Global Information Assurance Certification Paper

A COMPREHENSIVE SECURITY PLAN FOR GIAC ENTERPRISES

GCFW Practical Assignment
Version 1.7

MARK E. DONALDSON
AUGUST 4, 2002

TABLE OF CONTENTS

INTRODUCTION

COMPANY HISTORY & BACKGROUND

Quite simply, the story of GIAC Enterprises is one of "riches-to-rags", and one of "fame-to-shame".

GIAC Enterprises was born in the late 1990's during the midst of the .COM boom by two penniless young men with a great idea and ambition to spare. These men recognized the internet to be a burgeoning, limitless, and largely untapped market. The also saw a demand for a steady supply of clever and appropriately written fortunes for the fortune cookie market. Thus, GIAC Enterprises came to be.

GIAC Enterprises grew rapidly as web sales boomed. New equipment was purchased and installed hurriedly to keep pace with sales, and the workforce grew accordingly. Due to this rapid growth and expansion, GIAC Enterprises became a Windows shop exclusively, as out-of-the-box system performance was required and technical expertise was at a premium. Consequently, functionality and productivity was given precedence. Security was viewed as a hindrance, an agent of restraint, and a gauntlet-to-gold. It was, therefore, not considered seriously as an issue nor was it included in the business plan.

By the year 2000, GIAC Enterprises was a wealthy company of 300 employees and a large base of suppliers to meet the demand for new fortunes. Additionally, GIAC Enterprises brought on several partners to assist with world wide expansion of the lucrative operation. Indeed, GIAC Enterprises was the "darling" of the .COM world and a model eagerly copied by numerous upstarts. Plans were in the works to go IPO.

However, GIAC Enterprises "fortunes" began to change rapidly. By early 2001, the .COM world was experiencing serious financial disruption and upheaval. Numerous DOT COMS were suffering financial collapse and the "domino effect" was wrecking havoc in the world of e-commerce. Additionally, a series of direct "misfortunes" soon followed.

In February of 2001, GIAC Enterprises IIS web servers were repeatedly hacked and defaced by Chinese hackers. On July 19, 2001, GIAC Enterprises servers were struck by the Code Red Worm, shutting down operations and sales for nearly a week. On September 18, 2001, GIAC systems became infected with the Nimda Virus. Again, systems were shut down and sales were disrupted. Sales began to decline and the company soon came under fire.

As sales slumped, layoffs followed. During the height of the 2001 holiday sales season, a disgruntled former employee launched a Smurf attack and a series of other DOS attacks on the company systems, costing them millions in sales. At about the same

time, spammers discovered GIAC Enterprises unsecured mail server as an excellent relay agent for various distributions. GIAC's ISP eventually isolated the problem, but they did not take kindly to such a needless security failure.

In March of 2002, another former employee successfully installed the BackOrifice Trojan on several sensitive computers. He was thus able to steal the entire customer credit card database and publicly publish the information on the internet. The exploit was widely publicized, and the name "GIAC Enterprises" quickly became a four-letter-word in the IT world.

GIAC Enterprises itself was now on the verge of financial collapse. With an exhausted bank account and a remaining employee base of approximately 35, GIAC Enterprises contacted Bandwidthco Security Services in an attempt to salvage the business. This document is a comprehensive security plan as prepared by Bandwidthco Security Services and submitted to GIAC Enterprises for their acceptance and implementation.

ASSUMPTIONS

If accepted, the network and security architecture in this proposal must be thoroughly tested in a laboratory environment prior to being placed into production. Consequently, all IP addressing within is drawn from internal private address space as defined by RFC 1918. Due to earlier troubles with their ISP, GIAC Enterprises has been reduced to one four-bit block of public address space (16 addresses). These addresses will be drawn from the 172.16.0.0/16 space for testing purposes. The remaining systems at GIAC Enterprises will be assigned private addresses from the 192.168.0.0/24 to 192.168.2.0/24 reserved space.

The partners of GIAC Enterprises will be assigned private addresses from the 192.168.3.0/24 reserved space for testing purposes. Other public (Internet) addresses necessary for testing will be drawn from the 172.17.0.0/16.

SECTION 1 - Security Architecture

GENERAL REQUIREMENTS

GIAC Enterprises has very limited funding for the redesign and restructuring of their network and network services. Consequently, GIAC Enterprises management has placed the following requirements, restrictions, and demands on Bandwidthco Security Services for this project:

- No additional hardware is to be purchased. Only hardware currently available on the premises is to be used.
- No additional software is to be purchased. Only freely available open source software, or software currently available on the premises is to be used.
- No additional routable (public) IP addresses, other than those currently available, are to be used. GIAC Enterprises currently has six routable IP addresses.

Based upon these requirements, the following general design and architectural decisions were made and will be employed:

- The GIAC Enterprises internal network will remain largely a Windows based operation, utilizing Windows 2000 Professional on user workstations and Windows 2000 Server on company servers. Microsoft Exchange Server 5.5 will service all incoming and outgoing e-mail. Microsoft IIS will host the local intranet and provide Web, FTP, and News services. Oracle 9i for Windows Release 1 will house all company records, employee data, the "Fortune" database, and customer credit card records.
- All servers positioned outside the internal network, whether accessible to the outside or not, will be Windows 2000 Server boxes converted to Linux. These machines will be fitted with SuSE Linux 7.2, Kernel version 2.4.4. Such reasoning will be discussed in detail throughout this report.
- Heavy reliance upon Network Address Translation (NAT) and port redirection will be necessary as only fringe devices can be assigned routable IP addresses.

BUSINESS OPERATIONS & ACCESS REQUIREMENTS

GIAC Enterprises must provide secure and reliable network and internet services to very diverse groups of users that include their customers, potential customers (browsers), suppliers, and partners as well as their own employees. The bulk of the GIAC Enterprises employees are resident at the main office, but a small mobile sales

force obtain their services from the road. The needs and access requirements of each of these groups will now be discussed individually. Table 1 below summarizes these requirements.

GIAC Enterprises Customers

GIAC Enterprises chief revenue source is generated from the sale of fortunes through publicly accessible web servers. As the bread and butter of the operation, this must be a primary concern. Rapid and visually appealing access for browsers, shoppers and potential customers is essential as is the need for quick and secure credit card transactions for those who purchase fortunes. GIAC Enterprises cannot afford any more bad publicity or expense related to the theft or exposure of customer and credit information. To achieve this, the following is recommended:

- Use of an HTTP reverse cache proxy. Requests to port TCP 80 will be redirected to port TCP 8080 or port 8000.

- Use Secure Sockets Layer (SSL) for transaction processing. No specified port access will be required. This will require opening TCP port 443.

- Maintain Fortune Store items on Oracle database on local network segment. As the connection from web server to database server is local, no special port access will be required.

- Maintain the customer personal and credit information on the internally protected network segment. This will require indirect access to TCP ports 1521 (TNS Listener) and 1575 (Oracle Names Server) from the public web server to the internal Oracle database server.

- Maintain external DNS server for name resolution. This will require opening UDP port 53.

GIAC Enterprises Suppliers

In its prime, GIAC Enterprises employed nearly 100 staff writers whose sole task was to churn out hundreds of new fortunes and clever quips on a daily basis. However, once in decline, GIAC Enterprises was forced to contract out this aspect of the operation. A respectable force of freelance writers now keeps the company supplied with their fresh merchandise. This being the case, a simple, but effective and secure, means of fortune submission, review, acceptance, and payment must be put into place. It is recommended that the same online transaction processing system used for customer sales be extended to accommodate the fortune suppliers needs.

This approach will have two distinct benefits. First, system maintenance costs will be reduced as additional systems will not be required. Additionally, additional outside access to services will not be required over and above of that required by the

customers. Ultimately, this means that additional holes will not need to be punched in the firewall. Here is how it will work:

- The writer will submit their newly written fortunes using an SSL tunnel. Again, requests to port TCP 80 will be redirected to port 8080 and the HTTP reverse proxy. This will also require opening TCP port 443.

- A transaction will incur wherein the writer receives a receipt for their submissions, while the submissions are sent to the Oracle database on the internal network.

- GIAC Enterprises QC personnel review the submissions, and the submission is either accepted or rejected.

- Acceptance or rejection statements are then sent to the writer by e-mail the following day.

- Checks are mailed to the writers for their services the first Monday of each month.

GIAC Enterprises Partners

Despite the hard times, GIAC Enterprises has held on to several of their key partners in the fortune business. Located at several sites throughout the world, the partners are key in that they offer an additional revenue stream for GIAC Enterprises. The partners function is to take fortune submissions accepted by GIAC QC personnel, translate them into various foreign languages, and resell them abroad. Due to the nature of the work and their relationship with GIAC Enterprises management, the partners require private access to several internal network resources, including the internal web server and Oracle database. Additionally, video conferencing services are needed. To meet these requirements, the following is recommended:

- Install an IPSec capable VPN gateway with a publicly routable IP address. To provide confidentiality, authentication, integrity, access control, and light protection against traffic flow analysis, the VPN gateway must be both AH (Authentication Header) and ESP (Encapsulating Security Payload) capable. Access to UDP port 500 will be necessary.

- Control access to internal resources with an IPTables/Netfilter filtering firewall on the VPN gateway based on source address and authentication. Internal destination ports required for appropriate resource access include TCP port 80 for web access, TCP port 1521 (Oracle TNS Listener), and TCP port 1575 (Oracle Names).

- Provide Microsoft NetMeeting for video conferencing. Bandwidthco Security Services has advised against this. However since the need exists, they have agreed to comply and secure the use of NetMeeting to the highest possible degree. TCP ports 1503 (T.120 Whiteboard) and 1720 (T.120 Call Set-up) will require opening.

Additionally TCP ports 389 (LDAP Internet Locator Service - ILS), 522 (User Location Service – ULS), and 1731 (Audio Set-up) will require access for NetMeeting to be effectively deployed and used.

- Standard and encrypted e-mail service.

GIAC Enterprises Road Warriors

The GIAC Enterprises mobile workforce (from here-on-out referred to as the Road Warriors) main function is to sell fortunes to businesses and restaurants around the country, thus providing another source of revenue. Access and service needs are identical to those of the partners group. However, Road Warriors will be granted access to the GIAC Enterprises network through the VPN gateway utilizing transport mode. Since the connecting ISP will dynamically assign IP addresses, adequate access control to internal resources will prove more difficult.

GIAC Enterprises Internal Employees

GIAC Enterprises employees located at the main office must be subdivided as either "general use" or "IT administrative" in terms of need. General use employees require the following:

- Unrestricted access to all network resources on the protected network segment. There are no port or firewall considerations for strict internal use.

- Unrestricted access to access to company intranet resources.

- DNS (UDP port 53) for host name resolution.

- Unrestricted internet access for out-going connections only.

IT Administrative requirements are considerably more demanding and include the following recommended methods:

- General Server maintenance will require remote access in additional to console access. Remote connections to all servers on all segments of the network will be accomplished by SSH tunneling of required protocols. Requires opening TCP port 22.

- Backup and recovery operations will require remote access in additional to console access. Remote connections to all servers on all segments of the network will be accomplished by SSH tunneling of required protocols. Requires opening TCP port 22. Amanda will be used for full and incremental backups across network segments. Requires opening ports 10080 to 10083.

TABLE 1 Summary of Needs and Access Requirements				
Group	**Service**	**Protocol**	**DPORT**	**SPORT**
Customers	HTTP Cache Proxy	TCP	8080	>1023
	Secure Sockets Layer	TCP	443	>1023
	Oracle TNS Listener (Indirect)	TCP	1521	>1023
	Oracle Names (Indirect)	TCP	1575	>1023
	DNS	UDP	53	>1023
Suppliers	HTTP Cache Proxy	TCP	8080	>1023
	Secure Sockets Layer	TCP	443	>1023
	TNS Listener (Indirect)	TCP	1521	>1023
	Oracle Names (Indirect)	TCP	1575	>1023
	Mail (SMTP)	TCP	25	>1023
Partners	IPSec	UDP	500	>1023
		AH	N/A	N/A
		ESP	N/A	N/A
	HTTP Cache Proxy	TCP	8080	>1023
	Internal Web Services	TCP	80	>1023
	Oracle TNS Listener	TCP	1521	>1023
	Oracle Names	TCP	1575	>1023
	T.120 Whiteboard	TCP	1503	>1023
	T.120 Call Set-up	TCP	1720	>1023
	NetMeeting Audio	TCP	1731	>1023
	NetMeeting ILS	TCP	389	>1023
	NetMeeting ULS	TCP	522	>1023
Road Warriors	Identical To Partners			
Internal Employees	HTTP	TCP	80	>1023
	HTTP Cache Proxy	TCP	8080	>1023
	FTP Proxy	TCP	8080	>1023
	Mail (SMTP)	TCP	25	>1023
	DNS	UDP	53	>1023
IT & Administrative	SSH (Secure Shell)	TCP	22	>1023
	Amanda	TCP	10080:3	>1023
	Telnet	TCP	23	>1023
	TFTP	UDP	69	>1023
		TCP	>1023	>1023
	Oracle TNS Listener	TCP	1521	>1023
	Oracle Names	UDP	1575	>1023
	DNS	TCP	53	>1023
			>1023	53
		UDP	53	>1023
			>1023	53
	NTP	UDP	123	>1023
			>1023	123
			123	123
	Syslog	UDP	>1023	514
			514	>1023

- Router maintenance will require remote access in additional to console access. Restricted telnet will be used for configuration management. Requires opening TCP port 23. TFTP will be used for file management and backup. Requires opening UDP port 69.

- Oracle database management will require opening TCP ports 1521 (TNS Listener) and 1575 (Oracle Names).

- Primary and secondary DNS servers will be located on separate network segments to prevent single point failure. DNS zone transfers will require opening UDP port 53 and TCP port 53.

- NTP time synchronization among network servers and other network devices will required opening UDP port 123.

- All logging services will be centralized. This will require opening UDP port 514.

- Only static routing will be used, thus eliminating the need to accommodate dynamic routing protocols.

- Use a sendmail relay, eliminating the need to open MDA ports (POP TCP 110 or IMAP TCP 143) across the firewall for mail access or retrieval.

BASIC NETWORK ARCHITECTURE

The basic network architecture for GIAC Enterprises as proposed by Bandwidthco Security Services is based on the following criteria and priorities:

1. Access needs and requirements of all network users.
2. Business function, operation, and profitability.
3. Data security and integrity.
4. Fault tolerance, redundancy, and stability.
5. Disaster recognition and recoverability.
6. Administrative functionality, capability, and extensibility.

Based on these criteria and priorities, Bandwidthco Security Services recommends the following:

- The GIAC Enterprises network will be segmented into four quadrants, extending from and centralized around a central firewall. The firewall will be the primary focal point for traffic and access control.

- A border router will be placed between the firewall and the GIAC Enterprises connection to the internet. The border router will primarily be used for traffic control

at the internet gateway, but will also provide screening and access control functions, and provide protection for the firewall itself.

- The network quadrants, implemented by the four-homed firewall security policy, will consist of an internal (protected) network segment, a service segment, an administrative segment, and a perimeter segment.

- The internal segment will provide support to the GIAC Enterprises business and financial operation with a series of six primary Windows 2000 advanced servers, and provide resource access to company employees through workstations loaded with Windows 2000 professional.

- The service segment will consist of six hardened and specialized bastion hosts, and a VPN gateway. The service network will primarily provide secure e-commerce capabilities to internet users and GIAC Enterprises customers. However, the service network will additionally provide secure transaction services for company suppliers and secure communications for partners and employees working distant from the central facility.

- The administrative segment will exclusively provide security, analytical, and reporting and notification support for the firewall and the other three network segments. Logging, backup and recovery operations, and network data analytical capabilities will all be centralized and segregated here.

- The function of the perimeter segment will be to provide a semi-secure and private buffer between the GIAC Enterprises network and the internet.

- The firewall box itself, as well as all machines on the service and administrative networks, will be equipped and configured to run SuSE Linux 7.2. These servers were all originally purchased as Windows 2000 boxes, but can easily and comfortably be converted to Linux use with virtually no capital outlay.

Figure 1 provides an overall architectural view of the network. Below, each network segment and its components are discussed in detail.

ARCHITECTURAL COMPONENTS

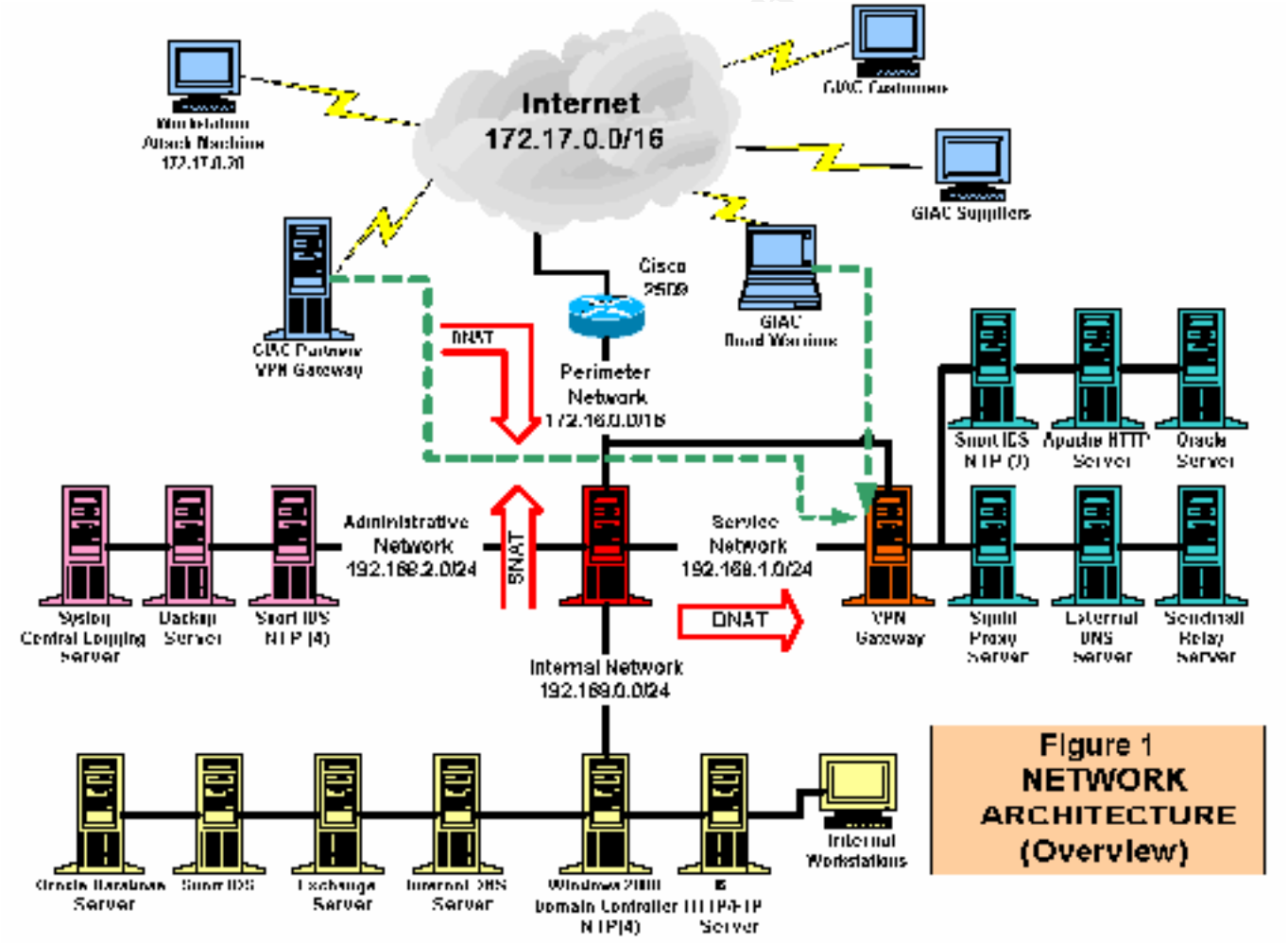
The Border Router

A Cisco 2509 access router has been selected to connect GIAC Enterprises to the outside world. This device has been selected only because it is the highest capacity router currently owned by GIAC Enterprises. Funding is not available to upgrade or replace the device so it will be integrated and utilized to maximize its strengths.

The Cisco 2500 series has reached the end of its production life and is being removed from Cisco's product line. However, they prove adequate for low to medium traffic environments. Equipped with a 20 MHz 68030 processor and 8 Megabytes of Flash Memory, the 2509 will only be configured to perform basic packet filtering and outgoing stateful monitoring. The objective is to achieve an approximate 20% load reduction on the central firewall.

The Firewall

The Linux OS firewall box will be a dual Pentium 4 system fitted with 1 GB of random access memory. The required four network interfaces will consist of matching Intel Pro 100 Server Adapters. SuSE Linux 7.2 was selected as the firewall operating system since it's native 2.4.4 kernel version contains inherent support for SMP, and for the integrated netfilter/iptables firewalling subsystem. Netfilter/iptables is, without question, the best non-commercial, low-cost, firewalling product available anywhere today. Netfilter/iptables delivers the functionality of both stateless and stateful packet filtering, several types NAT (Network Address Translation), and advanced packet processing in



the form of “packet mangling”.

Even more significant is netfilter/iptables exceptional built-in logging capability that directly interfaces with the system logging daemon. The LOG judgment target --log-prefix option, further enhances this. Certainly, this allows for easy parsing of log files in search of key words or statements. Additionally, this feature greatly simplifies the troubleshooting and analysis of filtering rule sets.

Finally, netfilter/iptables integrates well with several other modular open-source security products including Psionic Technologies TriSentry security suite. TriSentry is discussed in greater detail below.

All in all, the decision to use netfilter/iptables was one of the easier ones necessitated by the design and specification process. Many of our requirements are met in one packaged distribution.

The Service Network

The service network segment will contain the following seven key servers to support the GIAC Enterprises business operation:

- VPN Gateway
- Apache HTTP Server
- Squid Proxy Server
- External DNS Server
- Sendmail Relay Server
- Oracle 9i Server
- Snort IDS Sensor and NTP Server

Again, SuSE Linux 7.2 is the operating system of choice for all servers. The Apache, Squid, Sendmail, and Snort IDS servers will all be upgraded with the latest versions of these open-source software products, with no additional costs.

The VPN Gateway

Placement of the VPN Gateway was given careful thought during the design phase of the network structure. This proved to be one of the more difficult decisions. Five possible solutions were weighed:

- ***Place the VPN gateway outside the firewall***. Given the shortage of available public IP address and the related requirement to use NAT, this was a temping solution. However, this would create an additional single point of failure in the main traffic stream. Also, all traffic, whether destined to use the VPN or not, would be forced too pass through the gateway, creating both a latency and bandwidth bottleneck. Option eliminated.

- ***Place the VPN gateway behind the firewall***. Not a viable option. It is our intent to employ IPSec on the VPN gateway and SNAT or the outbound interface of the firewall. Any attempt to perform NAT operations on IPsec packets between the IPsec gateways creates a basic conflict. IPsec wants to authenticate packets and ensure they are unaltered on a gateway-to-gateway basis. Since NAT rewrites packet headers as they go by, IPsec authentication fails if packets are rewritten anywhere between the IPsec gateways. For AH, which authenticates parts of the packet header including source and destination IP addresses, this is fatal. If NAT changes those fields, AH authentication fails. For IKE and ESP, this is not necessarily fatal, but it presents other potential complications. Option eliminated.

- ***Place the VPN gateway on the firewall***. This could help prevent any conflicts between SNAT and AH. However, it would place a heavy load on the firewall. Our desire is to have the firewall perform only specialized filtering, inspection, and translation activities, and do them well. It is not our desire to have the firewall be a "Jack of all trades" and "a master of none." Option eliminated.

- ***Integrate the VPN gateway with the firewall***. This option allows immediate benefits in addition to permitting the gateway machine it to perform independently. The VPN gateway could necessarily be protected by the packet filtering capabilities of the firewall, and still benefit from its logging capabilities as well. However, this would require installing one or more additional network interfaces on the firewall, placing a great burden upon it. Option eliminated.

- ***Place the VPN gateway on the service network and integrate it with the firewall***. This option requires the VPN gateway to be a stand-alone device, and be "router capable". Incoming VPN traffic would be sent directly to the VPN device from the border router, and subsequently pass through the firewall to enter the protected network if necessary. Outgoing VPN traffic would necessarily pass first through the firewall, prior to being directed to the VPN gateway and exiting the border router. Access control and logging could be enabled on the VPN system itself to control and monitor incoming VPN traffic. This provides the best of all possible worlds, and it meets all the required needs. Option accepted.

The two required network interfaces on the VPN gateway machine will consist of matching Intel Pro 100 Server Adapters. The SuSE Linux 7.2 operating system will be installed on the VPN gateway for the identical reasons it was selected for the firewall machine. However, there are several additional considerations to be discussed below.

FreeS/WAN will be installed as the VPN software of choice. FreeS/WAN is a freely available, open source Linux implementation of the IPsec (IP security) protocols that supports both transport and tunnel mode of operation. Three protocols are used for IPsec:

- AH (Authentication Header) provides a packet-level authentication service.
- ESP (Encapsulating Security Payload) provides encryption plus authentication.
- IKE (Internet Key Exchange) negotiates connection parameters, including keys, for the other two protocols.

