojan via email. Since the users workstations are on the same segment as the GIAC databases, a tunnel to the most protected network could be established.

There are a lot of maybe's here. The attack depends upon some solid programming skills. We need to develop the attacker's client, and the Trojan server. This is not trivial work. There is also an intrusion sensor on the databases segment, so we may be discovered quickly; depending upon how the sensor is configured and how faithfully it is maintained. The vulnerability was published back in July 2001, which is three months ago. Some installations are slow to patch their servers, so this may still be vulnerable, but maybe not.

The Trojan would just be a way to get in the door, monitor what goes on, and possibly replicate itself. It could then send keystrokes back to the attacker, including passwords, hostnames and addresses, and additional email addresses to infect. The attack would probably be discovered after a while, so attacks on the other servers would need to be done quickly.

Server Attack

The servers on the public network are very exposed. The border router is the only line of defense, and it has a minimum of filtering on it. All of these servers need to rely upon host level security since the router is a lot in. Web servers, mail servers, and DNS have a variety of attacks. The practical does not specify the exact versions of these services, so I will show of few examples of DNS attacks that could be attempted.

Many of the DNS attacks are version dependent. The first step is to determine the version of DNS that is running. Providing that the DNS allows version queries, we can issue the following command. Note: I disabled the protection on my DNS server momentarily to show this example.

```
[root@bigben restrict]# dig @10.1.3.21 version.bind chaos txt
```

An unprotected DNS configuration will show the following:

```
; <<>> DiG 8.3 <<>> @10.1.3.21 version.bind chaos txt
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;;      version.bind, type = TXT, class = CHAOS

;; ANSWER SECTION:
VERSION.BIND.          0S CHAOS TXT    "8.2.3-REL"

;; Total query time: 3 msec
;; FROM: bigben.talianek.com to SERVER: 10.1.3.21
;; WHEN: Wed Oct  3 21:37:32 2001
;; MSG SIZE  sent: 30  rcvd: 64
```

Notice the text "8.2.3-REL". This is the version.

Depending upon the version, we can select an attack. The Internet Software Consortium maintains a widely used BIND implementation of DNS. The security alerts for the various versions can be found at: http://www.isc.org/products/BIND/bind-security.html. CERT published an advisory on BIND DNS alerts at http://www.cert.org/advisories/CA-2001-02.html.

Some samples of well-published attacks are listed below.

Infoleak and TSIG bugs – BIND versions 8.2.x
http://packetstormsecurity.org/0102-exploits/bind8x.c
http://packetstormsecurity.org/0102-exploits/tsl_bind.c

NXT bug – BIND versions 8.2, 8.2 patchlevel 1, 8.2.1
http://packetstormsecurity.org/9911-exploits/adm-nxt.c
http://packetstormsecurity.org/0003-exploits/NXT-Howto.txt (This is clearly a written article)

There are many of these publicly available. When the NXT exploit was published, I setup a small test network with a set of DNS servers and I ran the exploit. It was not difficult. Some of the

exploits have even been automated into self-propagating worms. Many of the exploits to DNS require TCP connections. By placing an access list on the border router to stop the TCP connections, some vulnerability can be mitigated. A sample ACL for the router follows:

access-list 110 deny tcp any any eq 53 log

One mitigating factor in the design is that the public segment has an intrusion detection system. If it is closely monitored, at least the administrators will be alerted to intrusion attempts and possible compromises.

Denial-of-Service Attack

Denial-of-Service (DoS) attacks are very prevalent on the Internet. They make the headlines when they affect a large business or prominent government installation, but for the most part, they are a quiet danger that cripples many computer facilities regularly.

The danger comes from a variety of components. High-speed cable and DSL connections to a large portion of the population provide a source for very high-volume traffic attacks. Many businesses with T-1 or T-3 connections do not secure their perimeters properly. Since the source address is often spoofed, following the traffic back to the true source can become nearly an impossible task. Internet Service Providers are often do not provide the victims with the required support to stop the problem. Faulty applications, loosely configured operating systems, and lack of user education all contribute the large pool of compromised machines on the Internet that feed the DoS monster.

The basic concept is to flood a network or server with so much traffic, that it becomes saturated and cannot service valid requests. A simple calculation can demonstrate this rather clearly. If an attacker has access to 50 cable or DSL connected machines that each has a bandwidth of 150K, we can generate around 7.5Mb of traffic. Considering a single T-1 can handle just over 1Mb, the 50 home user machines would easily suffice.

Some of the original designs such as Trinoo, TFN, and Stacheldraht built communications networks with client/server infrastructures. One master can launch an army of attackers. This structure had its own peer-to-peer communications technique. Some links regarding these tools:
http://www.cert.org/advisories/CA-2000-01.html
http://www.cert.org/tech_tips/denial_of_service.html
http://staff.washington.edu/dittrich/misc/stacheldraht.analysis
http://www.sans.org/y2k/TFN_toolkit.htm

Some of the newer designs are using chat rooms as the method of communications. A chat room is setup, often on a compromised server and the owner of the server is unaware of the situation. The attacker installs software onto the cable/DSL user's machines that contacts the chat room. The software registers with the chat room, and the attacker can keep track of active participants.

He is able to give the attack commands from the chat room, and the army carries out the attack. A great story outlining this technique can be found at: http://grc.com/dos/grcdos.htm

Much of the software that you need to carry out various DoS attacks can be found at http://packetstormsecurity.org.

In launching a DoS attack against MeauHuat's network, I would target the DNS server and Web servers. Although he does not mention the specific web server version, if it is an NT4 web server, it is likely to crash upon such an attack. By also flooding the DNS server, address resolutions will probably stop well before the inbound communications line is saturated. An interesting technique would be to write a cable "bot" (attack robot that is installed on a cable modem user's machine) that makes valid UDP DNS requests. The source address can be spoofed easily since it is UDP. Not only does this flood the network with incoming requests from the cable bots, but it also consumes resources on the DNS server as it tries to process the requests. Some amplification occurs in the process since the DNS responses are larger than the requests. If the DNS server allows recursive queries for external domains, those domains would also suffer.

References

SANS Courseware - Firewalls 101: Perimeter Protection with Firewalls
Chris Brenton

SANS Courseware - Firewalls 102: Perimeter Protections and Defense, In-Depth
Lance Spitzner, C. Brenton, S. Winters, S. Northcutt

SANS Courseware - VPNs and Remote Access
Chris Brenton

Advanced Linux Firewall Configuration
Robert Ziegler, SANS 2001 Baltimore

TCP/IP Illustrated, Volume 1: The Protocols
W. Richard Stevens

DNS and BIND, 3rd Edition
Paul Albitz and Cricket Liu

Web Site – SANS
http://www.sans.org

Web Site – PacketStorm
http://packetstormsecurity.org

Web Site – CERT
http://www.cert.org

Web Site – Cisco – There are the entry points for the online manuals for IOS and PIX
http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix_61/index.htm
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/index.htm

Web Site – Red Hat Linux
http://www.redhat.com

Web Site – TCPDUMP
http://www.tcpdump.org

Web Site – Fyodor's NMAP
http://www.insecure.org

Web Site – Check Point Alerts
http://www.checkpoint.com/techsupport/alerts/