The FreeS/WAN implementation has three main parts:

- KLIPS (kernel IPsec) implements AH, ESP, and packet handling within the kernel.
- Pluto (an IKE daemon) implements IKE, negotiating connections with other systems.
- Various scripts provide an administrative interface to the machinery.

Additionally, FreeS/WAN supports what is called "opportunistic encryption". This means that any two FreeS/WAN gateways will be able to encrypt their traffic, even if the two gateway administrators have had no prior contact and neither system has any preset information about the other. Using this technology, both systems in the communication pick up the authentication information they need through programmed DNS. Once configured, everything is automatic and the gateways look for opportunities to encrypt, and encrypt whatever they can.

Most significantly, in using KLIPS and Pluto, FreeS/WAN can seamlessly integrate with netfilter/iptables and provide dynamic stateful filtering capabilities. Thus, inbound access can be controlled, and logging implemented here as well.

Administrative Network

The administrative network segment will contain the following three key servers to support the GIAC Enterprises IT administrative operation:

- Central Syslog Server
- Central Backup Server
- Snort IDS Sensor and NTP Server

Again, SuSE Linux 7.2 is the operating system of choice for all servers. System logging for the border router and for all servers on all network segments will be directed to the central Syslog server where automated and proprietary perl scripts will continually analyse the data for potential and real security abnormalities. Additionally, Swatch and Psionic TriSentry will be installed on the central logging server.

Internal Network

The internal network segment will contain the following six key servers to support the GIAC Enterprises IT business and creative development operation:

- Windows 2000 AD Domain Controller and NTP Server
- IIS HTTP Server
- Internal DNS Server
- Microsoft Exchange Server
- Oracle 9i Server
- Snort IDS Sensor

TABLE 2
Summary of Network Devices

Device	Operating System Support Software	Hardware Core	Network Interfaces	IP Address
Cisco 2509 Router	Cisco IOS 12.2	20 MHz 68030	eth0 eth1	172.16.0.10 172.17.0.10
Firewall	SuSE Linux 7.2 Kernel Version 2.4.4 Netfilter/iptables 1.2.6a	Dual P4 1 GB RAM	eth0 eth1 eth2 eth3	192.168.0.1 172.16.0.1 192.168.2.1 192.168.1.1
VPN Gateway	SuSE Linux 7.2 FreeS/WAN 1.95	Pentium 4 1024 RAM	eth0 eth1	172.16.0.2 192.168.1.1
Squid Proxy Server	SuSE Linux 7.2 Squid Proxy 2.4	Pentium 4 1024 RAM	eth0	192.168.1.3
Apache HTTP Server	SuSE Linux 7.2 Apache 2.0.39	Pentium 4 1024 RAM	eth0	192.168.1.4
External DNS Server	SuSE Linux 7.2 Bind 8.3.3	Pentium 4 1024 RAM	eth0	192.168.1.5
Sendmail Mail Relay	SuSE Linux 7.2 Sendmail 8.12.5	Pentium 4 1024 RAM	eth0	192.168.1.6
External Oracle Server	SuSE Linux 7.2 Oracle 9I 1.0.1	Pentium 4 1024 RAM	eth0	192.168.1.7
Logging Server	SuSE Linux 7.2 Syslogd Proprietary Perl Apps	Pentium 4 1024 RAM	eth0	192.168.2.2
Backup Server	SuSE Linux 7.2 Amanda 2.4.2 SSH	Pentium 4 1024 RAM	eth0	192.168.2.3
First Windows DC	Windows 2000 Server Active Directory	Pentium 4 1024 RAM	eth0	192.168.0.2
IIS HTTP/FTP Server	Windows 2000 Server IIS 6.0	Pentium 4 512 RAM	eth0	192.168.0.4
Internal DNS Server	Windows 2000 Server Microsoft DNS	Pentium 4 512 RAM	eth0	192.168.0.5
Exchange Server	Windows 2000 Server Exchange Server 5.5	Pentium 4 512 RAM	eth0	192.168.0.6
Internal Oracle Server	Windows 2000 Server Oracle 9I 1.0.1	Pentium 4 1024 RAM	eth0	192.168.0.7
Service IDS	SuSE Linux 7.2 Snort 1.8.7	Pentium 4 1024 RAM	eth0	192.168.1.8
Administrative IDS	SuSE Linux 7.2 Snort 1.8.7	Pentium 4 1024 RAM	eth0	192.168.2.8
Internal IDS	Windows 2000 Server Snort 1.8.7	Pentium 4 1024 RAM	eth0	192.168.0.8

Intrusion Detection Systems

A Snort 1.8.7 sensor will be placed on all network systems to provide network intrusion detection. TCP Wrappers and the Psionic TriSentry software suite will be installed on all Linux servers with alerting mechanisms activated.

IP ADDRESSING

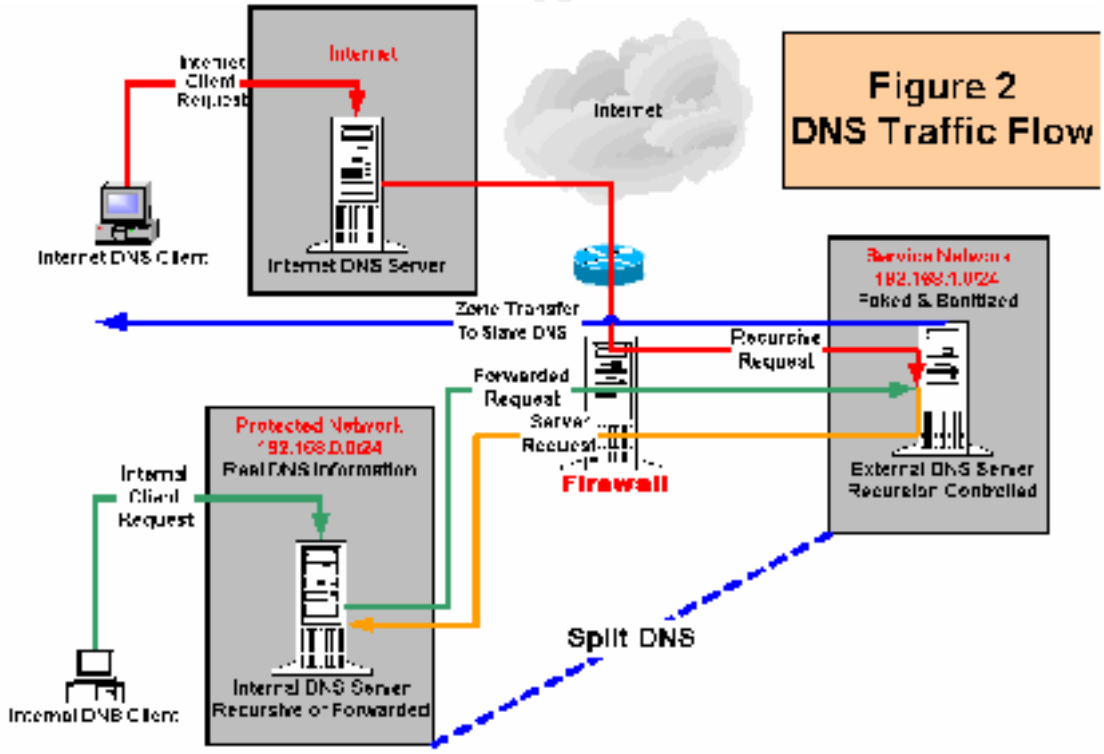
A proposed IP addressing scheme with system summary is detailed by Table 2 above.

PROTOCOLS & PROTOCOL REQUIREMENTS

Given the system user requirements and proposed network architecture, it may be prudent to review the affected network protocols and how they will traverse the network. This review will be critical to the development of effective security policy.

DNS Services

The primary problem with DNS information is availability and access, and to "who" and for "what". Since DNS information if often gathered and used as the basis for attack, GIAC Enterprises internal DNS information must remain private and hidden. At the same time, external hosts may require certain information to successfully locate and connect to the services available on the service network. A complex configuration utilizing "split DNS" will be required to achieve this. Figure 2 illustrates how this will work, and it is described below.



Two sets of DNS information will need to be maintained. Actual (real) DNS records for GIAC Enterprises will be maintained and published on the Internal DNS server on the internal network. Sanitized DNS records, containing only the essential records required to connect to hosts on the service network, will be maintained on the external primary (master) DNS server located on the service segment. The external DNS service will be published and available to the outside world to answer client requests. By correctly utilizing and configuring DNS "forwarding" and "recursion" properties, this split configuration should perform remarkably well.

External DNS client (resolver) requests requiring information concerning GIAC Enterprises from the Internet will be directed by DNAT through the firewall to the External DNS server. The request will be answered, and access will then be granted. Internal DNS clients will be configured to request DNS information from the internal DNS server only, or the server containing the real information regarding GIAC enterprises hosts. The internal DNS server will answer all requests about all hosts and services on the internal, service, and administrative segments. The Internal DNS server will not be able to respond to requests involving any information external to the network. When needed, however, the internal DNS server will then issue a recursive query to the external DNS server to obtain its information.

This presents an additional problem. Allowing a DNS server to be exposed to the Internet is generally not a good idea as it could potentially be used as a DNS recursive workhorse for the rest of the world. The solution will be to configure the external DNS server as a "forwarder" for the Internal DNS server. Secondly, the external DNS server must be configured to permit recursion only for hosts located on the internal network segment. This can be achieved by proper configuration of the /etc/named.conf file on the external DNS server with the following entry:

```
options {

   # restrict use of recursion to local queries
   allow-recursion { 192.168.0.0/24; };
}
```

Consequently, the external DNS server will now perform recursive requests for the internal DNS server and return the information to it.

A remaining configuration element involves the host servers on the service network acting as DNS clients by requesting DNS information for their own use. The solution here is to configure these machines to request their DNS information from the internal DNS server, which contains the actual DNS records for the network.

Finally, DNS zone transfers will be restricted. The firewall will be configured to allow zone transfers to occur only between the primary DNS server on the service network to

the secondary (slave) DNS server on the administrative network. Also, zone transfers will be restricted by proper entries in the primary DNS server's /etc/named.conf file as such:

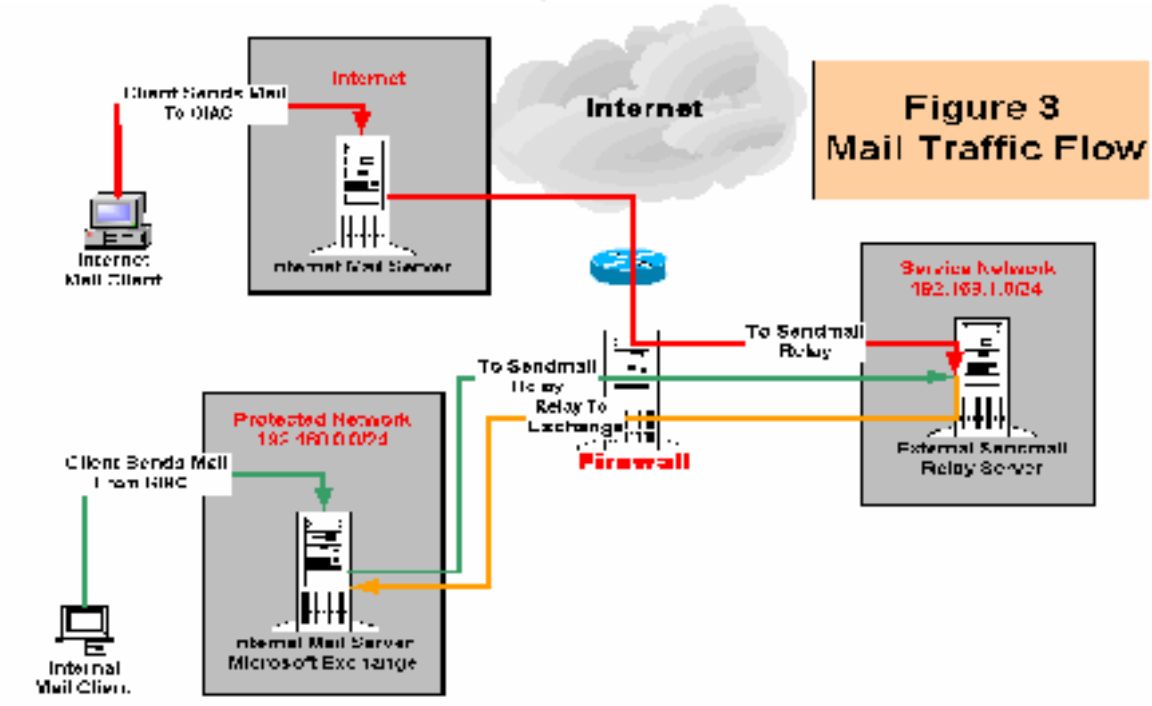
```
zone "giac.com" in {
    type master;
    file "giac.zone";
     allow-transfer { 192.168.2.2; };
};
```

The logging of zone transfers will also be enabled within the named.conf file with the following entry:

```
logging {
      channel bind_xfers {  //log zone transfers
            file "/var/log/bind_log";
            severity info;
      };

      category security { bind_xfers; };
};
```

GIAC Enterprises should have a secure, trustworthy, and effective name resolution system in place.



SMTP

The primary goal here is to protect the internal mail (Microsoft Exchange Server) server. Thus, we install a sendmail mail relay server on the service network segment to act as a buffer between the internal mail server and the outside world. DNAT will direct all TCP port 25 connections from the outside to the sendmail relay on the service network. Figure 3 demonstrates how this will work.

With this design, all SMTP communication from the Internet to GIAC Enterprises must first connect to the mail proxy server (sendmail relay) on the service network. The mail relay then picks up the connection and completes the delivery. Internal SMTP communications will operate in the reverse fashion. Communication and connection will be with the mail proxy, and the mail proxy will complete the delivery to the Internet. With this arrangement, an outside mail server is never permitted to talk directly with the internal mail server. Conversely, the internal mail server is never permitted to speak directly to a mail server on the outside. All requests must first go through the proxy.

Mail relaying is disabled by default in all versions of sendmail since version 8.9. Consequently, several configuration options will be enabled to allow the GIAC Enterprises mail relay to perform properly, and yet not provide a bouncing mechanism for spammers to use and abuse. For GIAC Enterprises, mail protection will be enabled primarily using the access.db and mailertable.db features. Consequently, the sendmail.cf file will be configured with the following options and features enabled in the linux.mc file:

```
MASQUERADE_AS(`giac.com')dnl
MASQUERADE_DOMAIN(`.giac.com')dnl
EXPOSED_USER(`root')dnl
FEATURE(`masquerade_entire_domain')dnl
FEATURE(`masquerade_envelope')dnl
FEATURE(`mailertable',`hash –o /etc/mail/mailertable.db')dnl
FEATURE(`access_db',`hash –o /etc/mail/access.db')dnl
```

Next, generate the new sendmail.cf file as follows:

```
m4 /etc/mail/linux.mc > /etc/sendmail.cf
```

The /etc/mail/access file will then be assigned the following entries to enable proper relaying:

```
# permit mail relaying for GIAC Enterprises
localhost.localdomain     RELAY
localhost                 RELAY
127.0.0.1                 RELAY
192.168                   RELAY
```

giac.com RELAY

The /etc/mail/mailertable will be assigned the following entry to enable forwarding of mail from the sendmail relay on the service network to the Microsoft Exchange server on the internal network:

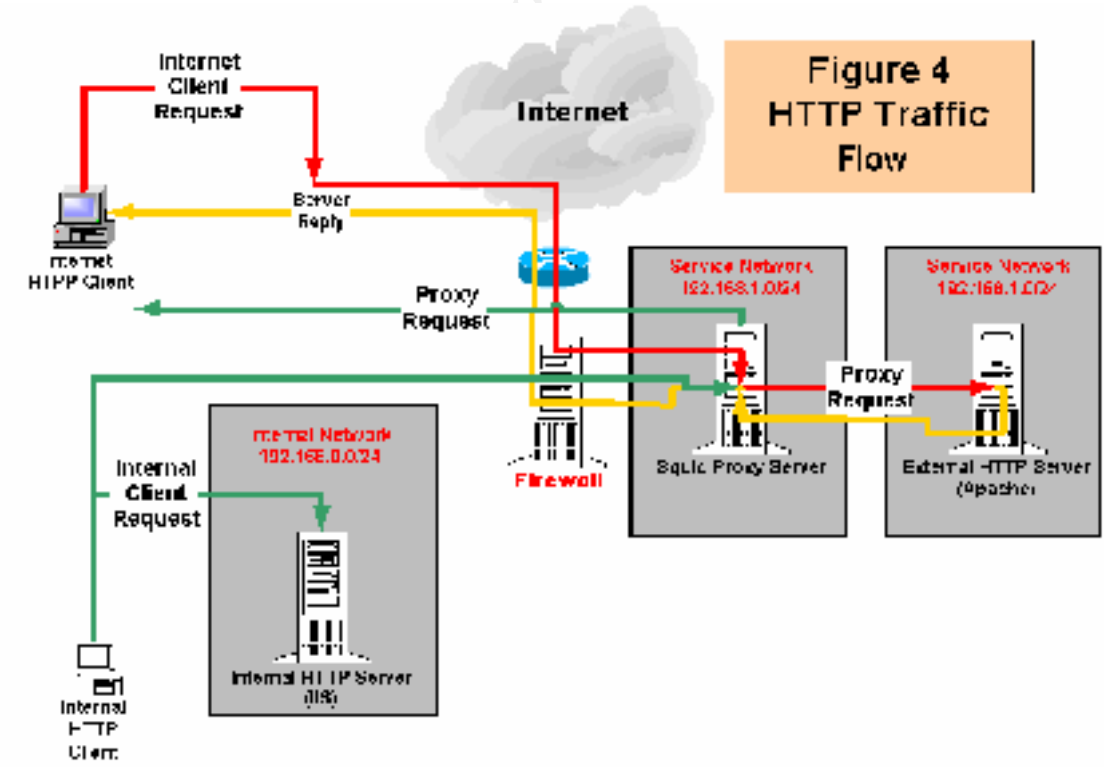
giac.com smtp:[192.168.0.6]

Finally, generate access and mailertable databases as follows:

makemap hash /etc/mail/access.db < /etc/mail/access
makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable

HTTP/FTP

Proxy using the freely available, open source Squid proxy-caching application will also handle HTTP communications. Squid is not only capable of managing outbound HTTP access, it can also act as a reverse proxy by managing inbound connections and requests. Squid will be configured as both a HTTP proxy server and HTTP reverse proxy server for GIAC Enterprises. Properly configured, Squid acts as an agent, accepting requests from either HTTP clients (such as browsers) or FTP clients, and passing them to the appropriate Internet or internal HTTP server. Figure 4 illustrates how this will work on the GIAC Enterprises network.



The following protocols will be supported by the proposed configuration:

- HTTP
- FTP
- SSL (Secure Socket Layer) for secure online transactions.

For the purposes of this operation, Squid will be configured to listen on port 8080, and on the default port of 3128, with the following entry in the squid.conf file:

http_port 8080 3128

External incoming client HTTP requests on port 80 will be redirected by DNAT on the firewall to port 8080 on the Squid Proxy server. The proxy server will then either accept or reject the connection. If accepted, the proxy will handle the client request and pass it to the Apache HTTP server on the service network. The proxy will handle all remain communications established by this connection.

Internal outgoing client request will be handled in a similar manner. However, the client will be instructed on how to locate the Squid server. This will be accomplished at GIAC Enterprises by using the Squid advanced configuration mode and configuring the rule server. Clients will then connect to this server on startup, and download information on which proxy server to connect to. Content control will also be implemented here if necessary.

IPsec/SSL/SSH

Figure 5 illustrates the traffic flow for all encrypted communications both into and out of the GIAC Enterprises network, and within the network itself. All online transaction processing for customer purchases and supplier submissions will utilize secure socket layer (SSL) tunnels. SSL traffic will pass through the firewall and be directed to the Squid proxy in the same fashion as unencrypted HTTP traffic. Again, the Squid proxy will establish a connection to the Apache server. The Apache server will in turn serve as a proxy for the Oracle database server for the transfer of sensitive data to and from the database.

Secure Shell (SSH) communications will pass through the internal interfaces of the firewall only, and be limited to only necessary IT administrative file transfer activities. The transfer of Oracle table space files between the service and internal network segments will only be performed in this manner. Likewise, backup copies transferred to the backup server on the administrative segment will be done using SSH tunnels. The shell script template below will be used for these backup types. The full script can be found in Appendix G.

```
#!/bin/bash
# backup server2 to server3

cd /
list="home boot etc var opt/oracle"

for file in $list
do
# keep the previous backup
tgz=`echo $file | sed -e 's/\//_/g'`;
old="${tgz}.old";

# Store the old backup
ssh server3 "mv -f /storage/backup/server2/$tgz /storage/backup/server2/$old";
sleep 5m

# perform current backup
tar -cz $file/* | ssh server3 "cat - > /storage/backup/server2/$tgz";
sleep 5m
done
```

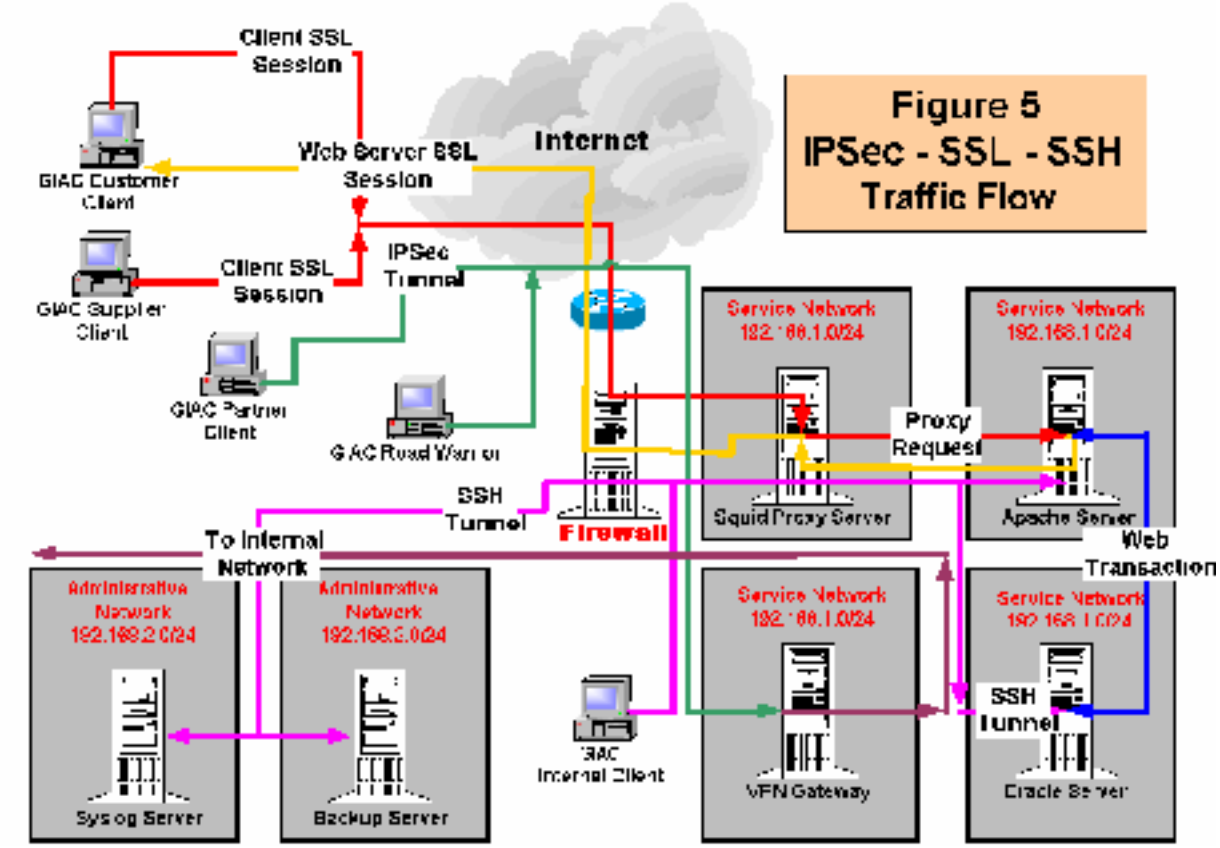
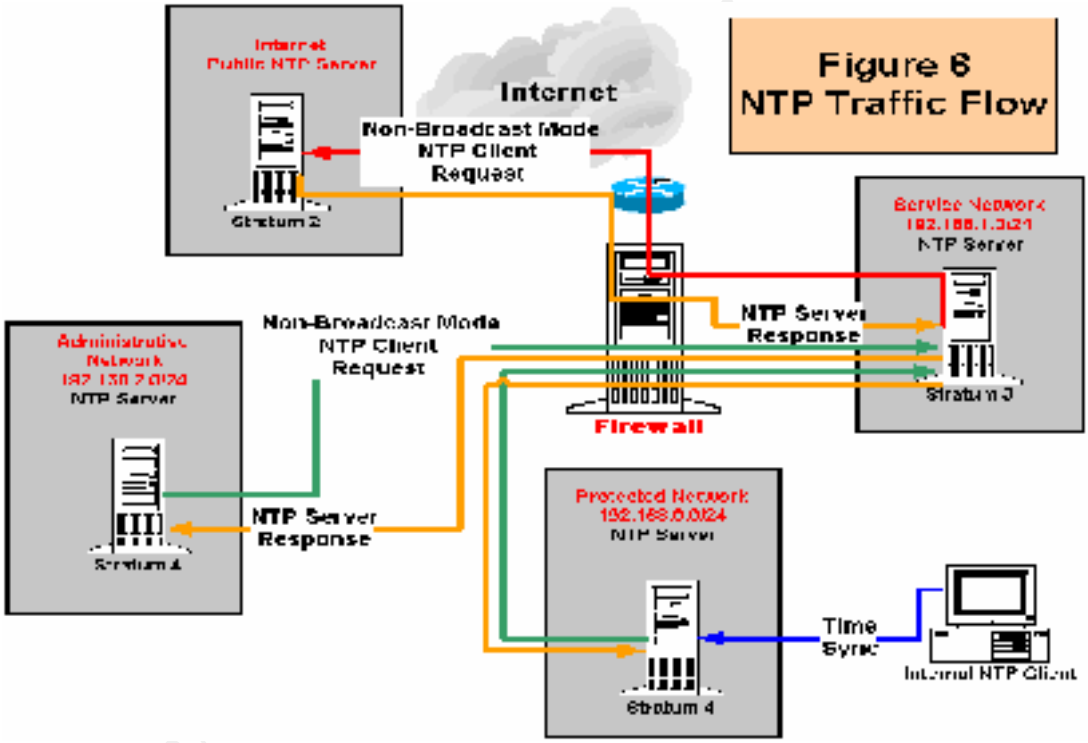


Figure 5
IPSec - SSL - SSH
Traffic Flow

As discussed earlier, incoming IPsec VPN traffic, either in tunnel mode or transport mode, will be directed by the border router directly to the VPN gateway machine. It will then pass through the firewall to the internal network based on policy. IPsec connections originating on the internal network will first pass through the firewall internal network interface, and in turn, be directed the VPN gateway and the internet. The GIAC Enterprises Road Warriors will be connecting to the VPN gateway using transport mode. All connections to and from partner's networks will utilize tunnel mode. The Security Policy Databases (SPD) and Security Association Databases (SAD) enabling IPsec connections will be discussed latter.

NTP

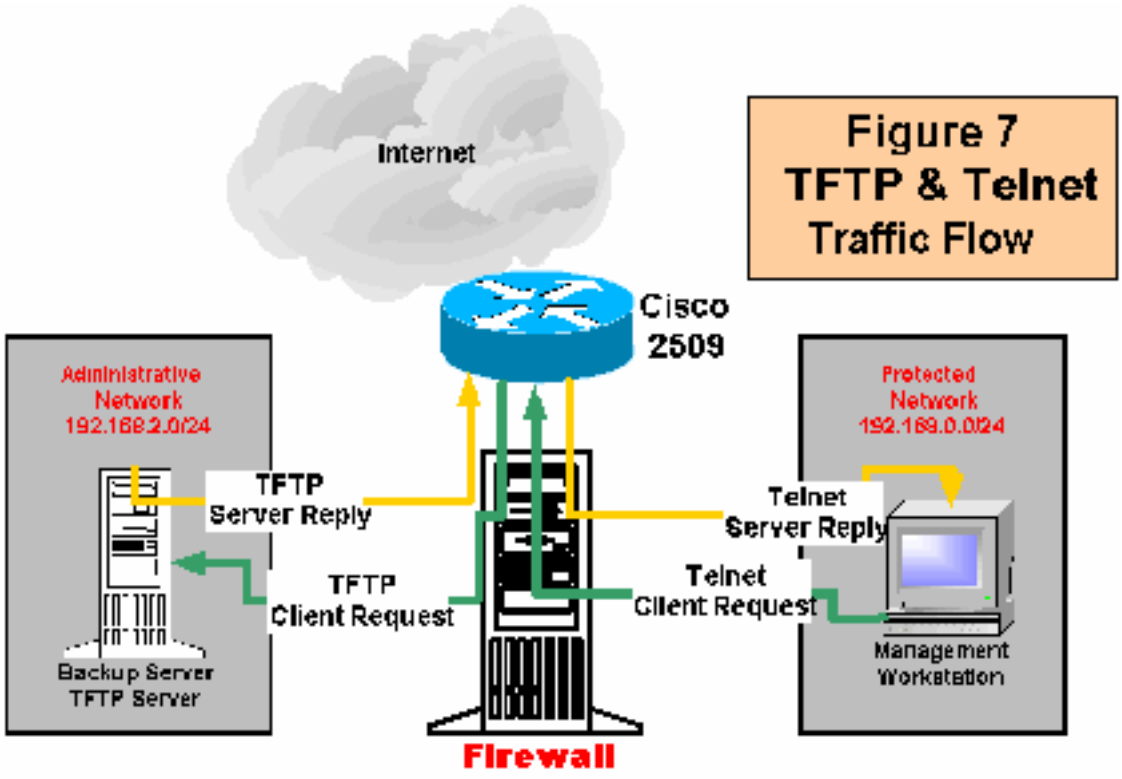
Precise time stamping is critical for secure network operations. GIAC Enterprises will utilize NTP (Network Time Protocol) to ensure accurate and consistent time keeping and time stamping on all network servers and workstations. NTP operations are illustrated by Figure 6.



The Snort IDS box on the service network segment will be configured to double as a Stratum 3 NTP server. To preserve network bandwidth, it will be configured to operate in "non-broadcast" mode and obtain time adjustments from a selected and approved Stratum 2 NTP server in the public domain.

Two Stratum 4 NTP servers, also operating in "non-broadcast" mode, will be operational on the network. One will be placed on the Snort IDS sensor on the administrative

segment. The second will be located on the Highest (First) Active Directory (AD) Domain Controller on the internal Windows network.



The process will work as follows:

- Client machines will synchronize time with the Stratum 4 NTP server on the internal network.
- Both Stratum 4 NTP servers will synchronize time with the Stratum 3 NTP server on the service network.
- The Stratum 3 NTP server on the service network will synchronize time with the Stratum 2 public NTP server.

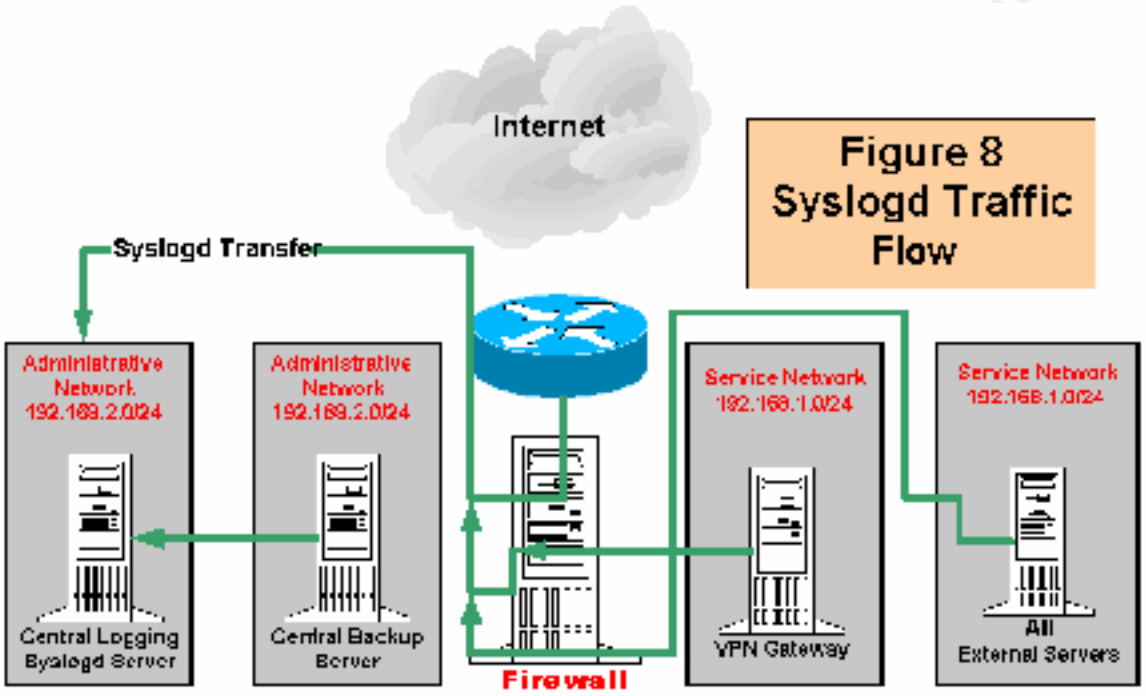
To prevent abuse of or attack on the NTP servers, access control lists will be configured in the /etc/ntp.conf file as follows:

```
# prohibit general access to this service
restrict default ignore

# permit systems on this network to synchronize with this time service
# do not permit those systems to modify the configuration of this service
# do not use those systems as peers for synchronization
```

```
restrict 192.168.0.0 mask 255.255.255.0 notrust nomodify notrap
restrict 192.168.1.0 mask 255.255.255.0 notrust nomodify notrap
restrict 192.168.2.0 mask 255.255.255.0 notrust nomodify notrap
restrict 172.16.0.0 mask 255.255.0.0

# permit access over loopback interface
restrict 127.0.0.1
```



Telnet & TFTP

The GIAC Enterprises border router will largely be administered and maintained remotely using Telnet and Trivial File Transfer Protocol (TFTP). Access to the router will be restricted to a single workstation on the internal network. Router configuration, script, and backup files will be stored and maintained from the central Backup server on the administrative network segment. Traversal of the firewall for both protocols will be necessary. Figure 7 illustrates how this will work.

Syslogd

Server, firewall, and router logging for the GIAC Enterprises will be centralized on the Syslog server on the administrative network. For this to occur on SuSE Linux, several configuration options must be properly set. First, the Syslog daemon on the Syslog server must be started using the **–r** flag or it will refuse to accept messages from other machines. Secondly, by default, Syslogd on SuSE Linux will refuse to forward logging messages to remote hosts on the network unless it is invoked with the **–h** flag. Therefore, the following commands must be added to the etc/init.d/syslog startup script:

 # add on central logging server
Syslogd –r

 # add on host machine to send to logging server
Syslogd -h

Once this has been done, the following must be added to the /etc/syslog.conf file to allow message transfers to occur:

***.* @192.168.2.2**

For the logging transfers to progress, the Syslogd protocol will be required to traverse the firewall. Figure 8 illustrates the logging information flow process.

SECTION 2 - Security Policy and Tutorial

RULE BASE DEVELOPMENT

This section of the security plan for GIAC Enterprises will present and accomplish the following:

- Present a general written security policy statement. This statement will provide the basis and foundation for applied device-specific policy.
- Present and explain the security configuration and security policy for the border router.
- Present and explain the security policy for the main firewall.
- Present and explain the security policy for the VPN gateway.
- Present and explain the security policy for host hardening.
- Provide a tutorial for specified policy development and provide a baseline for consistent and reasonable policy change management.

GIAC Enterprises rule base development will focus on applying the following guidelines:

- Adhere to and follow the requirements established by the written security policy.
- Assume that rules are read in sequential order and the first rule matched will be applied.
- Position the most frequently used rules first, and position specific rules before general rules.
- Identify and remove unnecessary or redundant rules .

GENERAL POLICY STATEMENT

The following written security policy statement will provide the foundation for applied rule base policy for GIAC Enterprises:

##
 # SECURITY POLICY FOR GIAC ENTERPRISES
##
We are ready to start building our rule base and therefore need to refer to our basic written security policy. All policy rules will be enforced, but enforcement will be distributed between the screening router and the main firewall. Some rules may be applied in both for redundancy
##
The general policy scheme for the network is as follows:

1. **Allow remote router & firewall management from single internal address (authorized administration).**
2. **Allow internal users to ping router and firewall interfaces (router & lockdown/connectivity).**
3. **Drop all remaining traffic addressed to firewall and router (router & firewall lockdown).**
4. **Deny unwanted protocols, services, and ports (noise reduction, attack prevention,).**
5. **Deny unwanted network addresses (attack and spoof prevention).**
6. **Deny loop back & private addresses (reliability).**
7. **Deny broadcasts (smurf & DOS attack prevention).**
8. **Allow established internal tcp connections to pass (performance).**
9. **Allow internally initiated udp sessions to pass (performance).**
10. **Allow internally initiated icmp queries, messages, and path MTU to pass (performance).**
11. **Allow centralized logging with and centralized backup (administration).**
12. **Allow inbound and outbound email and DNS services (performance).**
13. **Deny internal DNS records to internet (attack prevention).**
14. **Allow inbound access to web server (performance).**
15. **Allow secure transactions (attack prevention).**
16. **Allow encrypted communication (attack prevention).**
17. **Allow internal clients unrestricted access to internet (business policy).**
18. **Allow multicast (performance).**
19. **Screen hostile traffic (attack prevention).**
20. **Allow NTP synchronization between network devices over network segments (administration).**
21. **Deny everything else and log it (implicit deny)**

SCREENING (BORDER) ROUTER

Basic Security Configuration

Before applying any security policy, we must first secure and harden the router itself. The content and discussion that follows will be enabled by the Cisco IOS script router_security.conf. The full script can be found in Appendix A.

We begin by setting the enable and secret passwords, and the passwords for the console and auxiliary ports:

```
! start in configuration mode
config t
!
! set enable password
enable password xxxxxxxxx
! set enable secret password:
enable secret xxxxxxxxx
! set vty password
line vty 0 4
login
password xxxxxxxxx
! set console and auxiliary passwords
line con 0
login
password xxxxxxxxx
line aux 0
login
password xxxxxxxxx
exit
```

Next, we configure hostname and DNS capabilities:

```
! start in configuration mode
config t
!
hostname Router1
exit
ip domain-lookup
! set up a host table
config t
ip host Router1 172.16.0.10
ip host server2 192.168.0.1
ip host server1 192.168.0.2
ip host server3 192.168.1.2
ip host server4 192.168.2.2
!
```

```
! show host table
exit
show hosts
```

Now we configure IP addressing:

```
! start in configuration mode
config t
! eth0
int eth0
Ip address 172.16.0.10 255.255.0.0
no shut
description GIAC Enterprises Network Gateway
! eth1
int eth1
Ip address 172.17.0.10 255.255.0.0
no shut
description GIAC Internet Gateway
exit
```

Now we set message banners:

```
! start in configuration mode
config t
!
! set login banner
banner login #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
! set MOTD banner:
banner MOTD #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
! set incoming terminal line banner:
banner incoming #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
! set executive process creation banner:
banner exec #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
exit
```

Now, we harden the router by disabling all unnecessary services and protocols:

```
! start in configuration mode
config t
!
! force display passwords to be encrypted
service password encryption
! disable CDP (Cisco Discovery Protocol)
no cdp run
! disable finger service
no service finger
! disable echo, discard, chargen, and daytime services
no service udp-small-servers
no service tcp-small-servers
! drop source routing
no ip source-route
! disable servers
no ip bootp server
no ip http server
! disable SNMP
no snmp
!
! Add these to external interface of screening router
! run from interface config mode
int eth1
! don't send icmp for denied items in access-list
no ip unreachables
! prevent ICMP redirection
no ip redirects
! prevent layer 2 broadcast mapping and smurf amplification
no ip direct-broadcast
! disable proxy-arp
no ip proxy-arp
! disable NTP if not using external time source
    ! no enable ntp
exit
```

We are going to rely on static routing. It is time to set that up along with a default route:

```
! We want to avoid dynamic routing protocols if at all possible
! start in configuration mode
config t
!
! disable RIP
no router rip
! disable OSPF
no ospf
!
```

```
! set default route
ip route 0.0.0.0 0.0.0.0 172.16.0.1
ip classless
!
! set static routes
! Syntax: ip route route mask next hop
! Example: 172.16.0.0 255.255.0.0 192.168.0.1
ip route 192.168.0.0 255.255.255.0 172.16.0.1
ip route 192.168.1.0 255.255.255.0 172.16.0.1
ip route 192.168.2.0 255.255.255.0 172.16.0.1
```

Good start. But let's get serious now and set up our router logging. We are going to set up our router to log messages at the central logging facility on the administrative network. This is essentially a four-step process.

STEP 1 - Configure Loopback Interface

The Loopback interface (in this case) serves as the single network identity for the router. With dynamic routing enabled, its IP address is available through any active router interface. This availability makes it perfect to serve as the routers vty access interface and provide the source address for logging and SAA (secure system auditing and accounting) reporting and authentication requests. When using loopback addresses on public networks, the loopback prefixes should not be reachable from hosts outside of the networks administered routing domain to reduce security exposure. The use of RFC 1918 addresses is perfect for this function because they are not routable on the Internet.

```
! start in configuration mode
config t
interface loopback 0
ip address 192.168.30.1 255.255.255.255
exit
```

STEP 2 - Configure TIME Settings

Time, specifically timestamp, is a valuable piece of information when determining when a problem occurred. For timestamps to be of use, it is essential that all network devices derive time from a common network time source using the Network Time Protocol (NTP). To configure the router's time zone proprieties, the configuration commands <clock timezone {TZ} {gmt offset}> and <clock summer-time {DST TZ} recurring> are used. The configuration commands <ntp server {ip addr}> and <ntp source {interface}> define the NTP server(s). The router should synchronize with the source IP address of the NTP server.

```
! start in configuration mode
config t
!
! set clock for Pacific Time Zone (offset 8 hours from GMT) and PDST
```

```
clock timezone PST -8
clock summer-time zone recurring
! set NTP server
ntp update-calendar
ntp server 192.168.1.2
ntp source loopback 0
!
! Now set the system clock:
! Example: #clock set 21:51:20 7 April 2002
! configure the timestamp options for system logging and debug messages
service timestamps log datetime localtime
service timestamps debug datetime localtime
exit
! check NTP associations
sh ntp associations
```

STEP 3 - Configure Local (router) System Message Reporting

First, optionally disable console and buffered logging since it normally is not needed. Leaving console logging enabled sends messages to the console port, and, during an event that generates a large amount of logging messages, it clutters the command line, making command entry difficult. Buffer logging stores the messages in a buffer in the router's DRAM. This is an acceptable option if trap logging is not feasible. The more practical method for viewing logging messages locally is to enable monitor logging using the configuration command <logging monitor {syslog facility}>, then use the exec command <terminal monitor> to view messages as necessary.

```
! start in configuration mode
config t
no logging console
no logging buffered
! Now enable monitor logging
logging monitor debugging
```

STEP 4 - Configure Remote (server) System Message Reporting

Configuring trap logging is itself a four step process. First, define a syslog host using the configuration command <logging {hostname or IP address}>. Next, define the logging severity of the messages to be sent using the configuration command <logging trap {severity}>. Now, define the IP address that will be associated as the origin address of the logging messages. This is set using the configuration command <logging source-interface {interface type m/s/p}>. Finally, define the syslog "facility" that the messages are sent to on the remote syslog server. Use the configuration command <logging facility {syslog facility}>.

```
! start in configuration mode
config t
!
```

```
! logging facility local1
! logging source-interface Loopback 0
! Log everything to syslog
! no logging console
no logging buffered
logging 192.168.2.2
logging trap debug
logging monitor debugging
logging console emergencies
```

We are nearly complete with the security configuration. Just a few performance related issues remain. We will want to leave ICMP MTU discovery active as well as the ability to filter on TCP flag bits:

```
! start in configuration mode
config t
!
ip tcp path-mtu-discovery
ip tcp selective-ack
exit
```

Now that we have finished, lets save and backup the configuration:

```
! start in exec mode
sh run
show start
copy run start
! copy start tftp
```

Secure Access To Router

It is essential that access to the router be tightly controlled. We will restrict access to the router's virtual terminal and auxiliary ports by applying two standard access control lists with the script access-list vty.txt. The full script can be found in Appendix B.

```
! add access-lists:   access-list 1 & access-list 2
!
! allow only specific hosts to telnet into router protected network
access-list 1 permit 192.168.0.0 0.0.0.255
! use if SNAT working on firewall
access-list 1 permit 172.16.0.0 0.15.255.255
line vty 0 4
access-class 1 in
login
!
! block all access to aux port
access-list 2 deny 0.0.0.0 255.255.255.255
line aux 0
```

```
access-class 2 in
login
transport input all
exit
```

Apply Security Policy

As mentioned earlier, we do not want to over-burden the 2509 router. Consequently, we will only attempt to accomplish the following security policy requirements at the router:

- Screen-out some of the more offensive ports and protocols that we do not need or want.
- Provide some basic anti-spoofing measures through ingress and egress filtering.
- Control ICMP requests and messages.
- Permit outside access to public services.
- Permit DNS and mail server communications.
- Permit unrestricted access to the internet from the internal network segment through use of reflexive access list.

To accomplish this, we will apply three named extended ACL's. Two of them will be a matched pair to allow support for reflective inspection. These will be applied both "in" and "out" on the inbound interface of the router. The third ACL will be applied outbound "out". The full text of the three scripts can be found in Appendix C.

Let's begin by looking at policy for the external interface eth1 inbound. We create ACL **ip access-list extended inbound-eth1**.

First, we need to account for SNAT addressing. Returning packets addressed to the external interfaces of the firewall and VPN gateway must be permitted:

```
! permit packets addresses to external firewall and VPN gateway ports
permit ip any host 172.16.0.1 log
permit ip any host 172.16.0.2 log
```

Next, we'll start placing basic control over what we let in and what we want to filter out:

```
! deny rfc 1918 addresses
! deny ip 192.168.0.0 0.0.255.255 any log
! comment out for testing purposes
! deny ip 172.16.0.0 0.15.255.255 any log
```

```
deny ip 10.0.0.0 0.255.255.255 any log
!
! Deny packets with localhost, broadcast and multicast addresses
deny ip 127.0.0.0 0.255.255.255 any log
deny ip 255.0.0.0 0.255.255.255 any log
deny ip 224.0.0.0 7.255.255.255 any log
!
! deny packets without ip addresses
! deny ip host 0.0.0.0 any log
```

Now we set up anti-spoofing measures with ingress and egress filtering. Here, the edge or border router internal interface should only accept traffic with source addresses belonging to the internal network. The router external interface should only accept traffic with source addresses other than the internal network. Our concern is only with public (routable) addresses as any private addresses have already been dropped earlier. In our case we are only concerned with the external firewall interface, the external VPN Gateway interface, and both router ports:

```
deny ip host 172.16.0.1 any log
deny ip host 172.16.0.2 any log
!
! router interface ip addresses
!
deny ip host 172.16.0.10 any log
deny ip host 172.17.0.10 any log
```

Now, let's filter out unwanted protocols:

```
! deny NetBIOS proprietary protocols from coming in
! we don't particularly want to log these
deny tcp any any range 135 139
deny udp any any range 135 139
deny tcp any any eq 445 log
!
! deny X-Windows Traffic
!
deny tcp any any range 6000 6255 log
!
! deny TFTP, Syslog, and SNMP Traffic
!
deny udp any any eq 69 log
deny udp any any eq 514 log
deny udp any any range 161 162 log
!
! deny NFS & Sun RPC
!
deny tcp any any eq 111 log
```

```
deny udp any any eq 111 log
deny tcp any any eq 2049 log
deny udp any any eq 2049 log
```

That done, we will now set up our reflexive (stateful) calls for connections established from the internal network. Remember, reflexive lists must have a matched in-out pair on each interface:

```
! Reflexive Protocol Control
!
evaluate tcp-filter
evaluate udp-filter

! allow host on internet to talk to mail relay
permit tcp any host 172.16.0.1 eq 25 reflect smtp-filter
! allow host on internet to talk to external DNS server
permit udp any host 172.16.0.1 eq 53 reflect dns-filter
permit tcp any host 172.16.0.1 eq 53 reflect dns-filter
```

Let's make sure incoming mail and DNS queries are allowed to pass. We also must not forget we will DNAT all traffic at the external firewall interface and we must account for that:

```
! DNS Traffic Static
!
! DNS traffic DNAT forwarded from firewall
permit tcp any host 172.16.0.1 eq 53 log
permit udp any host 172.16.0.1 eq 53 log
!
! SMTP Traffic Static
!
! SMTP traffic DNAT forwarded from firewall
permit tcp any host 172.16.0.1 eq 25 log
```

We must permit our Stratum 3 NTP server to communicate to it's public NTP server:

```
! NTP Traffic Static
!
! NTP traffic DNAT forwarded from firewall
permit udp any host 172.16.0.1 eq 123 log
```

Now, let's provide some ICMP guidance:

```
! ICMP Control Static
!
! Allow only specific ICMP
! Again:  because we are using SNAT only firewall and VPN gateway
```

```
! external port IP addresses are of concern for all host behind the router
!
! net-unreachable
permit icmp any host 172.16.0.1 3 0
permit icmp any host 172.16.0.2 3 0
! host-unreachable
permit icmp any host 172.16.0.1 3 1
permit icmp any host 172.16.0.2 3 1
! port-unreachable
permit icmp any host 172.16.0.1 3 3
permit icmp any host 172.16.0.2 3 3
! packet-too-big (fragment)
permit icmp any host 172.16.0.1 3 4
permit icmp any host 172.16.0.2 3 4
! administratively-prohibited
permit icmp any host 172.16.0.1 3 13
permit icmp any host 172.16.0.2 3 13
! source-quench
permit icmp any host 172.16.0.1 4
permit icmp any host 172.16.0.2 4
! ttl-exceeded
permit icmp any host 172.16.0.1 10
permit icmp any host 172.16.0.2 10
```

Finally, let's deal with the fall-through default policy:

```
! Default
! Permit and Log everything that does not meet the above rules.
!
permit ip any any log
!
! HITS CISCO IOS IMPLICIT DENY
```

We will now need to apply our matched out-going reflective ACL for the internal connections to work properly. This will be the policy for the external interface eth1 inbound. For this, we create ACL **ip access-list extended outbound-eth1**. This should be quite simple to accomplish:

```
ip access-list extended outbound-eth1
!
! Reflexive Protocol Control
! reflexive list must have a matched in-out pair on each interface
! allow unrestricted access to internet from internal network
!
permit tcp host 172.16.0.1 any reflect tcp-filter
permit udp host 172.16.0.1 any reflect udp-filter
evaluate smtp-filter
```

evaluate dns-filter

Now set the default policy:

```
! Default
!
! Permit and Log everything that does not meet the above rules.
!
permit ip any any log
!
! HITS CISCO IOS IMPLICIT DENY
```

Our last task is to screen outbound traffic for the internal interface eth0 outbound. For this, we create ACL **ip access-list extended outbound-eth0**.

```
!
ip access-list extended outbound-eth0
!
```

The first duty will be to secure and lockdown the router from the inside. Traffic to the interface will only be permitted from the firewall external interface and the VPN gateway external interface:

```
! Router Lockdown & Management
!
permit ip host 172.168.0.1 host 172.16.0.10 log
permit ip host 172.168.0.1 host 172.16.0.10 log
! use following if SNAT not up
permit ip host 192.168.0.3 host 172.16.0.10 log
```

We will permit all users and hosts on the internal network to ping the firewall for connectivity checking:

```
! Connectivity Checking
!
! allow internal users to ping both sides of router
permit icmp host 172.16.0.1 host 172.16.0.10 8
permit icmp host 172.16.0.1 host 172.17.0.10 8
! use following if SNAT not up
permit ip 192.168.0.0 0.0.0.255 host 172.16.0.10
permit ip 192.168.0.0 0.0.0.255 host 172.16.0.10
```

That completed, we will now place restrictions on the type of traffic let out of the network::

```
! Deny rfc 1918 addresses
!
deny ip 192.168.0.0 0.0.255.255 any log
! comment out next line for testing purposes
! deny ip 172.16.0.0 0.15.255.255 any log
deny ip 10.0.0.0 0.255.255.255 any log

! Deny packets with localhost, broadcast and multicast addresses
!
deny ip 127.0.0.0 0.255.255.255 any log
deny ip 255.0.0.0 0.255.255.255 any log
deny ip 224.0.0.0 7.255.255.255 any log

! Deny packets without ip address
!
deny ip host 0.0.0.0 any log
```

To prevent mischief from GIAC Enterprises own employees, we will perform egress filtering. The edge router internal interface should only accept traffic with source addresses belonging to the internal network. Concern is only with public (routable) addresses as any private addresses have already been dropped earlier. In our case we are only concerned with the external firewall interface and the external VPN Gateway interface.

```
! Prevent spoofing
!
permit ip host 172.16.0.1 any log
permit ip host 172.16.0.2 any log
! use the following if SNAT not up
permit ip 192.168.0.0 0.0.0.255 any log
permit ip 192.168.0.0 0.0.0.255 any log
```

Now we will block offensive protocols from leaving the network:

```
! Deny NetBIOS protocols from going out
! we don't particularly want to log these
deny tcp any any range 135 139
deny udp any any range 135 139
deny tcp any any eq 445 log
!
! Deny X-Windows Traffic
!
deny tcp any any range 6000 6255 log
!
! Deny TFTP, Syslog, and SNMP Traffic
!
deny udp any any eq 69 log
```

```
deny udp any any eq 514 log
deny udp any any range 161 162 log
!
! Deny NFS & Sun RPC
!
deny tcp any any eq 111 log
deny udp any any eq 111 log
deny tcp any any eq 2049 log
deny udp any any eq 2049 log
```

Lastly, we set up the policy defaults:

```
! Default
!
! Deny and Log everything that does not meet the above rules.
! Use log-input to find violating machines
!
deny ip any any log-input
! HITS CISCO IOS IMPLICIT DENY
```

To complete the router ACL configuration, we apply the list to the appropriate interfaces:

```
! start in configuration mode
config t
int eth1
ip access-group inbound-eth1 in
ip access-group outbound-eth1 out
int eth0
ip access-group outbound-eth0 out
```

THE FIREWALL

GIAC Enterprises main firewall is charged with being the workhorse for the network and will therefore be implementing all the remaining policy actions. As mentioned earlier, netfilter/iptables has been selected for the firewalling technology.

This report will first discuss the operational aspects of netfilter/iptables, and then peruse a relatively detailed explanation of policy development and implementation. The script that we develop below can be found in its entirety in Appendix D.

In order to get started using netfilter, the kernel must be compiled for netfilter support. Most distributions include this support by default. If all the modules have been built and installed, all modules will auto-install when a rule is entered. If not, these can be loaded either manually or as part of the iptables startup script as such:

```
insmod ip_tables
insmod iptable_nat
```

insmod ip_nat_ftp (state tracking)
insmod ip_conntrack (state tracking)
insmod ip_conntrack_ftp (state tracking)

The command line syntax for iptables is as follows:

iptables [-t table] -ACDIR CHAIN rule-specification -j TARGET [target option]

where:

- **iptables** is the command itself.
- **-t table** is the table specification of filter, nat, or mangle.
- **CHAIN** is the chain name.
- **rule specification** is the part of the command to match.
- **TARGET** is a valid target of DROP, ACCEPT, user-chain, LOG, REJECT, RETURN, or QUEUE.
- **target option**

Netfilter has three tables consisting of “filter”, “nat” and “mangle”. If no table is specified, the "filter" table is assumed and is therefore the default. Each table has certain chains available to it. User-created chains can belong to one and only one table. You will see later that some built-in chains belong to more than one table, but this is only true for built-in chains. User created chains cannot be mixed with chains from the other tables.

The filter table is the basic packet-filter table and has the following built-in chains:

- INPUT = packet destination is firewall
- FORWARD = packet passing through firewall
- OUTPUT = packet originating from firewall

Rules added to user-created chains created in the filter table can only contain targets valid in the INPUT, FORWARD or OUTPUT chains. Packets traversing the filter table will pass through one and only one of INPUT, FORWARD or OUTPUT. The INPUT chain will only be traversed if the packet's destination is the local system. The FORWARD chain will only be traversed if the packet is passing through the local system and bound for another system. Only packets originating on the local system with an external destination traverse the OUTPUT chain. Any given packet will traverse only one chain.

The “nat” table performs network address translation. Built-in chains for nat are as follows:

- PREROUTING
- POSTROUTING
- OUTPUT

Each chain permits a particular target:

- PREROUTING = DNAT target
- POSTROUTING = SNAT target
- OUTPUT = SNAT target

The “mangle” table is used to change (mangle) information other than the IP address in the header. It can be used to mark the packets, change type of service (TOS) or change time-to-live (ttl) information.

The rule-specification portion of each iptables command is the heart of the command. By properly crafting your rule, you can specify exactly what packets will be subject to a given rule. These selection criteria can be general, or as specific as necessary. In most cases, more specific criteria should come before general criteria.

Netfilter has four built-in targets:

- ACCEPT
- DROP
- QUEUE
- RETURN

The DROP target replaces ipchain's DENY target. All other targets used are based on modules that load as a target. These include REJECT, LOG, MARK, MASQUERADE, MIRROR, REDIRECT and TCPMSS. Terminal targets, such as ACCEPT, DROP, REJECT, MASQUERADE, MIRROR and REDIRECT, terminate a chain. A LOG target cannot terminate a chain, nor can it ACCEPT, REJECT, or DROP a packet. When LOG is the target, the chain continues to be traversed. If a terminal target is not hit, the rest of the chain will be traversed until it hits the default policy rule.

The policy rule is the overall rule for the chain. If the FORWARD chain contains a policy of DROP, and no rules above match in the chain, the packet will terminate when it hits the policy rule. The policy rule can only be one of the built-in targets. REJECT cannot be used as a policy rule.

Netfilter/iptables is also capable of stateful inspection and filtering. The “state” extension invokes the connection-tracking analysis of the “ip_conntrack” module.

Specifying “-m state” allows an additional “—state” option, which is a comma-separated list of states to match (the `!' flag indicates not to match those states). These states are:

- NEW = A packet which creates a new connection. NEW applies to packets with only the SYN bit set (and the ACK bit unset).

- ESTABLISHED = A packet which belongs to an existing connection. ESTABLISHED permits traffic to continue where it has seen traffic before in both directions. ESTABLISHED not only applies to TCP connections but to UDP traffic, such as DNS queries and trace routes as well as ICMP pings.

- RELATED = A packet which is related to, but not part of, an existing connection. RELATED includes packets such as an ICMP error, or a packet establishing an ftp data connection. RELATED applies to active FTP, which opens a related connection on port 20, but also applies to ICMP traffic related to the TCP connection.

- INVALID = A packet which could not be identified for some reason. This includes ICMP errors that don't correspond to any known connection. INVALID applies to packets that have invalid sets of options, as in an XMAS tree scan.

There are several different options and possibilities for using netfilter/iptables. Start with three built-in chains INPUT, OUTPUT and FORWARD. The following operations can be used to manage whole chains:

- Create a new chain (-N)
- Delete an empty chain (-X)
- Change the policy for a built-in chain. (-P)
- List the rules in a chain (-L)
- Flush the rules out of a chain (-F)
- Zero the packet and byte counters on all rules in a chain (-Z)

The following operations can be used to manipulate rules inside a chain:

- Append a new rule to a chain (-A)
- Insert a new rule at some position in a chain (-I)
- Replace a rule at some position in a chain (-R)
- Delete a rule at some position in a chain (-D)
- Delete the first rule that matches in a chain (-D)

Let’s look at a very useful rule application as an example of how we might troubleshoot the completed rule base:

execute iptables -t <table> -L -nv

The above rule, with the -v included, will display the number of packets and bytes that are affected by a given rule. If there are rules with 0 bytes affected by that rule, it should be reviewed for position in the chain. However, just because a rule has affected packets doesn't mean it's in the right place. It may have only affected half the packets that really should have been affected.

Iptables and the defined rule base is normally started and loaded into kernel memory space at boot time. The most effective way to accomplish this is through a shell script placed in the /etc/init.d directory and linked to the appropriate run level. So, let's begin our script. Comments included within the script will provide much of the explanation. As with any good shell script, we start by declaring useful variables.

```
#!/bin/sh
#####################################################################
# iptables.rc Script: Script Start
#####################################################################
# declare a trusted path
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/usr/local/bin
export PATH

# modify the IPT (iptables executable) variable to point to iptables binary
IPT=/usr/sbin/iptables

# set script variable name
SCRIPT_NAME="iptables.rc"

# set variable for ip_forwarding
FWD="/proc/sys/net/ipv4/ip_forward"

# set variable for controlling number of state table entries
STATE_CONNECTIONS="/proc/sys/net/ipv4/ip_conntrack_max"

# set variable for tcp connection timeout
TIMEOUT="/proc/sys/net/ipv4/tcp_fin_timeout"

# set variable for tcp keep alive interval
KEEP_ALIVE="/proc/sys/net/ipv4/tcp_keepalive_intvl"

# set variable for route verification ( built-in egress & ingress filtering)
# drop spoofed packets which cannot be replied to on the received
# interface based on routing tables
# * = all interfaces
RP_FILTER="/proc/sys/net/ipv4/conf/*/rp_filter"

# set variable for ICMP broadcasts
ICMP_ECHO_BROADCASTS ="/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts"

# set variable for source routing
# * = all interfaces
```

```
ACCEPT_SOURCE_ROUTE="/proc/sys/net/ipv4/conf/*/accept_source_route"

# set variable for accepting ICMP redirects
# * = all interfaces
REDIRECTS="/proc/sys/net/ipv4/conf/*/accept_redirects"

# set variable for sending ICMP redirects
# * = all interfaces
MAKE_REDIRECTS="/proc/sys/net/ipv4/conf/*/send_redirects"

# set variables for each firewall interface IP address
FIRE_ETH0="192.168.0.1"
FIRE_ETH1="172.16.0.1"
FIRE_ETH2="192.168.2.1"
FIRE_ETH3="192.168.1.1"

# set variables for each router interface IP address
ROUTE_ETH0="172.16.0.10"
ROUTE_ETH1="172.17.0.10"

# set variables for each vpn gateway interface IP address
VPN_ETH0="172.16.0.2"
VPN_ETH1="192.16.1.2"

# set variable for internal network
INT_NET="192.168.0.0/24"

# set variable for service network
SERVICE_NET="192.168.1.0/24"

# set variable for administrative network
ADMIN_NET="192.168.2.0/24"

# set variable for partners network
PART_NET="192.168.100.0/24"
```

We begin execution by sending a message to the syslog daemon for time-stamped documentation, and then load any needed modules:

```
#####################################################################
# Begin Execution
#####################################################################
# display firewall startup message
if [ -x /usr/bin/logger ];
then
    logger -p warning "Activating firewall script $SCRIPT_NAME "
    echo "Activating firewall script $SCRIPT_NAME"
fi

# some modules we'll want to install if not compiled into kernel
# modprobe ip_tables
# modprobe ip_nat_ftp
```

```
# modprobe ip_conntrack_ftp
```

A matter of good policy is to flush and delete all table chains and rules. We do so as follows:

```
for i in filter nat mangle
do
    echo "Flushing Filter Chains . . . . . $i table"
    $IPT -t $i -F
    $IPT -t $i -X
done
```

Next, let's set any critical kernel parameters to fit the policy that is being implemented. It is generally wise to turn ip_forwarding off while the rule base is being loaded into memory:

```
# disable ip forwarding
echo 0 > $FWD

# set tcp connection timeout
# echo "30" > $TIMEOUT

# set tcp keep alive
# echo "1800" > $KEEP_ALIVE

# set up route verification ( built-in egress & ingress filtering)
# apply to all interfaces on machine

for f in $RP_FILTER;
do
    echo 1 > $f
done

# drop packets that are source routed
for f in $ACCEPT_SOURCE_ROUTE;
do
    echo 0 > $f
done

# drop ICMP redirect packets
for f in $REDIRECTS;
do
    echo 0 > $f
done

# deny sending of ICMP redirect packets
for f in /proc/sys/net/ipv4/conf/*/send_redirects;
do
    echo 0 > $f
done
```

A significant part of our security structure is to avoid using any routing protocols. This would be an appropriate place to assign and activate our static routes.

```
#####################################################################
# Set Default and Static Routes
#####################################################################
echo "Setting default and static routes for server2 . . . . . "
logger -p warning "Setting default and static routes for server2 . . . . . "

# set default route for server2
route add default gw 172.16.0.10

# set static routes for server2 (firewall)
route add -net 172.16.0.0 netmask 255.255.0.0 dev eth1
route add -net 172.17.0.0 netmask 255.255.0.0 dev eth1
route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth3
route add -net 192.168.2.0 netmask 255.255.255.0 dev eth2
```

Now we will define our default policy for the built-in table chains. A default deny policy will be placed the "filter" table, while the "mangle" and "nat" tables will be assigned a default accept policy.

```
#####################################################################
# Define Default Policy
#####################################################################
# default policy for firewall will be deny all unless explicitly permitted.
echo "Setting default policies . . . . . filter table"
$IPT -t filter -P OUTPUT DROP
$IPT -t filter -P INPUT DROP
$IPT -t filter -P FORWARD DROP

echo "Setting default policies . . . . . nat table"
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT

echo "Setting default policies . . . . . mangle table"
$IPT -t mangle -P PREROUTING ACCEPT
$IPT -t mangle -P OUTPUT ACCEPT
```

As discussed earlier, the security architecture for GIAC Enterprises will make heavy use of both DNAT and SNAT. We will define those rules first:

```
#####################################################################
# Create NAT Rules
#####################################################################

# since SNAT is POSTROUTING we must append the POSTROUTING chain
$IPT -t nat -A POSTROUTING -o eth1 -j LOG --log-prefix "SNAT OUT ETH1  "
```

```
$IPT -t nat -A POSTROUTING -o eth1 -j SNAT --to 172.16.0.1

# send all incoming web traffic to Squid Proxy (HTTP & SHTTP)
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth1 -j LOG --log-prefix "DNAT IN 8080  "
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth1 -j DNAT --to 192.168.1.2:8080
$IPT -t nat -A PREROUTING -p tcp --destination-port 443 -i eth1 -j DNAT --to 192.168.1.2:8080

# send all incoming DNS traffic to External DNS server
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p udp --destination-port 53 -i eth1 -j LOG --log-prefix "DNAT IN 53  "
$IPT -t nat -A PREROUTING -p udp --destination-port 53 -i eth1 -j DNAT --to 192.168.1.2:53
$IPT -t nat -A PREROUTING -p tcp --destination-port 53 -i eth1 -j DNAT --to 192.168.1.2:53

# send all incoming mail traffic to sendmail relay
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 25 -i eth1 -j LOG --log-prefix "DNAT IN 25  "
$IPT -t nat -A PREROUTING -p tcp --destination-port 25 -i eth1 -j DNAT --to 192.168.1.2:25

# send all outgoing web traffic to Squid Proxy (HTTP & SHTTP)
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth0 -j LOG --log-prefix "DNAT OUT 8080
"
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth0 -j DNAT --to 192.168.1.2:8080
$IPT -t nat -A PREROUTING -p tcp --destination-port 443 -i eth0 -j DNAT --to 192.168.1.2:8080

# send all outgoing FTP traffic to Squid Proxy
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 21:22 -i eth0 -j LOG --log-prefix "DNAT FTP  "
$IPT -t nat -A PREROUTING -p udp --destination-port 21:22 -i eth0 -j DNAT --to 192.168.1.2:8080
```

Now will be an excellent place to introduce “user-defined” chains. We are going to create several of them for different purposed, but mainly we are looking at using the computer resources efficiently by minimizing the number of that must be loaded into memory and scanned for packet analysis. What we are going to do is create a chain for all of our main protocols, and then apply the rules only to those specific protocols. We also will create a specialized rules for attack detection and other tasks.

```
############################################################
# Create User Defined Chains
############################################################
echo "Creating User Defined Chains . . . . . "
# TCP rules
$IPT -t filter -N tcprules

# UDP rules
$IPT -t filter -N udprules

# ICMP rules
$IPT -t filter -N icmprules
```

```
# AH rules
$IPT -t filter -N ahrules

# ESP rules
$IPT -t filter -N esprules

# OSPF rules
$IPT -t filter -N ospfrules

# IGRP rules
$IPT -t filter -N igrprules

# GRE rules
$IPT -t filter -N grerules

# RULE_X is a stealth port scan detector
$IPT -t filter -N RULE_X

# RULE_Y is a trojan detector
$IPT -t filter -N RULE_Y

# RULE_Z is a spoofing detector
$IPT -t filter -N RULE_Z
```

With our chains created, we mow apply the rule base to them. First, let's define our rules for TCP:

```
####################
# TCP RULES INBOUND
####################
# here we INSERT at top of chain and shove down
echo "Creating TCP Rules - Inbound . . . . . "
# block IRC
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6667 -j DROP

# block NetBIOS
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 445 -j DROP

# block X-Windows Traffic
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6000:6200 -j DROP

# Block NFS & Sun RPC
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 111 -j LOG --log-prefix "SUN RPC ATTEMPT
"
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 111 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 2049 -j LOG --log-prefix "NFS ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 2049 -j DROP
```

```
# permit Ident to mail relay
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 113 -j LOG --log-prefix "IDENT CALLED  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 113 -j ACCEPT

# FIREWALL LOCKDOWN
# permit tcp connections from router and VPN gateway
# block everything else
$IPT -t filter -I tcprules -i eth1 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK TCP TO FIREWALL
72  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp -s 172.16.0.10 -j LOG --log-prefix "TCP FROM ROUTER  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp -s 172.16.0.10 -j ACCEPT
#$IPT -t filter -I tcprules -i eth1 -p tcp -s 172.16.0.2 -j LOG --log-prefix "TCP FROM VPN  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp -s 172.16.0.2 -j ACCEPT
```

Note that we are using both the –I and –A flags for our rules. The –I flag inserts the rule at the top of the chain, allowing it to be the first applied. Insertion can also be used by telling iptables which number the rule should apply at in the chain. The –A flag simply appends the rule at the end of the chain. Up until now we have been inserting. Now, let's append some rules:

```
# APPEND
# accept new inbound connections to Squid Proxy
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 8080 -m state --state NEW -j LOG --log-
prefix "SQUID-NEW  "
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 8080 -m state --state NEW -j ACCEPT

# permit inbound mail to mail relay
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 25 -j LOG --log-prefix "INCOMING MAIL  "
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 25 -j ACCEPT

# accept all remaining inbound traffic established internally
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A tcprules -i eth3 -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A tcprules -i eth2 -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT

# process and drop all other inbound TCP connections
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state NEW -j LOG --log-prefix "BLOCK TCP
ATTEMPT  "
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state NEW -j DROP
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state INVALID -j LOG --log-prefix "INVALID TCP
ATTEMPT "
# $IPT -t filter -A tcprules -i eth1 -p tcp -m state --state INVALID -j RULE_X
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state INVALID -j DROP

# PLACE THIS LAST (last resort)
# $IPT -t filter -A tcprules -i eth1 -j REJECT --reject-with icmp-host-unreachable
```

```
####################
# TCP RULES OUTBOUND
####################
echo "Creating TCP Rules - Outbound . . . . . "
# INSERT
# block IRC
$IPT -t filter -I tcprules -i eth0 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp --destination-port 6667 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 6667 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6667 -j DROP

# block NetBIOS
$IPT -t filter -I tcprules -i eth0 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules 2 -i eth0 -p tcp --destination-port 445 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 445 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 445 -j DROP

# block X-Windows Traffic
$IPT -t filter -I tcprules -i eth0 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp --destination-port 6000:6200 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 6000:6200 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6000:6200 -j DROP

# FIREWALL AND ROUTER LOCKDOWN
# permit administrative connections from 192.168.0.3 only
# block everything else
$IPT -t filter -I tcprules -i eth0 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK HOME TO FIREWALL
"
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth0 -p tcp -d 172.16.0.10 -j LOG --log-prefix "BLOCK HOME TO ROUTER
"
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -d 172.16.0.10 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK SERVICE TO
FIREWALL "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp -d 172.16.0.10 -j LOG --log-prefix "BLOCK SERVICE TO
ROUTER "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp -d 172.16.0.10 -j DROP
$IPT -t filter -I tcprules -i eth2 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK ADMIN TO
FIREWALL "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth2 -p tcp -d 172.16.0.10 -j LOG --log-prefix "BLOCK ADMIN TO ROUTER
$IPT -t filter -I tcprules 2 -i eth2 -p tcp -d 172.16.0.10 -j DROP
# permit
```

```
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 192.168.0.1 -j LOG --log-prefix "HOME CONNECT FIREWALL 01 "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 192.168.0.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 192.168.1.1 -j LOG --log-prefix "HOME CONNECT FIREWALL 11 "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 192.168.1.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 192.168.2.1 -j LOG --log-prefix "HOME CONNECT FIREWALL 21"
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 192.168.2.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.1 -j LOG --log-prefix "HOME CONNECT FIREWALL 72"
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.10 -j LOG --log-prefix "HOME CONNECT TO ROUTER "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.10 -j ACCEPT

# permit all ssh connections from 192.168.0.3 only
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j LOG --log-prefix "SSH FROM HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j ACCEPT

# permit all telnet connections from 192.168.0.3 only
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 23 -j LOG --log-prefix "TELNET FROM HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 23 -j ACCEPT

# permit and log service network & administrative network access to mail
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 25 -j LOG --log-prefix "TO MAIL RELAY "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 25 -j ACCEPT
$IPT -t filter -I tcprules -i eth2 -p tcp --destination-port 25 -j LOG --log-prefix "TO MAIL RELAY "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp --destination-port 25 -j ACCEPT

# permit DNS zone transfer from primary to secondary external DNS
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 53 -j LOG --log-prefix "ZONE TRANSFER "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 53 -j ACCEPT
$IPT -t filter -I tcprules -i eth2 -p tcp --destination-port 53 -j LOG --log-prefix "ZONE TRANSFER "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp --destination-port 53 -j ACCEPT

# permit oracle service between internal and service networks
$IPT -t filter -I tcprules -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1521 -j LOG --log-prefix "ORACLE CONNECTION "
$IPT -t filter -I tcprules 2 -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1521 -j ACCEPT
$IPT -t filter -I tcprules -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1575 -j LOG --log-prefix "ORACLE CONNECTION  "
$IPT -t filter -I tcprules 2 -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1575 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1521 -j LOG --log-prefix "ORACLE CONNECTION  "
$IPT -t filter -I tcprules 2 -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1521 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1575 -j LOG --log-prefix "ORACLE CONNECTION  "
$IPT -t filter -I tcprules 2 -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1575 -j ACCEPT

# permit and log amanda backups
```

```
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 10080:10083 -j LOG --log-prefix "AMANDA BACKUP OPS  "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 10080:10083 -j ACCEPT
$IPT -t filter -I tcprules -i eth2 -p tcp --destination-port 10080:10083 -j LOG --log-prefix "AMANDA BACKUP OPS  "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp --destination-port 10080:10083 -j ACCEPT

# APPEND
# permit unrestricted access to the internet from the internal network
#$IPT -t filter -A tcprules -i eth0 -p tcp -m state --state NEW -j LOG --log-prefix "STATE TCP ACCEPT  "
$IPT -t filter -A tcprules -i eth0 -p tcp -m state --state NEW -j ACCEPT
$IPT -t filter -A tcprules -i eth0 -p tcp -j LOG --log-prefix "NO STATE TCP ACCEPT  "
$IPT -t filter -A tcprules -i eth0 -p tcp -j ACCEPT

# deny all other tcp connections from remaining devices on internal network
$IPT -t filter -A tcprules -i eth0 -p tcp -s 192.168.0.0/24 -j LOG --log-prefix "TCP DENY FROM HOME "
$IPT -t filter -A tcprules -i eth0 -p tcp -s 192.168.0.0/24 -j DROP

# deny all other tcp connections from remaining devices on service network
$IPT -t filter -A tcprules -i eth3 -p tcp -s 192.168.1.0/24 -j LOG --log-prefix "TCP DENY FROM SERVICE  "
$IPT -t filter -A tcprules -i eth3 -p tcp -s 192.168.1.0/24 -j DROP

# deny all other tcp connections from remaining devices on admin network
$IPT -t filter -A tcprules -i eth2 -p tcp -s 192.168.2.0/24 -j LOG --log-prefix "TCP DENY FROM ADMIN "
$IPT -t filter -A tcprules -i eth2 -p tcp -s 192.168.2.0/24 -j DROP
```

We'll now define the rule bases for the remaining protocols:

```
####################
# UDP RULES INBOUND
####################
echo "Creating UDP Rules - Inbound . . . . . "
# here we INSERT at top of chain and shove down
# block NetBIOS
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 35:39 -j DROP

# Block NFS & Sun RPC
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 111 -j LOG --log-prefix "SUN RPC ATTEMPT "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 111 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 2049 -j LOG --log-prefix "NFS ATTEMPT  "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 2049 -j DROP

# block remaining TFTP, Syslog, and SNMP Traffic
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 69 -j LOG --log-prefix "TFTP ATTEMPT "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 69 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 514 -j LOG --log-prefix "SYSLOG ATTEMPT "
```

```
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 514 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 161:162 -j LOG --log-prefix "SNMP ATTEMPT "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 161:162 -j DROP

# FIREWALL LOCKDOWN
# permit tcp connections from router and VPN gateway
# block everything else
$IPT -t filter -I udprules -i eth1 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK UDP TO FIREWALL 72 "
$IPT -t filter -I udprules 2 -i eth1 -p udp -d 172.16.0.1 -j DROP
# permit syslog and tftp from router and VPN
$IPT -t filter -I udprules -i eth1 -p udp -s 172.16.0.10 --destination-port 514 -j LOG --log-prefix "SYSLOG FROM ROUTER "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 172.16.0.10 --destination-port 514 -j ACCEPT
# from router loop back logging facility
$IPT -t filter -I udprules -i eth1 -p udp -s 192.168.3.1 --destination-port 514 -j LOG --log-prefix "SYSLOG FROM ROUTER "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 192.168.3.1 --destination-port 514 -j ACCEPT

$IPT -t filter -I udprules -i eth1 -p udp -s 172.16.0.10 --destination-port 69 -j LOG --log-prefix "TFTP FROM ROUTER "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 172.16.0.10 --destination-port 69 -j ACCEPT
#$IPT -t filter -I udprules -i eth1 -p udp -s 172.16.0.2 --destination-port 514 -j LOG --log-prefix "SYSLOG FROM VPN "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 172.16.0.2 --destination-port 514 -j ACCEPT

# permit DNS inbound to external DNS server
#$IPT -t filter -I udprules -i eth1 -p udp --destination-port 53 -j LOG --log-prefix "EXTERNAL DNS CALL "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 53 -j ACCEPT

# permit communication with external NTP stratum 2 server
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 1233 -j LOG --log-prefix "INTERNET NTP CALL "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 1233 -j ACCEPT

# APPEND
# accept connections established internally
$IPT -t filter -A udprules -i eth1 -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A udprules -i eth3 -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A udprules -i eth2 -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

# process and drop all other UDP connections
#$IPT -t filter -A udprules -i eth1 -p udp -m state --state NEW -j LOG --log-prefix "BLOCK EXTERNAL UDP "
$IPT -t filter -A udprules -i eth1 -p udp -m state --state NEW -j DROP
$IPT -t filter -A udprules -i eth1 -p udp -m state --state INVALID -j LOG --log-prefix "BLOCK INVALID UDP "
$IPT -t filter -A udprules -i eth1 -p udp -m state --state INVALID -j DROP
```

```
####################
# UDP RULES OUTBOUND
####################
echo "Creating UDP Rules - Outbound . . . . . "
# here we INSERT at top of chain and shove down
# block NetBIOS
$IPT -t filter -I udprules -i eth0 -p udp --destination-port 35:39 -j DROP
$IPT -t filter -I udprules -i eth3 -p udp --destination-port 35:39 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 35:39 -j DROP

# firewall and router lockdown
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.3 -d 192.168.0.1 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.3 -d 192.168.0.1 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.2 -d 192.168.0.1 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.2 -d 192.168.0.1 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.3 -d 172.16.0.10 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.3 -d 172.16.0.10 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.2 -d 172.16.0.10 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.2 -d 172.16.0.10 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK HOME TO
FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -d 172.16.0.1 -j DROP
$IPT -t filter -I udprules -i eth0 -p udp -d 172.16.0.10 -j LOG --log-prefix "BLOCK HOME TO
ROUTER "
$IPT -t filter -I udprules 2 -i eth0 -p udp -d 172.16.0.10 -j DROP
$IPT -t filter -I udprules -i eth3 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK SERVICE TO
FIREWALL "
$IPT -t filter -I udprules 2 -i eth3 -p udp -d 172.16.0.1 -j DROP
$IPT -t filter -I udprules -i eth3 -p udp -d 172.16.0.10 -j LOG --log-prefix "BLOCK SERVICE TO
ROUTER "
$IPT -t filter -I udprules 2 -i eth3 -p udp -d 172.16.0.10 -j DROP
$IPT -t filter -I udprules -i eth2 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK ADMIN TO
FIREWALL "
$IPT -t filter -I udprules 2 -i eth2 -p udp -d 172.16.0.1 -j DROP
$IPT -t filter -I udprules -i eth2 -p udp -d 172.16.0.10 -j LOG --log-prefix "BLOCK ADMIN TO
ROUTER "
$IPT -t filter -I udprules 2 -i eth2 -p udp -d 172.16.0.10 -j DROP

# permit and log network access to DNS
#$IPT -t filter -I udprules -i eth3 -p udp --destination-port 53 -j LOG --log-prefix "SERVICE DNS
CALL  "
$IPT -t filter -I udprules 2 -i eth3 -p udp --destination-port 53 -j ACCEPT
#$IPT -t filter -I udprules -i eth2 -p udp --destination-port 53 -j LOG --log-prefix "ADMIN DNS CALL
"
$IPT -t filter -I udprules 2 -i eth2 -p udp --destination-port 53 -j ACCEPT
#$IPT -t filter -I udprules -i eth0 -p udp --destination-port 53 -j LOG --log-prefix "INTERNAL DNS
CALL  "
$IPT -t filter -I udprules 2 -i eth0 -p udp --destination-port 53 -j ACCEPT
```

```
# permit and log all internal network access to logging
$IPT -t filter -I udprules -i eth3 -p udp -d 192.168.2.2 --destination-port 514 -j LOG --log-prefix "SYSLOG OPERATION  "
$IPT -t filter -I udprules 2 -i eth3 -p udp -d 192.168.2.2 --destination-port 514 -j ACCEPT
$IPT -t filter -I udprules -i eth2 -p udp -d 192.168.2.2 --destination-port 514 -j LOG --log-prefix "SYSLOG OPERATION  "
$IPT -t filter -I udprules 2 -i eth2 -p udp -d 192.168.2.2 --destination-port 514 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -d 192.168.2.2 --destination-port 514 -j LOG --log-prefix "SYSLOG OPERATION  "
$IPT -t filter -I udprules 2 -i eth0 -p udp -d 192.168.2.2 --destination-port 514 -j ACCEPT

# permit and log all internal NTP activity
$IPT -t filter -I udprules -i eth3 -p udp --destination-port 123 -j LOG --log-prefix "NTP OPERATION  "
$IPT -t filter -I udprules 2 -i eth3 -p udp --destination-port 123 -j ACCEPT
$IPT -t filter -I udprules -i eth2 -p udp --destination-port 123 -j LOG --log-prefix "NTP OPERATION  "
"……remaining lines removed for the sake of brevity……
……

# deny all remaining udp connections from remaining devices on service network
#$IPT -t filter -A udprules -i eth3 -p udp -s 192.168.1.0/24 -j LOG --log-prefix "UDP DENY FROM SERVIE  "
$IPT -t filter -A udprules  -i eth3 -p udp -s 192.168.1.0/24 -j DROP

# deny all remaining udp connections from remaining devices on administrative network
#$IPT -t filter -A udprules -i eth2 -p udp -s 192.168.0.0/24 -j LOG --log-prefix "UDP DENY FROM ADMIN  "
$IPT -t filter -A udprules  -i eth2 -p udp -s 192.168.0.0/24 -j DROP

####################
# ICMP RULES INBOUND
####################
echo "Creating ICMP Rules - Inbound . . . . . "
# here we INSERT at top of chain and shove down
# do not allow firewall to be pinged from internet
iptables -t filter -I icmprules -i eth1 -p icmp --icmp-type echo-request -j DROP

# APPEND
# permit icmp
$IPT -t filter -A icmprules -i eth1 -p icmp -j ACCEPT

####################
# ICMP RULES OUTBOUND
####################
echo "Creating ICMP Rules - Outbound . . . . . "
# APPEND
# permit icmp
$IPT -t filter -A icmprules -i eth0 -p icmp -j ACCEPT
$IPT -t filter -A icmprules -i eth2 -p icmp -j ACCEPT
$IPT -t filter -A icmprules -i eth3 -p icmp -j ACCEPT
```

All the IPsec protocols should be going through the VPN gateway and not the firewall, so we will want to drop all of them by default. We will want to keep all routing protocols out completely.

```
######################
# AH RULES: PROTOCOL 51
######################
echo "Creating AH Rules . . . . . . . . . . . . "
# deny all AH inbound
$IPT -t filter -A ahrules -i eth1 -p 51 -j LOG --log-prefix "AH DENIED  "
$IPT -t filter -A ahrules -i eth1 -p 51 -j DROP

######################
# ESP RULES: PROTOCOL 50
######################
echo "Creating ESP Rules . . . . . . . . . . . . "
# deny all ESP inbound
$IPT -t filter -A esprules -i eth1 -p 50 -j LOG --log-prefix "ESP DENIED  "
$IPT -t filter -A esprules -i eth1 -p 50 -j DROP

######################
# OSPF RULES: PROTOCOL 89
######################
echo "Creating OSPF Rules . . . . . . . . . . . . "
# deny all OSPF inbound
$IPT -t filter -A ospfrules -i eth1 -p 89 -j LOG --log-prefix "OSPF DENIED  "
$IPT -t filter -A ospfrules -i eth1 -p 89 -j DROP

####################
# IGRP RULES: PROTOCOL 9
####################
echo "Creating IGRP Rules . . . . . . . . . . . . "
# deny all IGRP inbound
$IPT -t filter -A igrprules -i eth1 -p 9 -j LOG --log-prefix "IGRP DENIED  "
$IPT -t filter -A igrprules -i eth1 -p 9 -j DROP

####################
# GRE RULES: PROTOCOL 47
####################
echo "Creating GRE Rules . . . . . . . . . . . . "
# deny all IGRP inbound
$IPT -t filter -A grerules -i eth1 -p 47 -j LOG --log-prefix "GRE DENIED  "
$IPT -t filter -A grerules -i eth1 -p 47 -j DROP
```

The OUTPUT rules will control what the firewall can do itself, and what protocols it can use itself.

```
####################
# OUTPUT RULES
####################
echo "Creating OUTPUT Rules  . . . . . . . . . . . . "
```

```
# here we insert at top of chain and shove down
# allow MTU discovery
$IPT -t filter -I OUTPUT -p icmp --icmp-type 3/4 -j LOG --log-prefix "ICMP DON'T FRAGMENT  "
$IPT -t filter -I OUTPUT 2 -p icmp --icmp-type 3/4 -j ACCEPT

# allow echo replies to single workstation on internal network
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p icmp --icmp-type 0 -j ACCEPT

# permit ftp to single workstation on internal network
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p tcp --destination-port 20 -j LOG --log-prefix "FIREWALL
FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p tcp --destination-port 20 -j ACCEPT
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p tcp --destination-port 21 -j LOG --log-prefix "FIREWALL
FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p tcp --destination-port 21 -j ACCEPT
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p udp --destination-port 21 -j LOG --log-prefix "FIREWALL
FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p udp --destination-port 21 -j ACCEPT
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p udp --destination-port 20 -j LOG --log-prefix "FIREWALL
FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p udp --destination-port 20 -j ACCEPT

# permit telnet connections to single workstation on internal network
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p tcp --destination-port 23 -j LOG --log-prefix "FIREWALL
TELNET  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p tcp --destination-port 23 -j ACCEPT

# permit syslog connections to syslog server on network
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p udp --destination-port 123 -j LOG --log-prefix "FIREWALL
NTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p udp --destination-port 123 -j ACCEPT

# permit syslog connections to NTP server on network
#$IPT -t filter -I OUTPUT -d 192.168.2.2 -p udp --destination-port 514 -j LOG --log-prefix "FIREWALL
SYSLOG  "
$IPT -t filter -I OUTPUT 2 -d 192.168.2.2 -p udp --destination-port 514 -j ACCEPT

# permit connections to mail relay
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p tcp --destination-port 25 -j LOG --log-prefix "FIREWALL
MAIL  "
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p tcp --destination-port 25 -j ACCEPT

# APPEND
# permit established
$IPT -t filter -A OUTPUT -p tcp -m state --state ESTABLISHED,RELATED -j LOG --log-prefix
"FIREWALL PERMIT EST-TCP  "
$IPT -t filter -A OUTPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A OUTPUT -p udp -m state --state ESTABLISHED,RELATED -j LOG --log-prefix
"FIREWALL PERMIT EST-UDP  "
$IPT -t filter -A OUTPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

# permit access to all of firewall own interfaces
```

```
#$IPT -t filter -A OUTPUT -p tcp -d 192.168.0.1 -j LOG --log-prefix "FIREWALL TO SELF TCP01 "
"……remaining lines removed for the sake of brevity……
……

# firewall lockdown
#$IPT -t filter -A OUTPUT -p tcp -j LOG --log-prefix "BLOCKED TCP FROM FIREWALL "
$IPT -t filter -A OUTPUT -p tcp -j DROP
#$IPT -t filter -A OUTPUT -p udp -j LOG --log-prefix "BLOCKED UDP FROM FIREWALL "
$IPT -t filter -A OUTPUT -p udp -j DROP
```

With the protocols chains completed, let's begin work on our specialized attack detection chains. We will implement chains that specialize in stealth port scans, trojans, and address spoofing:

```
#####################################################################
# RULE_X: Attack & Stealth Port Scan Detection
#####################################################################
echo "Creating RULE_X: Stealth Scan Detection . . . . . "
# Block & Log SYN/FIN scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,FIN -j LOG --log-prefix "RULE_X DENY: SYN/FIN Scan "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,FIN -j DROP

# Block & Log FIN scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL FIN -j LOG --log-prefix "RULE_X DENY: FIN Scan "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL FIN -j DROP

# Block & Log ACK scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL ACK -j LOG --log-prefix "RULE_X DENY: ACK Scan"
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL ACK -j DROP

# Block & Log SYN/ACK scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK -j LOG --log-prefix "RULE_X DENY: SNY/ACK Scan "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK -j DROP

# Block & Log XMAS TREE scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK,URG,PSH,RST,FIN -j LOG --log-prefix "RULE_X DENY: XMAS TREE Scan "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK,URG,PSH,RST,FIN -j DROP

# Block & Log NULL scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL NONE -j LOG --log-prefix "RULE_X DENY: NULL Scan "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL NONE -j DROP

#####################################################################
# RULE_Y: Trojan Rules
#####################################################################
echo "Creating RULE_Y: Trojan Detection . . . . . "
# block netbus, subseven, and backorifice trojans
# here we insert at top of chain and shove down
```

```
# insert in FORWARD chain as needed
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 12345:12346 -j LOG --log-prefix
"NETBUS DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 12345:12346 -j DROP
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 6666:6667 -j LOG --log-prefix
"NETBUS DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 6666:6667 -j DROP
#$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 27374 -j LOG --log-prefix
"SUBSEVEN DENIED"
#$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 27374 -j DROP
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 2773:2774 -j LOG --log-prefix
"SUBSEVEN DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 2773:2774 -j DROP
#$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 7000 -j LOG --log-prefix "SUBSEVEN
DENIED"
#$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 7000 -j DROP
#$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 54320:54321 -j LOG --log-prefix
"BACK ORIFICE DENIED  "
#$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 54320:54321 -j DROP
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 31337:31338 -j LOG --log-prefix
"BACK ORIFICE DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 31337:31338 -j DROP

#####################################################################
# RULE_Z: Spoofing Rules (egress-filtering & ingress-filtering)
#####################################################################
echo "Creating RULE_Z: Spoofing Rules . . . . . egress & ingress filtering"
# block all routable addresses with source address of network

# EGRESS FILTERING
# block any packets coming from the network with spoofed addresses (egress filtering)
$IPT -t filter -I RULE_Z -i eth0 ! -s 192.168.0.0/24 -j LOG --log-prefix "INTERNAL SPOOF ATTEMPT
"
$IPT -t filter -I RULE_Z 2 -i eth0 ! -s 192.168.0.0/24 -j DROP
$IPT -t filter -I RULE_Z -i eth3 ! -s 192.168.1.0/24 -j LOG --log-prefix "INTERNAL SPOOF ATTEMPT
"
$IPT -t filter -I RULE_Z 2 -i eth3 ! -s 192.168.1.0/24 -j DROP
$IPT -t filter -I RULE_Z -i eth2 ! -s 192.168.2.0/24 -j LOG --log-prefix "INTERNAL SPOOF ATTEMPT
"
$IPT -t filter -I RULE_Z 2 -i eth2 ! -s 192.168.2.0/24 -j DROP

# INGRESS FILTERING
# block any packets coming into the network with spoofed addresses (ingress filtering)
$IPT -t filter -A RULE_Z -i eth1 -s 10.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 10.0.0.0/8 -j DROP
# comment out 172.16 for testing
#$IPT -t filter -A RULE_Z -i eth1 -s 172.16.0.0/12 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT
"
#$IPT -t filter -A RULE_Z -i eth1 -s 172.16.0.0/12 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.0.0/16 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT
"
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.0.0/16 -j DROP
```

```
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.4.0/22 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT
"
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.4.0/22 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.8.0/21 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT
" ……remaining lines removed for the sake of brevity……
……
```

Briefly here, we will demonstrate the "string" function, another powerful feature of iptables:

```
echo "Creating Specialized Rules . . . . . "
# Protecting Data
# It's very difficult to make sure someone on the inside isn't passing
# data through your firewall that shouldn't go out. We can try to prevent
# certain data from leaving by marking that data, then looking for that
# mark using the string match. It may be a good idea to implement a policy of
# putting a string, such as "Copyright: GIAC Enterprises - not for publication"
# at the top of those files you don't want sent through the firewall.
# Then, on the inner firewall, you might want something like:
# iptables -t filter -I FORWARD -i eth2 -m string --string="Copyright: GIAC Enterprises - not for
publication" -j DROP
```

The final step is to implement all the rule chains. We do this by linking our "user-defined" chains to the INPUT, OUTPUT, and FORWARD built-in chains as such:

```
#####################################################################
# Implement All Rules
#####################################################################
echo "Implementing All Firewall Rules . . . . . "
echo "Configuring so RULE_X_Y_Z can be turned ON and OFF easily."
#direct to appropriate rule chain
# call RULE_X only for new connection attempts

$IPT -t filter -A INPUT -p tcp -m state --state NEW -j RULE_X
$IPT -t filter -A INPUT -p tcp -j RULE_Y
$IPT -t filter -A INPUT -j RULE_Z
$IPT -t filter -A FORWARD -p tcp -m state --state NEW -j RULE_X
$IPT -t filter -A FORWARD -p tcp -j RULE_Y
$IPT -t filter -A FORWARD -j RULE_Z

$IPT -t filter -A INPUT -p tcp -j tcprules
$IPT -t filter -A INPUT -p udp -j udprules
$IPT -t filter -A INPUT -p icmp -j icmprules
$IPT -t filter -A INPUT -p 50 -j esprules
$IPT -t filter -A INPUT -p 51 -j ahrules
$IPT -t filter -A INPUT -p 89 -j ospfrules
$IPT -t filter -A INPUT -p 9 -j igrprules
$IPT -t filter -A INPUT -p 47 -j grerules

$IPT -t filter -A FORWARD -p tcp -j tcprules
$IPT -t filter -A FORWARD -p udp -j udprules
$IPT -t filter -A FORWARD -p icmp -j icmprules
```

```
$IPT -t filter -A FORWARD -p 50 -j esprules
$IPT -t filter -A FORWARD -p 51 -j ahrules
$IPT -t filter -A FORWARD -p 89 -j ospfrules
$IPT -t filter -A FORWARD -p 9 -j igrprules
$IPT -t filter -A FORWARD -p 47 -j grerules
```

An explanation of what ha been done here is in order. One powerful features of iptables is the ability for the user to create new chains, in addition to the three built-in ones (INPUT, FORWARD and OUTPUT). However, a user defined chain will only be called if it is the listed target of one of the built-in chains. We must then link the user defined chains we developed to at least one of the built-in chains in order to implement it. That is exactly what this last series of commands has accomplished.

When a packet matches a rule whose target is a user-defined chain, the packet begins traversing the rules in that user-defined chain. If that chain doesn't terminate the packet, then once traversal on that chain has finished, traversal resumes on the next rule in the current chain. Figure 9 demonstrates how this works.

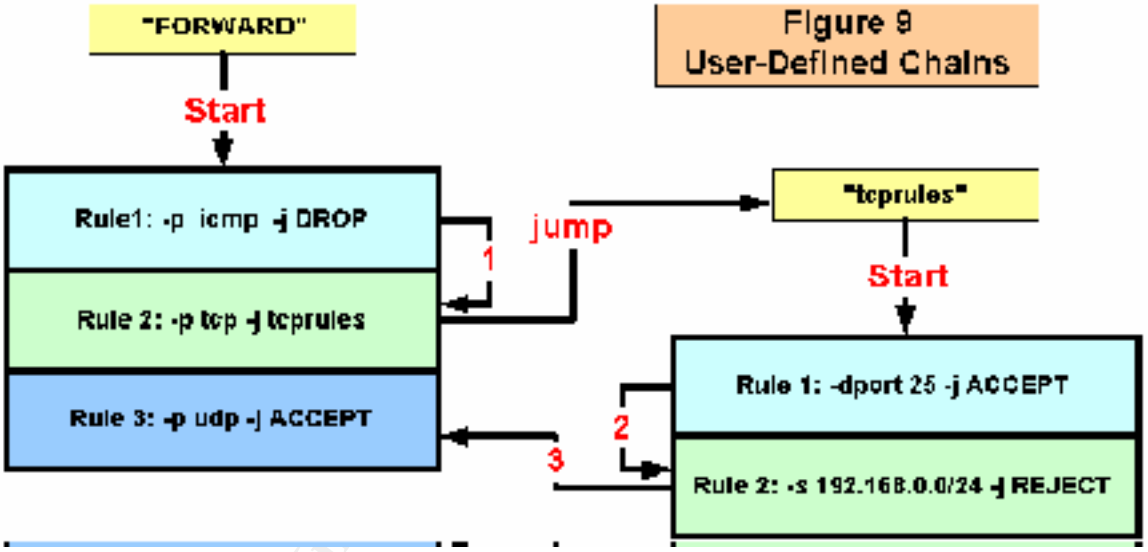


Figure 9
User-Defined Chains

Let's assume a packet begins its route by entering the "FORWARD" chain, where it is checked against Rule 1. If there is no match, the packet is then compared with Rule 2. The TARGET of Rule2 is the tcprules chain defined earlier. Thus, the packet then jumps to the "tcprules" chain, where it is first compared to Rule1 for a match and so on. If the packet traverses the complete "tcprules" chain without a match, it jumps back to the "FORWARD" chain at the point where it originally "jumped" the chain. From there, it continues on its way. The packet path is then:

- FORWARD chain Start
- 1
- jump

- tcprules chain Start
- 2
- 3
- FORWARD chain Continue

This functionally gives iptables a great deal of power and flexibility.

The firewall rule base is now in place. The last remaining task is to turn ip_forwarding back on:

```
# turn on ip_forwarding
echo "Turning on ip_forwarding . . . . . "
logger -p warning "Turning on ip_forwarding . . . . . "
echo 1 > $FWD
```

It would be highly remiss not to explain the superb logging features of netfilter/iptables that we have relied upon heavily throughout this script. The target LOG permits matched rules to be logged by the syslog daemon to the log files. The LOG option –log-prefix allows for the insertion of user-defined text up to 29 characters to be logged with the entry as well. This allows log parsing utilities to key on these same phrases. Needless to say, this greatly simplifies log analysis.

To get the most out of this feature, the rules need to be entered in matched pair. One pair will log the action, and the other will perform the action. For example:

```
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.0.1 -j LOG --log-prefix "SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 224.0.0.0/3 -j DROP
```

This rule pair logs “SPOOF ATTEMPT” if the source address of 192.168.0.1 shows up at eth1 with the first rule, and then drops the packet with the second rule. This functionality will be integrated with Psionic Technologies TriSentry Logcheck for a means of host-based intrusion detection.

VPN GATEWAY

GIAC Enterprises VPN gateway will be running from a SuSE 7.2 Linux box with iptables enabled with static routing only. As mentioned above, the FreeS/WAN version 1.95 implementation of IPsec will be the software configured and applied. Thus, security policy will largely be confined to the iptables startup file ipsec.rc and the FreeS/WAN ipsec.config file.

The VPN gateway will be supporting Ipsec connection from the Partners network in gateway-to-gateway tunnel mode, and connections from the remote sales force in host-to-gateway transport mode. That said, the basic security policy will be as follows:

```
##################################################################
# GIAC Enterprises VPN Security Policy                           #
##################################################################
```

1. The VPN gateway will utilize the IPsec standard.

2. Traffic will travel in tunnel mode for connections to and from the 192.168.100.0/24 network.

3. Traffic will travel in transport mode for connections from externally dynamically assigned IP addresses.

4. Automatic key generation using Internet Key Exchange (IKE) main mode will be applied.

5. Authentication will use 2048-bit RSA.

6. Set key life 300 at minutes.

7. Use 3DES encryption.

8. Use Encapsulating Security Payload (ESP) for the security protocol.

9. Use MD5 (Message Digest Version 5) for the Hashed Message Authentication Code (HMAC).

Although FreeS/WAN supports several options for authentication other hosts and gateway machines, public key RSA has been selected for the following reasons:

1. There is little security concern when distributing public keys to potential users. With a public key technique, you transmit only the public key. The system is designed to negate the effects of an attacker obtaining the public key. The private key never leaves the machine itself.

2. Key management is simplified. For instance, if shared secrets are used, the larger the number of connections, the more secrets that must be managed and secured. The number theoretically increases quadrilateral. This can become a great administrative burden, and a security problem in itself. With public key techniques, the only thing that must be kept secret is the private key. That, of course must be tightly secured and kept secret from everyone. Also, the number of public keys used only increases linearly relative to the number of nodes.

3. Static IP addresses are not required. When RSA authentication is in use, the initiator can identify itself by name before the key must be determined. The responder then checks that the message is signed with the public key corresponding to that name.

We will now apply this policy by properly configuring the ipsec.conf file. The configuration for GIAC Enterprises is shown below. The complete file can be found in Appendix F.

First, we again need to assign and activate the static routes:

```
echo "Setting default and static routes for server3 . . . . . "
logger -p warning "Setting default and static routes for server2 . . . . . "

# set default route for server3
route add default gw 192.168.1.1

# set static routes for vpn gateway
route add -net 172.16.0.0 netmask 255.255.0.0 dev eth1
route add -net 172.17.0.0 netmask 255.255.0.0 dev eth1
route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
route add -net 192.168.2.0 netmask 255.255.255.0 dev eth0
```

There are three sections to the ipsec.conf file. The first section is referred to as the configuration setup, or basic configuration. Here, the VPN interface is identified and set, and the debug mode for KLIPS and Pluto are defined. Parameters in this section can be over ridden if needed.

```
# basic configuration
config setup
    # THIS SETTING MUST BE CORRECT or almost nothing will work.
    # this section applies to all connections
    # external interface of VPN gateway is eth1
    interfaces="ipsec0=eth1"
    # Debug-logging controls:  "none" for (almost) none, "all" for lots.
    klipsdebug=none
    plutodebug=none
    # Use auto= parameters in conn descriptions to control startup actions.
    plutoload=%search
    plutostart=%search
```

The second section determines any connection defaults that need to be set. These parameters will apply to all connections. However, they are subject to override. In this section, we define IKE as the means of key exchange, define RSA as the means of authentication, and set the key lifetime value. method

```
# defaults for subsequent connection descriptions
conn %default
    # define items that need to be applied to all connections
    # general defaults here can be subject to override later
    # How persistent to be in (re)keying negotiations (0 means very).
    keyingtries=0
    # means to authenticate gateways
    # Set up autokeying using IKE and RSA
    keyexchange=ike
    keylife=8h
    # authentication can be by shared secrets or digital signatures
    # digital signatures are superior in every way to shared secrets
    authby=rsasig
    # load all connection descriptions by default
    # can override this later with auto=start
    auto=add
```

The default connection section is followed by the specific connection definitions, and is the actual Security Policy Database (SPD). Any connection type that is needed can be defined here. Our policy just requires one for the Partners Network and one for the GIAC Enterprises Road Warriors.

The first parameter listed is the connection type, which can be either "tunnel" or "transport". The majority of the section, however, dedicated to defining the connection parameters for both ends of the VPN connection. One side of the connection is referred to as the "left", and other is the "right". There is no correct order as wither side can be either "left" or "right".

FreeS/WAN is very flexible as to the placement or location of public RSA keys. We have included the appropriate keys in the connection section here, but they may be placed in a separate "include" file or even placed in DNS HINFO or TEXT records.

Finally, a suggested starting SPI number and the required encryption method for the connection is defined, and the connection is designated is loaded automatically upon request.

```
# PARTNERS
# vpn tunnel for partners network
# Tunnel mode, static IP, RSA key authentication
# Here we just use ESP for both encryption and authentication
conn partners
    type=tunnel
    # declare identify for authentication
    leftid=182.16.0.2
    leftrsasigkey=0x0f985uf8w35w4t98qr...
```

```
    # optionally may place keys in dns
    # leftrsasigkey=%dns
    # left security gateway (GIAC Enterprises) (public-network address)
    left=172.16.0.2
    # next hop to reach right (GIAC Enterprises router)
    leftnexthop=172.16.0.10
    # subnet behind left
    leftsubnet=192.168.1.0/24
    # declare identify for authentication
    rightid=172.18.0.2
    rightrsasigkey=0x05j4535o6drw76cf34ca...
    # optionally may place keys in dns
    # rightrsasigkey=%dns
    # right security gateway (partners network) (public-network address)
    right=172.18.0.2
    next hop to reach left
    rightnexthop=172.18.0.10
    rightsubnet=192.168.100.0/24
    # SPI number
    spi=0x200
    # (manual) encryption/authentication algorithm and parameters to it
    esp=3des-md5-96
    espenckey=[192 bits]
    espauthkey=[128 bits]
    auto=start

# ROAD WARRIORS
# host on one end and a subnet (behind a security gateway) on the other
# This case is also called "road warrior"
# the road warrior is getting a dynamic ip address from an isp
conn road-warrior
    type=transport
    # declare identify for authentication
    leftid=182.16.0.2
    leftrsasigkey=0x0f985uf8w35w4t98qr...
    # optionally may place keys in dns
    leftrsasigkey=%dns
    # left security gateway (GIAC Enterprises) (public-network address)
    left=172.16.0.2
    # next hop to reach right (GIAC Enterprises router)
    leftnexthop=172.16.0.10
    # subnet behind left
    leftsubnet=192.168.1.0/24
    # accept any address for right
    right=%any
    # any address, provided authentication works
    rightid=0.0.0.0
    rightrsasigkey=0xd9a24765fe...
```

```
#
# no subnet for a typical road warrior
# so the rightsubnet= parameter is omitted
# SPI number
spi=0x300
# (manual) encryption/authentication algorithm and parameters to it
esp=3des-md5-96
espenckey=[192 bits]
espauthkey=[128 bits]
# let the road warrior start the connection
auto=add
# override the default retry for road warriors
# we don't want to retry if IP connectivity is gone
keyingtries=1
```

With the VPN gateway connections configured, we must now configure the firewall code to permit the Ipsec protocols needed, and lockdown access to the machine itself. As we did for the firewall, we will accomplish this with a shell script, ipsec.rc that is run a system boot. Much of the initial parts of the script will duplicate what we have already reviewed in the firewall discussion. For that reason, we will only present the script portions that deal with protocol filtering. The ipsec.rc script can be found in its entirety in Appendix F.

The major concern is that the IPsec gateway must have packet filters that allow IPsec packets, at least when talking to other IPsec gateways. The basic methodology will be to allow IPsec packets, IKE on UDP port 500, ESP protocol 50, and perhaps AH protocol 51 incoming, if the destination address is that of the gateway, and outgoing with the from address of the gateway machine. From that point, Pluto handles all the IKE and KLIPS handles the ESP or AH. Firewall rules must then allow UDP port 500, and at least one of AH or ESP on the interface that communicates with the other gateway. Here are the applicable sections of the ipsec.rc script:

Once again, we must define default policies for all three tables, and again we default to "deny all unless expressly permitted."

```
#####################################################################
# Define Default Policy
#####################################################################
# default policy for firewall will be deny all unless explicitly permitted.
echo "Setting default policies . . . . . filter table"
logger-p warning "Setting default policies . . . . . filter table"
$IPT -t filter -P OUTPUT DROP
$IPT -t filter -P INPUT DROP
$IPT -t filter -P FORWARD DROP

echo "Setting default policies . . . . . nat table"
logger -p warning "Setting default policies . . . . . nat table"
```

```
$IPT -t nat -P PREROUTING DROP
$IPT -t nat -P OUTPUT DROP
$IPT -t nat -P POSTROUTING DROP

echo "Setting default policies . . . . . . . mangle table"
logger -p warning "Setting default policies . . . . . . . mangle table"
$IPT -t mangle -P PREROUTING DROP
$IPT -t mangle -P OUTPUT DROP
#####################################################################
```

Next we deal with our Ipsec protocols, which, of course, we want to let through. Thus we set the target to ACCEPT.

```
#####################################################################
# Create IPsec Rules
#####################################################################
# allow IPsec
#
# allow IKE negotiations
$IPT -A INPUT  -p udp --sport 500 --dport 500 -j ACCEPT
$IPT -A OUTPUT -p udp --sport 500 --dport 500 -j ACCEPT

# allow ESP encryption and authentication
$IPT -A INPUT  -p 50 -j ACCEPT
$IPT -A OUTPUT -p 50 -j ACCEPT

# uncomment if AH authentication header used
# $IPT -A INPUT  -p 51 -j ACCEPT
# $IPT -A OUTPUT -p 51 -j ACCEPT
```

Next we define a user chain that permits new stateful connections on the Ipsec interface ipsec0. Everything else will be dropped. We do need some administrative access though, so will insert some provisions for SSH, logging, time synchronization, and any outgoing mail that might be sent to “root” from another program. Finally, we must lock the machine down. The system is now ready for testing and auditing.

```
#####################################################################
# Create User Defined Chains
#####################################################################
echo "Creating User Defined Chains . . . . . . . . . . . . . . . . "
logger -p warning "Creating User Defined Chains . . . . . . . . . . . . . . . . "

# create a new chain that permits new connections on ipsec0
# but blocks everything else
$IPT -N block
$IPT -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A block -m state --state NEW -i eth1 -j ACCEPT
$IPT -A block -j DROP

# Jump to that chain from INPUT and FORWARD chains.
# $IPT -A INPUT -j block
```

```
# $IPT -A FORWARD -j block

#####################################################################
# VPN LOCKDOWN
#####################################################################
# permit all ssh connections from 192.168.0.3 only
$IPT -t filter -I INPUT -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j ACCEPT
$IPT -t filter -I FORWARD 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j ACCEPT

# permit syslog connections to syslog server on network
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p udp --destination-port 123 -j LOG --log-prefix "VPN NTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p udp --destination-port 123 -j ACCEPT

# permit syslog connections to NTP server on network
#$IPT -t filter -I OUTPUT -d 192.168.2.2 -p udp --destination-port 514 -j LOG --log-prefix "VPN
SYSLOG  "
$IPT -t filter -I OUTPUT 2 -d 192.168.2.2 -p udp --destination-port 514 -j ACCEPT

# permit connections to mail relay
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p tcp --destination-port 25 -j LOG --log-prefix "VPN MAIL  "
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p tcp --destination-port 25 -j ACCEPT

# deny all other connections from remaining devices on internal network
$IPT -t filter -A INPUT -i eth0 -p tcp -s 192.168.0.0/24 -j DROP
$IPT -t filter -A FORWARD -i eth0 -p tcp -s 192.168.0.0/24 -j DROP

# deny all other tcp connections from remaining devices on service network
$IPT -t filter -A INPUT -i eth0 -p tcp -s 192.168.1.0/24 -j DROP
$IPT -t filter -A FORWARD -i eth0 -p tcp -s 192.168.1.0/24 -j DROP

# deny all other tcp connections from remaining devices on admin network
$IPT -t filter -A INPUT -i eth0 -p tcp -s 192.168.2.0/24 -j DROP
$IPT -t filter -A FORWARD -i eth0 -p tcp -s 192.168.2.0/24 -j DROP

# permit connections from router and block everything else
$IPT -t filter -I INPUT -i eth1 -p tcp -s 172.16.0.10 -j ACCEPT
$IPT -t filter -I FORWARD -i eth1 -p tcp -s 172.16.0.10 -j ACCEPT
#$IPT -t filter -A OUTPUT -p tcp -j LOG --log-prefix "BLOCKED TCP FROM VPN  "
$IPT -t filter -A OUTPUT -p tcp -j DROP
#$IPT -t filter -A OUTPUT -p udp -j LOG --log-prefix "BLOCKED UDP FROM VPN  "
$IPT -t filter -A OUTPUT -p udp -j DROP

# turn on ip_forwarding
echo "Turning on ip_forwarding . . . . . "
logger -p warning "Turning on ip_forwarding . . . . . "
echo 1 > $FWD
#####################################################################
```

HOST HARDENING & LOCKDOWN

This report will not directly be concerned with the hardening and lockdown of GIAC Enterprises Windows 2000 servers positioned on the internal network. However, at additional cost, GIAC Enterprises may optionally purchase "WSAS" (Windows Security

Auditing and Scanning Tool) proprietary software from Bandwidthco Security Services. "WSAS", a suite of perl scripts, performs a thorough scan of a Windows 2000 server and automatically corrects or repairs security configuration errors and vulnerabilities. Close attention will be paid to all Linux servers on the service network and administrative network, as well as the firewall box itself.

Linux servers will all be hardened in the following manner:

- Identify and disable (remove) all unnecessary services.
- Identify and disable (remove) all unnecessary programs, particularly programs or scripts with *setuid* and *setgid* privilege.
- Disable routing and routing services and protocols as necessary.
- Disable all unnecessary accounts.
- Modify banners to prevent information leakage.
- Bastille hardening applied.
- Install and stringently configure TCPWrapper.
- Install and stringently configure Psionic TriSentry Suite.

Services

The following services will be left enabled:

- Inetd - active services will be protected and controlled with TCPWrapper ACL's.
- Init – source code will be reviewed on all enabled run scripts.
- Swap
- Page
- Cron
- Syslogd
- Booting services to include tftp, bootd, bootpd, and dhcpd

The following services will be necessarily disabled or removed:

- NFS and all related services.
- RPC and all related services including NIS
- BSD "r" services
- Routed
- Fingerd
- Ftpd

- Uucpd
- Rwhod
- Lpd

Programs and Applications

All programs deemed necessary for the operation of the system will be regularly scanned by cron for setuid/setgid bit permissions as follows:

find / -type f \(-perm –0400 –o perm -02000 \ ) –ls

Bastille Hardening

Bastille Linux is a powerful set of system-hardening Perl scripts that secures Linux systems and allows for a custom security configuration on each system. Bastille will perform configuration checking on the following subsystems:

- IPChains and IPTables
- File permissions
- Patch status and updates
- Boot security
- Inetd to include warning banners
- System and user tools
- PAM
- Logging
- Service Daemons
- Sendmail
- Remote access

TCP Wrappers

TCP Wrappers will be configured to control access to all remaining active inet daemons on the system by forcing all connections through the tcpd daemon process. Attempted access by unauthorized persons or processes will be effectively handled within the /etc/hosts.deny file by notifying the system administrator through email, and by presenting the intruder with a warning banner. The /etc/hosts.deny ACL file will be configured similarly as follows:

ALL : ALL
: severity auth.warning
: spawn finger –l @%h | (/usr/bin/mail –s "denied tcp\: %h[%a] %d)" root &
: banner /usr/local/banners
: twist /bin/echo "You are not authorized to use this system"

Wrapper rules will be checked with the tcpdmatch tool provided with the TCPWrapper program.

TriSentry

Psionic TriSentry is a freely available suite of three tools (PortSentry, LogSentry, and Host Sentry) that work together to perform attack detection, prevention, and warning, and log analysis and reporting.

PortSentry is a port scan detector capable of detecting the following:

- Strobe-style scans (full connect() scans)
- SYN/Half open scans
- FIN scans
- NULL scans
- XMAS scans
- UDP scans

PortSentry will be configured to work in conjunction with route, iptables, and TCPWrapper to block any scan or intrusion it detects with the following entries in the portsentry.conf file:

```
# Linux supports the reject flag for the route command
KILL_ROUTE="/sbin/route add -host $TARGET$ reject"

# iptables support
KILL_ROUTE="/usr/local/bin/iptables -I INPUT -s $TARGET$ -j DROP"

# tcp wrappers support
KILL_HOSTS_DENY="ALL: $TARGET$"
```

The LogSentry package is designed to automatically run and check system log files for security violations and unusual activity. Logcheck utilizes a program called “logtail” that remembers the last position read from in a log file and uses this position on subsequent runs to process new information. Any suspicious activity, as determined by entries placed in the logcheck.violations file, will be compiled into a report and mailed to the system administrator. For the purposes of GIAC Enterprises, all the LOG –log-prefix options included in the iptables configuration will be added to the logcheck.violations file and read automatically by LogSentry. Administrator notification will be enabled with the following entry in the LogSentry configuration file:

```
# send log activity to
SYSADMIN=root@giac.com
```

HostSentry is a host based intrusion detection tool that operates on the principle of Login Anomaly Detection (LAD), or what is called login cross-correlation. HostSentry utilizes the wtmp and utmp files, which store all user login/logout information including

username, login host, login, time, and login tty. HostSentry will then react to any suspicious login or logout activity and report it.

SECTION 3 – Firewall Security Policy Audit

AUDIT PLAN

Overview

GIAC Enterprises "Comprehensive Security Plan" is now implemented. Management has asked Bandwidthco Security Services to conduct an audit of the primary firewall to verify that security policies are correctly enforced, and to identify any vulnerability that may exist within the system. We, of course, have agreed to the undertaking.

A security audit can prove beneficial and be a valuable asset to an organization, but only if they are carefully planned and executed, and properly documented. Hence, we must first define an objective, and consider all aspects of the operation and organization that may be affected. Additionally, we want the results to be meaningful and useful.

With all this in mind, the general audit process will consist of the following stages and events:

1. Brief GIAC Enterprises management on the inherent risks involved with performing a comprehensive audit. For instance, management must be fully aware that some systems may not react well to certain tests and may hang, crash, or suffer serious corruption. Management must also be aware that system resources and bandwidth will be heavily stressed during the course of the audit.

2. Prepare a written audit strategy and plan, and carefully review it with GIAC Enterprises management. The plan should contain cost and time factors estimates.

3. Obtain written permission from GIAC Enterprises management to perform the audit as detailed by the written plan.

4. Verify that all system backups are current and available for reliable restoration prior to execution of the audit plan. Additionally, redundant systems should also be verified as functioning properly.

5. Perform the audit.

6. Analyze and evaluate the data.

7. Provide recommendations.

Time, Manpower & Costs

To minimize the impact on both the financial and internal functions of GIAG Enterprises operations, the security audit will be performed on weekends, only between the hours of 6:00 pm and 6:00 am. Two certified security analysts will be assigned to conduct the audit.

Security Analysts:	**2**
Man-hours:	**48**
Cost per man-hour:	**$150.00**
Audit Cost:	**$7,2000**
Analytical Report:	**$1000.00**
Total Cost:	**$8,200.00**

STRATEGY & METHODOLOGY

The audit will consist of a series of seven separate test groups:

- Physical Security of the firewall and related or interconnected devices.
- Port scans of the firewall from the internal network segment and the Internet.
- Firewall rule base audit from the internal network and the Internet.
- Denial of Service (DOS) challenge.
- DNS performances scan.
- Mail relay performances scan.
- System logging effectiveness.

Table 3 below explains and summarizes the purpose of each test group.

Tools

The following tools will be used in testing of the firewall:

- nmap 2.54
- windump
- tcpdump
- netstat
- nslookup
- dig
- sendmail 8.11.3
- Sam Spade 1.14
- Nessus 1.2.3
- Nemesis 1.32

TABLE 3 Firewall Audit Test Groups		
Test Group	**Purpose**	**Desired Result**
Physical Security	Evaluate access and determine physical access limitations to firewall device and related equipment	Physical access to hardware, consoles, input devices is restricted and only accessible by authorized personnel.
Firewall Ports	Evaluate access to ports and services on the firewall unit. Consider TCP, UDP, and ICMP protocols.	No open ports or services available.
Firewall Rule Base	Evaluate the enforcement of security policy. Verify default deny.	All rules are performing effectively and in the correct order. The security policy is effectively enforced and operations perform as expected.
DOS	Evaluate system performance, resistance, and response to abnormal, stressful, and demanding conditions.	The firewall resists attack or continues to perform well while under attack.
DNS	Evaluate DNS record security and information leakage.	DNS does not leak or provide critical network information to external sources.
Mail Relay	Evaluate mail relaying properties and propensities of mail relay server	The mail relay only accepts and relays mail for the private network only. Anti-spam measures are functioning properly.
System Logging	Evaluate logging facility effectiveness.	The system logging facility captures traffic information and detects scanning activity.

Open Ports On Firewall

Determining "open" or "listening" ports on the firewall is a rather straightforward process. We will first determine what the firewall machine itself believes its listening ports to be by running a simple **netstat –l** command. The next six test will be remote port scans of the firewall using nmap targeting the firewall from both the internal network and the internet (see Figure 10). Scans for tcp, upd, and icmp using verbose output will be performed on all 65,000 ports.

REM From Internet (172.17.0.2):

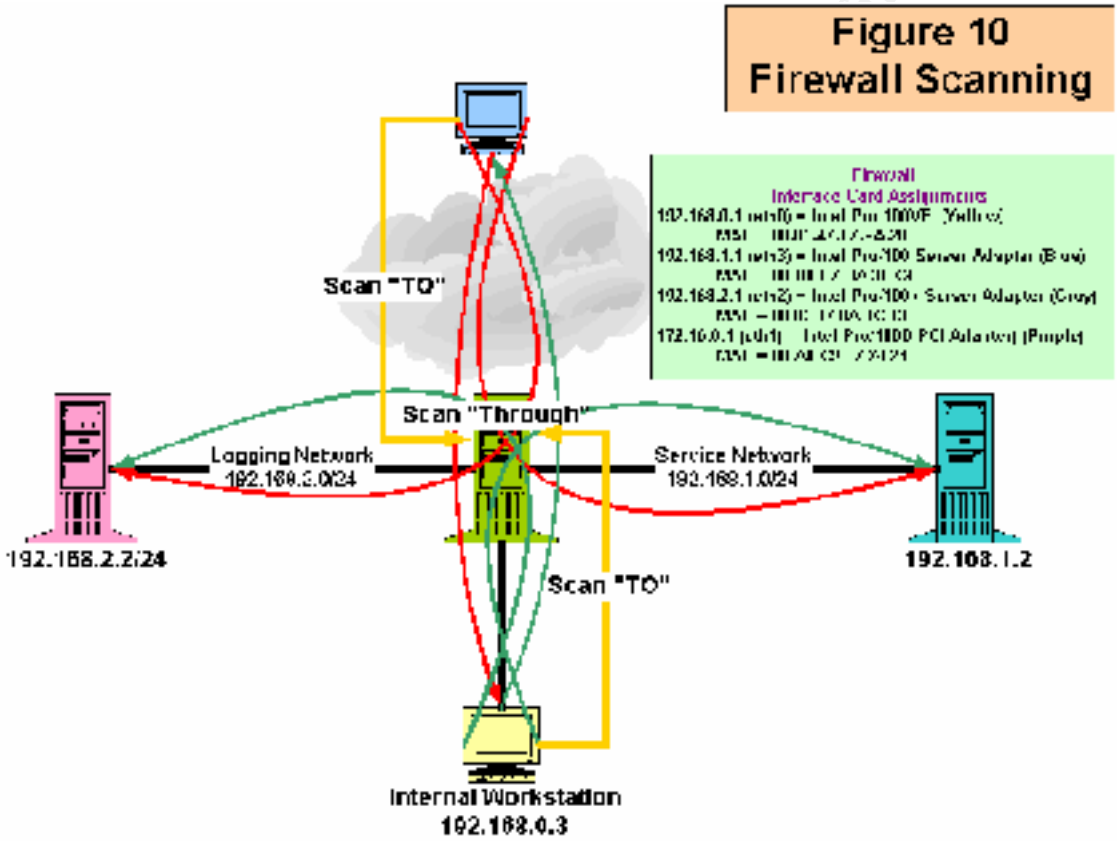
Test 2: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_2.txt 192.168.0.1
Test 3: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_3.txt 192.168.0.1
Test 4: nmap -v -g53 -sS -sR -PI -O -T 3 -p1-65000 -oN C:\temp\test_4.txt 192.168.0.1

REM From Internal Network (192.168.0.3):

Test 5: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_5.txt 192.168.0.1
Test 6: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_6.txt 192.168.0.1
Test 7: nmap -v -g53 -sS -sR -PI -O -T 3 -p1-65000 -oN C:\temp\test_7.txt 192.168.0.1

Firewall Rule Base

Testing the firewall rule base will involve pushing packets "through" the firewall instead of directing them "to" the firewall. In doing so, the objective will be to determine what ports are not filtered by the firewall rule base. We can apply port-scanning techniques to accomplish this as well. However, we must scan form a host, through the firewall, to another host behind the firewall or in front of the firewall. In order to test all firewall interfaces, scans must be made from all directions, to and from all network segments (see Figure 10).



Filtered TCP ports can effectively determined with nmap either using the Syn Stealth scan or an ACK scan. The nmap Syn Stealth, or half open SYN scan works by initiating a new connection with a remote host by sending the initial SYN. The remote host. If the packet successfully made it past the firewall filter, the remote host should respond with a SYN/ACK pair. The scan is considered stealth because nmap will quickly shut the connection down at this point. Hopefully, this will avoid detection or logging of any type.

The nmap ACK scan was designed specifically for testing a firewall rule base works. It works by sending a lone ACK packet through the firewall to a remote host. An ACK packet will always receive a RST packet in response if the packet was able to successfully traverse the firewall. We will be using the Syn Stealth scan for the testing of all 65,000 TCP ports.

The testing of UPD port filters is considerable hampered by the connectionless nature of UDP, and is therefore much more difficult. Meaningful responses from the remote host cannot be relied upon. In fact, if a given port is open, and the host accepts the packet, no response at all will be returned. Even if a response did return from the host, that would only tell us the status of port on the remote host, on supply an insight as to whether the port was filtered on the firewall. To overcome the limitations or restrictions of UPD, we will need to deploy a protocol sniffer on the backside of the firewall to see exactly which protocol packets are making it through. Once this determination is made, it can be assumed the firewall is not filtering that particular UDP port.

One additional obstacle is in the way. UDP communications are by nature extremely noisy. Thus we will need a means to separate the packets that came through the firewall from other packets on the network. To solve this problem, we will set the source port on the nmap scan to a known port with the –g flag, and then key the sniffer on the other side of the firewall on that port. For instance, if we set the source port on the nmap scan to ***–g 53***, we can then set tcpdump or windump parameters to ***tcpdump src port 53*** to help determine which scan packets had come through the firewall.

To test the firewall rule base, we will perform the following nmap scans for TCP ports, and nmap scans with tcpdump or windump for UDP ports:

REM From Internet (172.17.0.2):
Test 8: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_8.txt 192.168.0.3
Test 9: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_9.txt 192.168.1.2
Test 10: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_10.txt 192.168.2.2

Test 11: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_11.txt 192.168.0.3
Test 12: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_12.txt 192.168.1.2
Test 13: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_13.txt 192.168.2.2

REM From Internal Network (192.168.0.3):
Test 14: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_14.txt 172.17.0.2
Test 15: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_15.txt 192.168.1.2
Test 16: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_16.txt 192.168.2.2

Test 17: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_17.txt 172.17.0.2
Test 18: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_18.txt 192.168.1.2
Test 19: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_19.txt 192.168.2.2

To test spoofing, we will perform the following nmap scans with –S option and RFC 1918 addresses. The firewall logs must then be checked to determine if the packets were indeed dropped:

From Internet (172.17.0.2):
Test 20: nmap -v -g53 -sS -P0 -e eth0 -S 10.0.0.10 -O -T 3 -p1-1024 -oN C:\temp\test_20.txt 192.168.0.3
Test 20: nmap -v -g53 -sS -P0 -e eth0 -S 172.20.1.1 -O -T 3 -p1-1024 -oN C:\temp\test_20.txt --append_output 192.168.0.3
Test 20: nmap -v -g53 –sS -P0 -e eth0 -S 192.168.90.1 -O -T 3 -p1-1024 -oN C:\temp\test_20.txt --append_output 192.168.0.3

From Internal Network (192.168.0.3):
Test 21: nmap -v -g53 –sS -P0 -e eth0 -S 10.0.0.10 -O -T 3 -p1-1024 -oN C:\temp\test_21.txt 172.17.0.2
Test 21: nmap -v -g53 -sS -P0 -e eth0 -S 172.20.1.1 -O -T 3 -p1-1024 -oN C:\temp\test_21.txt --append_output 172.17.0.2
Test 21: nmap -v -g53 –sS -P0 -e eth0 -S 192.168.90.1 -O -T 3 -p1-1024 -oN C:\temp\test_21.txt --append_output 172.17.0.2

DOS Attack

Nessus will be used to perform various stressful and diagnostics attacks on the firewall.

MAIL & DNS

The external mail relay was configured to handle mail for the GIAC Enterprises network only. Relaying must be tested to determine if it's performing according to the configuration, and that it cannot be abused for spamming purposes.

The external primary (master) DNS server was configured to perform recursion only for the local network. Recursion must be tested to determine its performing according to the configuration, and that it cannot be abused. Additionally, the external DNS server was configured to only permit zone transfers to the secondary DNS on the service network. This configuration must be tested to determine that zone transfers cannot be executed by other systems or hosts, and that sensitive records and network information cannot be leaked.

LOGS

Review firewall logs to determine if scans detected and verify that the appropriate traffic is being logged.

Proposed Audit Summary

Table 4 below provides a summary of all the test to be performed.

TABLE 4 Firewall Audit Strategy & Methodology Summary						
Test No.	**Test Type**	**Test Name**	**Protocol**	**Src. IP**	**Dst. IP**	**Tools**
1	Listening Ports	netstat	TCP	Firewall	Firewall	netstat
2	Open Ports	Port Scan To Syn Stealth/RPC	TCP	172.17.0.2	Firewall	nmap
3	Open Ports	Port Scan To UDP/RPC	UDP	172.17.0.2	Firewall	nmap
4	Open Ports	Port Scan To Syn Stealth/ICMP Ping	ICMP	172.17.0.2	Firewall	nmap
5	Open Ports	Port Scan To Syn Stealth/RPC	TCP	192.168.0.3	Firewall	nmap
6	Open Ports	Port Scan To UDP/RPC	UDP	192.168.0.3	Firewall	nmap
7	Open Ports	Port Scan To Syn Stealth/ICMP Ping	ICMP	192.168.0.3	Firewall	nmap
8	Rule Base	Port Scan Through Syn Stealth/RPC	TCP	172.17.0.2	192.168.0.3	nmap
9	Rule base	Port Scan Through Syn Stealth/RPC	TCP	172.17.0.2	192.168.1.2	nmap
10	Rule Base	Port Scan Through Syn Stealth/RPC	TCP	172.17.0.2	192.168.2.2	nmap
11	Rule Base	Port Scan Through UDP/RPC	UDP	172.17.0.2	192.168.0.3	nmap windump
12	Rule Base	Port Scan Through UDP/RPC	UDP	172.17.0.2	192.168.1.2	nmap tcpdump
13	Rule Base	Port Scan Through UDP/RPC	UDP	172.17.0.2	192.168.2.2	nmap tcpdump
14	Rule Base	Port Scan Through Syn Stealth/RPC	TCP	192.168.0.3	172.17.0.2	nmap
15	Rule Base	Port Scan Through Syn Stealth/RPC	TCP	192.168.0.3	192.168.1.2	nmap
16	Rule Base	Port Scan Through Syn Stealth/RPC	TCP	192.168.0.3	192.168.2.2	nmap
17	Rule Base	Port Scan Through UDP/RPC	UDP	192.168.0.3	172.17.0.2	nmap windump
18	Rule Base	Port Scan Through UDP/RPC	UDP	192.168.0.3	192.168.1.2	nmap tcpdump
19	Rule Base	Port Scan Through UDP/RPC	UDP	192.168.0.3	192.168.2.2	nmap tcpdump
20	Spoofing Rules	Syn Stealth	TCP	172.17.0.2	192.168.0.3	nmap logs
21	Spoofing Rules	Syn Stealth	TCP	192.168.0.3	172.17.0.2	nmap logs
22	DOS	Teardrop	TCP	172.17.0.2	Firewall	nessus
23	DNS Recursion	Recursion Request	DNS	172.17.0.2 192.168.0.3	192.168.1.2 192.168.1.2	Nslookup SamSpade
24	Zone Transfer	Request Transfer	DNS	172.17.0.2 192.168.0.3	192.168.1.2 172.17.0.2	Nslookup SamSpade
25	Mail Relay	Mail Relay Request	SMTP	172.17.0.2	192.168.1.2	Sendmail SamSpade

FIREWALL AUDIT RESULTS

Open Ports

Test 1: Netstat run from the firewall machine reveals tcp/printer and 111/udp ports to be open. These services must be disabled.

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:login                 *:*                     LISTEN
tcp        0      0 *:filenet-nch           *:*                     LISTEN
tcp        0      0 *:printer               *:*                     LISTEN
tcp        0      0 *:time                  *:*                     LISTEN
tcp        0      0 *:auth                  *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
udp        0      0 *:syslog                *:*
udp        0      0 *:ndmp                  *:*
udp        0      0 *:time                  *:*
udp        0      0 *:sunrpc                *:*
udp        0      0 server2.bandwidthco:ntp *:*
udp        0      0 server2.bandwidthco:ntp *:*
udp        0      0 server2.bandwidthco:ntp *:*
udp        0      0 server2.bandwidthco:ntp *:*
udp        0      0 localhost:ntp           *:*
udp        0      0 *:ntp                   *:*
raw        0      0 *:tcp                   *:*
```

Test 2: nmap Syn Stealth scan run from internet shows only tcp/25 and tcp/80 ports open on firewall box as they should be. All other ports are disabled or filtered.

```
# nmap (V. 2.54BETA36) scan initiated Tue Jul 30 21:32:48 2002 as: nmap -v -g53 -sS -sR -P0 -O -T 3 -
p1-65000 -oN C:\temp\test_2.txt 192.168.0.1
Interesting ports on  (192.168.0.1):
(The 64997 ports scanned but not shown below are in state: filtered)
Port       State       Service (RPC)
25/tcp     open        smtp
113/tcp    open        auth
8080/tcp   closed      http-proxy
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 13.244 days (since Wed Jul 17 18:06:12 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=4954341 (Good luck!)
IPID Sequence Generation: All zeros

# Nmap run completed at Tue Jul 30 23:57:17 2002 -- 1 IP address (1 host up) scanned in 8669 seconds
```

These findings are also confirmed by a review of the firewall logs that show tcp ports are being effectively blocked from the Internet.

Jul 30 21:24:21 server2 kernel: BLOCK TCP ATTEMPT IN=eth1 OUT= MAC=00:a0:c9:f7:24:24:00:d0:58:a4:2b:96:08:00 SRC=172.17.0.2 DST=192.168.0.1 LEN=40 TOS=0x00 PREC=0x00 TTL=40 ID=57359 PROTO=TCP SPT=54 DPT=35936 WINDOW=2048 RES=0x00 SYN URGP=0

Jul 30 21:24:21 server2 kernel: BLOCK TCP ATTEMPT IN=eth1 OUT= MAC=00:a0:c9:f7:24:24:00:d0:58:a4:2b:96:08:00 SRC=172.17.0.2 DST=192.168.0.1 LEN=40 TOS=0x00 PREC=0x00 TTL=40 ID=27990 PROTO=TCP SPT=54 DPT=4825 WINDOW=2048 RES=0x00 SYN URGP=0

Test 3: nmap UDP scan from Internet reveals no active UDP services or open ports on firewall machine.

nmap (V. 2.54BETA36) scan initiated Tue Jul 30 23:57:18 2002 as: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_3.txt 192.168.0.1
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 65000 scanned ports on (192.168.0.1) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=7/31%Time=3D48BCEC%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed at Wed Jul 31 21:45:32 2002 -- 1 IP address (1 host up) scanned in 78494 seconds

Test 4: nmap ICMP scan against firewall from Internet reveals no information.

nmap (V. 2.54BETA36) scan initiated Wed Jul 31 21:45:33 2002 as: nmap -v -g53 -sS -sR -PI -O -T 3 -p1-65000 -oN C:\temp\test_4.txt 192.168.0.1
Nmap run completed at Wed Jul 31 21:46:04 2002 -- 1 IP address (0 hosts up) scanned in 30 seconds

Test 5: nmap Syn Stealth scan of firewall from internal network shows ports 37/tcp, 111/tcp, 513/tcp, 515/tcp, 3000/tcp, 6000/tcp, and one unknown port to be open. These ports must be disabled. 1241/tcp and 3001/tcp are open for nessus scans.

nmap (V. 2.54BETA36) scan initiated Tue Jul 30 22:39:44 2002 as: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN C:\temp\test_5.txt 192.168.0.1
Interesting ports on server2.bandwidthco.com (192.168.0.1):
(The 64985 ports scanned but not shown below are in state: closed)

Port	State	Service (RPC)
22/tcp	open	ssh
23/tcp	open	telnet
25/tcp	open	smtp
37/tcp	open	time
80/tcp	open	http
111/tcp	open	sunrpc
113/tcp	open	auth

```
513/tcp    open       login
515/tcp    open       printer
1241/tcp   open       msg
3000/tcp   open       ppp
3001/tcp   open       nessusd
6000/tcp   open       X11
10000/tcp  open       snet-sensor-mgmt
32769/tcp  open       unknown
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.063 days (since Tue Jul 30 21:09:06 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=4020201 (Good luck!)
IPID Sequence Generation: All zeros

# Nmap run completed at Tue Jul 30 22:40:01 2002 -- 1 IP address (1 host up) scanned in 17 seconds
```

Test 6: nmap UDP scan of firewall from internal network shows all UDP ports effectively filtered.

```
# nmap (V. 2.54BETA36) scan initiated Tue Jul 30 22:40:01 2002 as: nmap -v -g53 -sU -sR -P0 -O -T 3 -p1-65000 -
oN C:\temp\test_6.txt 192.168.0.1
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 65000 scanned ports on server2.bandwidthco.com (192.168.0.1) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D49D25C%O=-1%C=-1)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=N)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=N)

# Nmap run completed at Thu Aug 01 17:29:16 2002 -- 1 IP address (1 host up) scanned in 154155 seconds
```

Test 7: nmap ICMP scan against firewall from internal network reveals no unusual information.

Firewall Rule Base

Test 8: nmap Syn Stealth scan from internet to internal network shows all but two tcp ports to be filtered. TCP rules inbound appear to be very effective.

```
# nmap (V. 2.54BETA36) scan initiated Wed Jul 31 21:46:04 2002 as: nmap -v -g53 -sS -sR -P0 -O -T 3 -
p1-65000 -oN C:\temp\test_8.txt 192.168.0.3
Interesting ports on  (192.168.0.3):
(The 64998 ports scanned but not shown below are in state: filtered)
Port       State      Service (RPC)
25/tcp     open       smtp
113/tcp    closed     auth
No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-
bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D48E747%O=25%C=113)
```

```
TSeq(Class=RI%gcd=1%SI=225F79%IPID=Z%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=225635%IPID=Z%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=22563F%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T4(Resp=N)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Uptime 14.278 days (since Wed Jul 17 18:06:17 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=2250303 (Good luck!)
IPID Sequence Generation: All zeros

# Nmap run completed at Thu Aug 01 00:46:16 2002 -- 1 IP address (1 host up) scanned in 10812
seconds
```

Test 9: nmap Syn Stealth scan from Internet to service network shows all but two tcp ports to be filtered. TCP rules inbound and DNAT port redirection appears to be very effective.

```
# nmap (V. 2.54BETA36) scan initiated Thu Aug 01 00:46:16 2002 as: nmap -v -g53 -sS -sR -P0 -O -T 3 -
p1-65000 -oN C:\temp\test_9.txt 192.168.1.2
Interesting ports on  (192.168.1.2):
(The 64997 ports scanned but not shown below are in state: filtered)
Port       State       Service (RPC)
25/tcp     open        smtp
113/tcp    open        auth
8080/tcp   open        http-proxy
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 14.412 days (since Wed Jul 17 18:06:18 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=1520936 (Good luck!)
IPID Sequence Generation: All zeros

# Nmap run completed at Thu Aug 01 03:59:21 2002 -- 1 IP address (1 host up) scanned in 11585
seconds
```

Firewall logs also verify TCP inbound rules to be functioning effectively.

```
Aug  1 00:59:33 server2 kernel: BLOCK TCP ATTEMPT  IN=eth1 OUT=eth3 SRC=172.17.0.2
DST=192.168.1.2 LEN=40 TOS=0x00 PREC=0x00 TTL=53 ID=33127 PROTO=TCP SPT=53
DPT=53130 WINDOW=4096 RES=0x00 SYN URGP=0

Aug  1 00:59:33 server2 kernel: BLOCK TCP ATTEMPT  IN=eth1 OUT=eth3 SRC=172.17.0.2
DST=192.168.1.2 LEN=40 TOS=0x00 PREC=0x00 TTL=53 ID=4946 PROTO=TCP SPT=53 DPT=8833
WINDOW=4096 RES=0x00 SYN URGP=0
```

Test 10: nmap Syn Stealth scan from internet to administrative network shows all but two tcp ports to be filtered. TCP rules inbound appear to be very effective.

```
# nmap (V. 2.54BETA36) scan initiated Thu Aug 01 03:59:22 2002 as: nmap -v -g53 -sS -sR -P0 -O -T 3 -
p1-65000 -oN C:\temp\test_10.txt 192.168.2.2
Interesting ports on  (192.168.2.2):
(The 64997 ports scanned but not shown below are in state: filtered)
Port       State       Service (RPC)
25/tcp     open        smtp
113/tcp    open        auth
8080/tcp   closed      http-proxy
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 14.531 days (since Wed Jul 17 18:06:19 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=2557334 (Good luck!)
IPID Sequence Generation: All zeros

# Nmap run completed at Thu Aug 01 06:50:42 2002 -- 1 IP address (1 host up) scanned in 10280
seconds
```

Firewall logs also verify TCP inbound rules are functioning effectively.

```
Aug  1 05:06:23 server2 kernel: BLOCK TCP ATTEMPT  IN=eth1 OUT=eth2 SRC=172.17.0.2
DST=192.168.2.2 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=2413 PROTO=TCP SPT=53 DPT=18094
WINDOW=3072 RES=0x00 SYN URGP=0

Aug  1 05:06:23 server2 kernel: BLOCK TCP ATTEMPT  IN=eth1 OUT=eth2 SRC=172.17.0.2
DST=192.168.2.2 LEN=40 TOS=0x00 PREC=0x00 TTL=44 ID=35596 PROTO=TCP SPT=53 DPT=917
WINDOW=3072 RES=0x00 SYN URGP=0
```

Test 11: nmap UDP scan from Internet to internal network shows only 53/udp and 123/udp ports not filtered. This was determined by a perl script parsing the windump file of captured packets gathered from the internal network throughout the duration of the scan. UDP rules inbound appear to be very effective.

Test 12: nmap UDP scan from Internet to service network shows only 53/udp, 123/udp, and 514/udp ports not filtered. This was determined by a perl script parsing the windump file of captured packets gathered from the internal network throughout the duration of the scan. UDP rules inbound appear to be very effective.

Test 13: nmap UDP scan from Internet to administrative network shows only 53/udp, 123/udp, and 514/udp ports not filtered. This was determined by a perl script parsing the windump file of captured packets gathered from the internal network throughout the duration of the scan. UDP rules inbound appear to be very effective.

Test 14: nmap Syn Stealth scans from internal network to Internet shows 135/tcp and 139/tcp as open and not filtered. These ports must be closed. Possible problem with outbound TCP rule base.

```
# nmap (V. 2.54BETA36) scan initiated Sat Aug 03 17:38:03 2002 as: nmap -v -g53 -sS -P0 -O -T 3 -p1-
65000 -oN C:\temp\test_14.txt 172.17.0.2
Interesting ports on WORKSTATION2 (172.17.0.2):
(The 64788 ports scanned but not shown below are in state: closed)
Port       State       Service
23/tcp     open        telnet
35/tcp     filtered    priv-print
36/tcp     filtered    unknown
37/tcp     filtered    time
38/tcp     filtered    rap
39/tcp     filtered    rlp
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
445/tcp    filtered    microsoft-ds
1028/tcp   open        unknown
6000/tcp   filtered    X11
6001/tcp   filtered    X11:1
6002/tcp   filtered    X11:2
6003/tcp   filtered    X11:3
6004/tcp   filtered    X11:4
6005/tcp   filtered    X11:5
6006/tcp   filtered    X11:6
6007/tcp   filtered    X11:7
6008/tcp   filtered    X11:8
6009/tcp   filtered    X11:9
6010/tcp   filtered    unknown
6011/tcp   filtered    unknown
……..
……..
6200/tcp   filtered    unknown
6667/tcp   filtered    irc
Remote operating system guess: Windows Millennium Edition (Me), Win 2000, or WinXP
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=11123 (Worthy challenge)
IPID Sequence Generation: Incremental

# Nmap run completed at Sat Aug 03 17:52:02 2002 -- 1 IP address (1 host up) scanned in 839 seconds
```

Test 15: nmap Syn Stealth scan from internal network to service network shows 79/tcp, 110/tcp, 111/tcp, 513/tcp, 515/tcp, 135/tcp and 139/tcp as open and not filtered. These ports must be closed. Possible problem with outbound TCP rule base.

```
# nmap (V. 2.54BETA36) scan initiated Sat Aug 03 17:52:02 2002 as: nmap -v -g53 -sS -P0 -O -T 3 -p1-
65000 -oN C:\temp\test_15.txt 192.168.1.2
Interesting ports on server3.bandwidthco.com (192.168.1.2):
(The 64772 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
23/tcp     open        telnet
25/tcp     open        smtp
35/tcp     filtered    priv-print
```

```
36/tcp    filtered   unknown
37/tcp    filtered   time
38/tcp    filtered   rap
39/tcp    filtered   rlp
53/tcp    open       domain
79/tcp    open       finger
80/tcp    open       http
110/tcp   open       pop-3
111/tcp   open       sunrpc
113/tcp   open       auth
445/tcp   filtered   microsoft-ds
513/tcp   open       login
515/tcp   open       printer
3128/tcp  open       squid-http
6000/tcp  filtered   X11
6001/tcp  filtered   X11:1
6002/tcp  filtered   X11:2
6003/tcp  filtered   X11:3
6004/tcp  filtered   X11:4
6005/tcp  filtered   X11:5
6006/tcp  filtered   X11:6
6007/tcp  filtered   X11:7
6008/tcp  filtered   X11:8
6009/tcp  filtered   X11:9
6010/tcp  filtered   unknown
.....................................................
.....................................................
6200/tcp  filtered   unknown
6667/tcp  filtered   irc
10000/tcp open       snet-sensor-mgmt
32777/tcp open       sometimes-rpc17
32782/tcp open       unknown
32787/tcp open       sometimes-rpc27
32788/tcp open       unknown
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 16.993 days (since Wed Jul 17 18:04:32 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=4204978 (Good luck!)
IPID Sequence Generation: All zeros

# Nmap run completed at Sat Aug 03 17:55:01 2002 -- 1 IP address (1 host up) scanned in 179 seconds
```

Test 16: nmap Syn Stealth scan from internal network to administrative network shows ports 79/tcp, 110/tcp, 111/tcp, 513/tcp, 515/tcp, 561/tcp, 3000/tcp, 135/tcp and 139/tcp as open and not filtered. These ports must be closed. Ports 1241/tcp and 3001/tcp are temporarily open for nessus scans. Possible problem with outbound TCP rule base.

```
# nmap (V. 2.54BETA36) scan initiated Sat Aug 03 17:55:01 2002 as: nmap -v -g53 -sS -P0 -O -T 3 -p1-
65000 -oN C:\temp\test_16.txt 192.168.2.2
Interesting ports on server4.bandwidthco.com (192.168.2.2):
(The 64775 ports scanned but not shown below are in state: closed)
Port       State      Service
```

```
22/tcp    open       ssh
23/tcp    open       telnet
25/tcp    open       smtp
35/tcp    filtered   priv-print
36/tcp    filtered   unknown
37/tcp    filtered   time
38/tcp    filtered   rap
39/tcp    filtered   rlp
53/tcp    open       domain
79/tcp    open       finger
80/tcp    open       http
111/tcp   open       sunrpc
113/tcp   open       auth
445/tcp   filtered   microsoft-ds
513/tcp   open       login
515/tcp   open       printer
561/tcp   open       monitor
1241/tcp  open       msg
3000/tcp  open       ppp
3001/tcp  open       nessusd
6000/tcp  filtered   X11
6001/tcp  filtered   X11:1
6002/tcp  filtered   X11:2
6003/tcp  filtered   X11:3
6004/tcp  filtered   X11:4
6005/tcp  filtered   X11:5
6006/tcp  filtered   X11:6
6007/tcp  filtered   X11:7
6008/tcp  filtered   X11:8
6009/tcp  filtered   X11:9
6010/tcp  filtered   unknown
6011/tcp  filtered   unknown
……..
……..
6200/tcp  filtered   unknown
6667/tcp  filtered   irc
10000/tcp open       snet-sensor-mgmt
32769/tcp open       unknown
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 8.324 days (since Fri Jul 26 10:11:44 2002)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=3013212 (Good luck!)
IPID Sequence Generation: All zeros

# Nmap run completed at Sat Aug 03 17:58:02 2002 -- 1 IP address (1 host up) scanned in 181 seconds
```

Test 17: nmap UDP scan from internal network to internet shows only 53/udp, 123/udp, 514/udp, and 1233/udp ports as not filtered. This was determined by a perl script parsing the windump file of captured packets gathered from the internal network throughout the duration of the scan. UDP rules outbound appear to be very effective. Investigate UDP port 1233.

Test 18: nmap UDP scan from internal network to service network shows only 53/udp, 123/udp, and 514/udp ports as not filtered. This was determined by a perl script parsing the windump file of captured packets gathered from the internal network throughout the duration of the scan. UDP rules outbound appear to be very effective.

```
# nmap (V. 2.54BETA36) scan initiated Sat Aug 03 06:59:46 2002 as: nmap -v -g53 -sU -P0 -O -T 3 -p1-65000 -oN C:\temp\test_18.txt 192.168.1.2
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
Interesting ports on server3.bandwidthco.com (192.168.1.2):
(The 5200 ports scanned but not shown below are in state: closed)
Port       State       Service
1/udp      closed        tcpmux
2/udp      open        compressnet
3/udp      open        compressnet
……
……
64999/udp  closed       unknown
65000/udp  closed       unknown
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/3%Time=3D4C0557%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

# Nmap run completed at Sat Aug 03 09:31:19 2002 -- 1 IP address (1 host up) scanned in 9093 seconds
```

Test 19: nmap UDP scan from internal network to administrative network shows only 53/udp, 123/udp, 514/udp, and 517/udp ports as not filtered. This was determined by a perl script parsing the windump file of captured packets gathered from the internal network throughout the duration of the scan. UDP rules outbound appear to be very effective. Port udp/517 should be investigated to determine why it is not being filtered.

Spoofing Rules

All tests and firewall logs show spoofing rules are working correctly. All spoofed packets coming into the network and going out of the network were successfully identified and dropped.

Test 20

Nmap scan shows all spoofed addresses leaving the internal network were correctly identified and dropped:

```
# nmap (V. 2.54BETA36) scan initiated Thu Aug 01 18:45:49 2002 as: nmap -v -g53 -sS -P0 -e eth0 -S 10.0.0.10 -O -T 3 -p1-1024 -oN C:\temp\test_20.txt 192.168.0.3
```

Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 1024 scanned ports on (192.168.0.3) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D49E9FE%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed at Thu Aug 01 19:10:06 2002 -- 1 IP address (1 host up) scanned in 1457 seconds
nmap (V. 2.54BETA36) scan initiated Thu Aug 01 19:10:06 2002 as: nmap -v -g53 -sS -P0 -e eth0 -S 172.20.1.1 -O -T 3 -p1-1024 -oN C:\temp\test_20.txt --append_output 192.168.0.3
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 1024 scanned ports on (192.168.0.3) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D49EFAF%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed at Thu Aug 01 19:34:23 2002 -- 1 IP address (1 host up) scanned in 1457 seconds
nmap (V. 2.54BETA36) scan initiated Thu Aug 01 19:34:24 2002 as: nmap -v -g53 -sS -P0 -e eth0 -S 192.168.0.3 -O -T 3 -p1-1024 -oN C:\temp\test_20.txt --append_output 192.168.0.3
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 1024 scanned ports on (192.168.0.3) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D49F560%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed at Thu Aug 01 19:58:40 2002 -- 1 IP address (1 host up) scanned in 1456 seconds

Firewall logs also confirm spoofed packets were correctly identified and dropped:

Aug 1 18:47:24 server2 kernel: EXTERNAL SPOOF ATTEMPT IN=eth1 OUT=eth0 SRC=10.0.0.10 DST=192.168.0.3 LEN=40 TOS=0x00 PREC=0x00 TTL=50 ID=49477 PROTO=TCP SPT=53 DPT=103 WINDOW=1024 RES=0x00 SYN URGP=0

Aug 1 18:47:24 server2 kernel: EXTERNAL SPOOF ATTEMPT IN=eth1 OUT=eth0 SRC=172.20.1.1 DST=192.168.0.3 LEN=40 TOS=0x00 PREC=0x00 TTL=50 ID=7036 PROTO=TCP SPT=53 DPT=741 WINDOW=1024 RES=0x00 SYN URGP=0

Aug 1 19:29:07 server2 kernel: EXTERNAL SPOOF ATTEMPT IN=eth1 OUT=eth0 SRC=172.20.1.1 DST=192.168.90.1 LEN=40 TOS=0x00 PREC=0x00 TTL=42 ID=20338 PROTO=TCP SPT=57 DPT=544 WINDOW=1024 RES=0x00 SYN URGP=0

Test 21

Nmap scan shows all spoofed addresses entering the network from the Internet were correctly identified and dropped:

nmap (V. 2.54BETA36) scan initiated Thu Aug 01 18:29:32 2002 as: nmap -v -g53 -sS -P0 -e eth0 -S 10.0.0.10 -O -T 3 -p1-1024 -oN C:\temp\test_21.txt 172.17.0.2
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 1024 scanned ports on (172.17.0.2) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D49E94B%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed at Thu Aug 01 19:07:07 2002 -- 1 IP address (1 host up) scanned in 2255 seconds
nmap (V. 2.54BETA36) scan initiated Thu Aug 01 19:07:07 2002 as: nmap -v -g53 -sS -P0 -e eth0 -S 172.20.1.1 -O -T 3 -p1-1024 -oN C:\temp\test_21.txt --append_output 172.17.0.2
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 1024 scanned ports on (172.17.0.2) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D49F1B5%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed at Thu Aug 01 19:43:01 2002 -- 1 IP address (1 host up) scanned in 2154 seconds
nmap (V. 2.54BETA36) scan initiated Thu Aug 01 19:43:02 2002 as: nmap -v -g53 -sS -P0 -e eth0 -S 192.168.90.1 -O -T 3 -p1-1024 -oN C:\temp\test_21.txt --append_output 172.17.0.2
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
All 1024 scanned ports on (172.17.0.2) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess
TCP/IP fingerprint:
SInfo(V=2.54BETA36%P=i686-pc-windows-windows%D=8/1%Time=3D49FA49%O=-1%C=-1)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed at Thu Aug 01 20:19:37 2002 -- 1 IP address (1 host up) scanned in 2195 seconds

Firewall logs also confirm spoofed packets were correctly identified and dropped:

Aug 1 19:29:07 server2 kernel: INTERNAL SPOOF ATTEMPT IN=eth1 OUT=eth0 SRC=192.168.0.3 DST=10.0.0.1 LEN=40 TOS=0x00 PREC=0x00 TTL=42 ID=49238 PROTO=TCP SPT=57 DPT=961 WINDOW=1024 RES=0x00 SYN URGP=0

DNS Recursion & Zone Transfer Test

Tests 23 through 26 were all successful and confirmed no network Information leakage is occurring through the DNS:

C:\nslookup
Default Server: server1.giac.com
Addresses: 192.168.0.2

Request external DNS server:

> server server3.giac.com
Default Server: server3.giac.com
Addresses: 192.168.1.2, 172.16.0.2

Request address for external web server:

> www.giac.com
Server: server3.giac.com
Addresses: 172.16.0.1

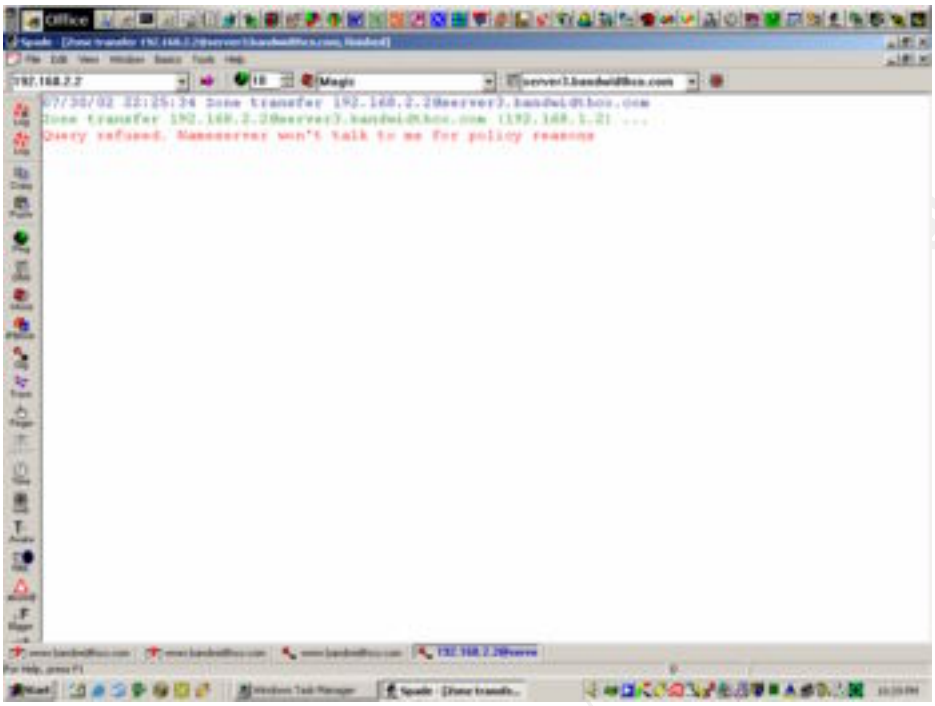
Gives address of external interface to firewall. Now, request address for internal web server:

> www1.giac.com
Server: server3.giac.com
Non existent host/domain

Now we request zone information:

> ls –d giac.com
can't list domain giac.com

Additionally, Sam Spade was unable to provoke the external DNS server into performing a zone transfer:



Mail Relay Test

The mail relay test performed well and showed the sendmail relay unwilling to handle or forward mail not TO or FROM its configured domain as identified in the /etc/mail/access.db file:

telnet 192.168.1.2 25
220 server3.giac.com ESMTP Sendmail 8.11.3/8.11.3/SuSE Linux 8.11.1
Mon, 29 July 2002 20:57:14

EHLO mail
250 Hello Workstation1.giac.com [192.168.0.3] pleased to meet you

MAIL FROM: bob@microsoft.com
220 Cannot resolve domain

telnet 192.168.1.2 25
220 server3.giac.com ESMTP Sendmail 8.11.3/8.11.3/SuSE Linux 8.11.1
Mon, 29 July 2002 20:59:11

EHLO mail
250 server3.giac.com Hello Workstation1.giac.com [192.168.0.3] pleased to meet you

MAIL FROM: root@giac.com
250 2.1.0 root@giac.com . . . sender ok

RCPT TO: root@bandwidthco.com

250 2.1.0 root@giac.com . . . sender ok
DATA
This is a test.
.

250 message accepted
QUIT

RECOMMENDATIONS

Based on extensive testing of the GIAC Enterprises firewall and related network segments and services, Bandwidthco Security Services recommends the following actions be taken:

1. Disable ports 37/tcp, 111/tcp, 513/tcp, 515/tcp, 3000/tcp, 6000/tcp, printer/tcp, and 111/udp on firewall machine.

2. Review eth0 outbound TCP rule base. Ports 79/tcp, 110/tcp, 111/tcp, 513/tcp, 515/tcp, 561/tcp, 3000/tcp, 135/tcp and 139/tcp are not being filtered.

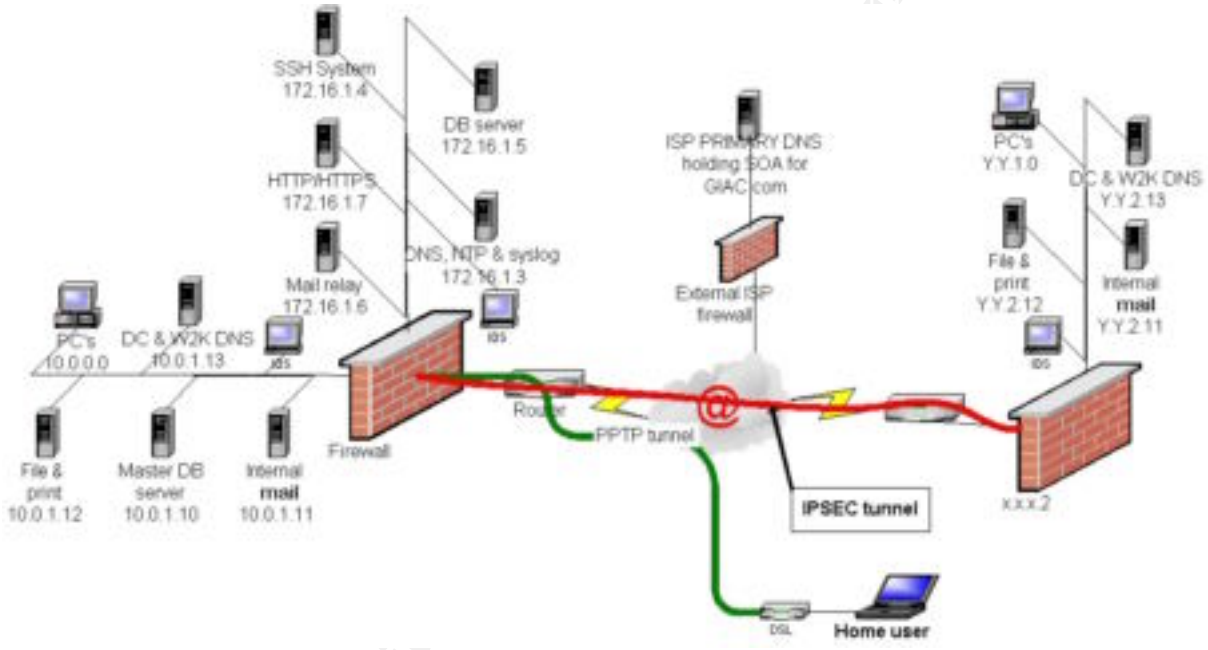
3. Ports 517/udp and 1233/udp eth0 outbound are not filtered. Investigate reason these ports are not filtered.

4. The firewall rule base is extensive. Additional test are recommended to determine if certain rules contained within can be eliminated.

SECTION 4 – Design Under Fire

INTRODUCTION

I have selected the firewall design of Peter Vestergaard from April 17, 2002 as the subject of my attack (http://www.giac.org/GCFW.php). My choice was not based upon what I see as a vulnerable design. Quite contrary in fact. Peter has done an excellent job, and he was detailed and thorough in all respects. Rather, I selected Peter's design because of similarities to mine, both in terms of architecture and software. I saw this as an opportunity to better understand my own design. Peter's network is shown below.



Peter selected the Linux "netfilter/iptables" for his primary firewall software. He also uses SSH extensively for remote access to his service network. My primary focus will be potential or real exploitable weaknesses in these systems.

ATTACKING THE FIREWALL

Research on the netfilter/iptables Linux 2.4.X and 2.5 firewall subsystem revealed few documented weaknesses and vulnerabilities. Because of this, one must assume the coding is very solid in general. However, two weaknesses were identified that are worth discussing. We will also attempt to get some hacking or cracking mileage out of them.

MAC Module Vulnerability

The first vulnerability we will investigate was discovered by Chris Wilson on October 8, 2001, and it effects iptables versions prior to 1.2.3. The flaw may allow remote users to pass packets through the firewall under certain situations. Advisories, in addition to those issued by netfilter.org, for this flaw were issued by SecurityTracker at http://securitytracker.com/alerts/2001/Oct/1002520.html, and the Neohapsis Archives at http://archives.neohapsis.com/archives/bugtraq/2001-10/0057.html.

The flaw resides in the MAC extension module of iptables. It is designed to match packets based on their Ethernet MAC address. The module is intended to protect against internal or directly connected users from spoofing or changing their IP address. The report states that the MAC module does not correctly match very small packets. The end result is, if MAC matching/filtering is used, a remote user may be able to pass small packets through the firewall.

Chris Wilson was able to reproduce the problem using two machines (victim and attacker) as demonstrated below.

1. On "victim", flush the INPUT chain and insert a rule to DROP a source MAC address:

   iptables -P INPUT ACCEPT
   iptables -F INPUT
   iptables -I INPUT -p icmp -m mac --mac-source AT:TA:CK:ER:00:00 -j DROP

2. View the effects of the rule:

   iptables -L INPUT -v
   Chain INPUT (policy ACCEPT xxxx packets, xxxxxxx bytes) pkts bytes target prot opt in out source destination 0 0 DROP icmp -- any any anywhere anywhere MAC AT:TA:CK:ER:00:00

   At this point, both the packet and byte counters are still zero.

3. Now ping "victim" from "attacker" with one 8-byte packet:

   ping -s 8 -c 1 victim
   PING victim (xx.xx.xx.xx) from xx.xx.xx.xx : 8(36) bytes of data
   --- xx.xx.xx.xx ping statistics ---
   1 packets transmitted, 0 packets received, 100% packet loss

   We note that the ICMP packets were correctly dropped.

4. From "victim", list the rule statistics once again::
   iptables -L INPUT -v

Chain INPUT (policy ACCEPT 231 packets, 39475 bytes) pkts bytes target prot opt in out source destination 1 36 DROP icmp -- any any anywhere anywhere MAC 00:03:47:87:BA:C5

The packet counter now correctly shows 1 packet

5. Ping "victim" again, this time with one 4-byte packet:

ping -s 4 -c 1 victim
PING Victim (xx.xx.xx.xx) from xx.xx.xx.xx : 4(32) bytes of data.
12 bytes from xx.xx.xx.xx: icmp_seq=0 ttl=255

--- xx.xx.xx.xx ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss

Note that the smaller 4-byte packet was not dripped, but was, in fact, replied to.

6. Time to check the statistics again:

iptables -L INPUT -v
Chain INPUT (policy ACCEPT 231 packets, 39475 bytes) pkts bytes target prot opt in out source destination 1 32 DROP icmp -- any any anywhere anywhere MAC AT:TA:CK:ER:00:00

We should now see that the packet counters have not incremented, as the small packet was not matched, nor was it dropped by the firewall system.

The obvious significance with this flaw is that an attacker could use very small packets to bypass or circumvent the firewall. Unfortunately, Peter did not employ the MAC module, so we cannot use this known problem against his design. However, the second flaw we discuss below, can be used against his design in an attempt to attack and penetrate his firewall.

ICMP Information Leakage

Philipe Biondi on May 8, 2002, and effects iptables versions prior to 1.2.6a discovered the second vulnerability we will investigate. The vulnerability may allow remote users pass packets through the firewall in certain situations. Advisories for this flaw, in addition to those issued by netfilter.org, were issued by Internet Security Systems at http://www.iss.net/security_center/static/9043.php, and by SecurityFocusOnline at http://online.securityfocus.com/archive/1/271530/2002-05-06/2002-05-12/0. The alert issued by Netfilter.org can be found at http://www.netfilter.org/security/2002-04-02-icmp-dnat.html.

Philipe Biondi describes the problem as follows:

> When a NAT rule applies to the first packet of a connection and that packet later causes the system to generate an ICMP error message, the ICMP error message is sent out with translated addresses included. This address information incorrectly gives the IP address to which the connection would have been forwarded if the ICMP error message was not generated, which exposes information about the netfilter configuration (which ports are being translated) and about the network topology (which address the ports are being forwarded to). Also, the incorrect ICMP packets may be dropped by other intervening stateful firewalls as malformed packets.

This basically means that if netfilter/iptables is being used to DNAT packets to an internal host, and the internal host responds with an ICMP error message (which includes the IP packet as sent to the internal host), the sender, or an attacker, will be able to determine that the packet was address or port redirected, and they will also be able to discover the "real" internal address.

Since Peter has used DNAT port redirection in his firewall design, we may have found an opening into his network. Here are the applicable rules he used which may be exploited:

```
echo -n "Setting up DMZ Portforwarding   :"

# Rules for the portforwarding to the servers on the DMZ
# Portforwarding from wan interface tcp port 25 to mail relay server port 25 on service network
$IPTABLES -t nat -A PREROUTING -i $WAN_INT -p tcp -d $WAN_IP --dport 25 -j DNAT --to-destination
$EXT_MAILSERVER:25
$IPTABLES -A forwardfromwantodmz -p tcp --destination $EXT_MAILSERVER --dport 25 -j ACCEPT

# Portforwarding from wan interface tcp port 80 to web server port 80 on service network
$IPTABLES -t nat -A PREROUTING -i $WAN_INT -p tcp -d $WAN_IP --dport 80 -j DNAT --to-destination
$EXT_WEBSERVER:80
$IPTABLES -A forwardfromwantodmz -p tcp --destination $EXT_WEBSERVER --dport 80 -j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $WAN_INT -p tcp -d $WAN_IP --dport 443 -j DNAT --to-destination
$EXT_WEBSERVER:443
$IPTABLES -A forwardfromwantodmz -p tcp --destination $EXT_WEBSERVER --dport 443 -j ACCEPT

# Portforwarding from wan interface tcp port 22 to SSH system port 22 on service network
$IPTABLES -t nat -A PREROUTING -i $WAN_INT -p tcp -d $WAN_IP --dport 22 -j DNAT --to-destination
$EXT_SSH_SERVER:22
$IPTABLES -A forwardfromwantodmz -p tcp --destination $EXT_SSH_SERVER --dport 22 -j ACCEPT

# Portforwarding from wan interface udp port 123 to NTP server on service network
$IPTABLES -t nat -A PREROUTING -i $WAN_INT -p udp -d $WAN_IP --dport 123 -j DNAT --to-destination
$EXT_NTPSERVER:123
$IPTABLES -A forwardfromwantodmz -p udp --destination $EXT_NTPSERVER --source $BORDERROUTER --
dport 123 -j ACCEPT

# Portforwarding from wan interface udp port 514 to Syslog server on service
$IPTABLES -t nat -A PREROUTING -i $WAN_INT -p udp -d $WAN_IP --dport 514 -j DNAT --to-destination
$EXT_SYSLOGSERVER:514
$IPTABLES -A forwardfromwantodmz -p udp --destination $EXT_SYSLOGSERVER --source
$BORDERROUTER --dport 514 -j ACCEPT

echo "Done"
```

We will check the validity of this potential exploit by running heavy traffic scans on ports 22, 25, 80, 123, and 514 repeatedly into the firewall with nmap while capturing any return ICMP traffic with tcpdump. From this, we should see if any usable information can be obtained about the addressing structure inside the network.

The scans were run against iptables version 1.2.1 with ports 22, 25, 80, 123, and 514 DNAT redirected to the service network servers. Firewall logs show that the redirection was successfully completed, and all packets reached their respective servers. However, all the scans produced negative results for useful information. Tcpdump, set to capture any ICMP return traffic, collected no ICMP packets from within the network. Thus, the vulnerability as described above, could not be reproduced in actual application.

THE DoS ATTACK

Peter selected the Cisco 1720 router for his border router. This was an appropriate selection for him in terms of its native VPN support. However, the 1720 is not particularly robust in its ability to support a large volume of traffic, and therefore should be quite susceptible to a DoS attack. For this attack, we will use DNS cache poisoning to compromise and redirect the 50+ cable modem or DSL systems traffic to Peter's external web site. As this traffic must be handled by the 1720 router, the intent will be to overwhelm the router with traffic beyond its handling capabilities, effectively rendering it useless.

This exploit will be based on the BIND vulnerability prior to version 8.1.1 that allows a hostile user or attacker to cache defective DNS information on a victim DNS server by sending "answer" records in a DNS message datagram. This BIND vulnerability is documented by Cert Advisory CA-1997-2 "BIND – The Internet Named Daemon", revised May 26, 1998. This CERT advisory can be found at http://www.cert.org/advisories/CA-1997-22.html.

Before we initiate this attack, let's take a quick look at the DNS protocol structure to better understand how we can affect this exploit. The figure on the following page illustrates the format and layout of a DNS datagram. Byte 13 and byte 14 will be our focus. From this, we can see that the DNS protocol by design permits both "questions" and "answers" to be rendered within the same message. It is this DNS property, coupled with the identified BIND vulnerability that will enable us to poison the cache of a name server.

STEP 1: Obtain Address of Victim Web Site

The first thing we need to do is obtain the IP address of GIAC Enterprises web site as maintained by their ISP's primary name server. So we perform a simple query using nslookup:

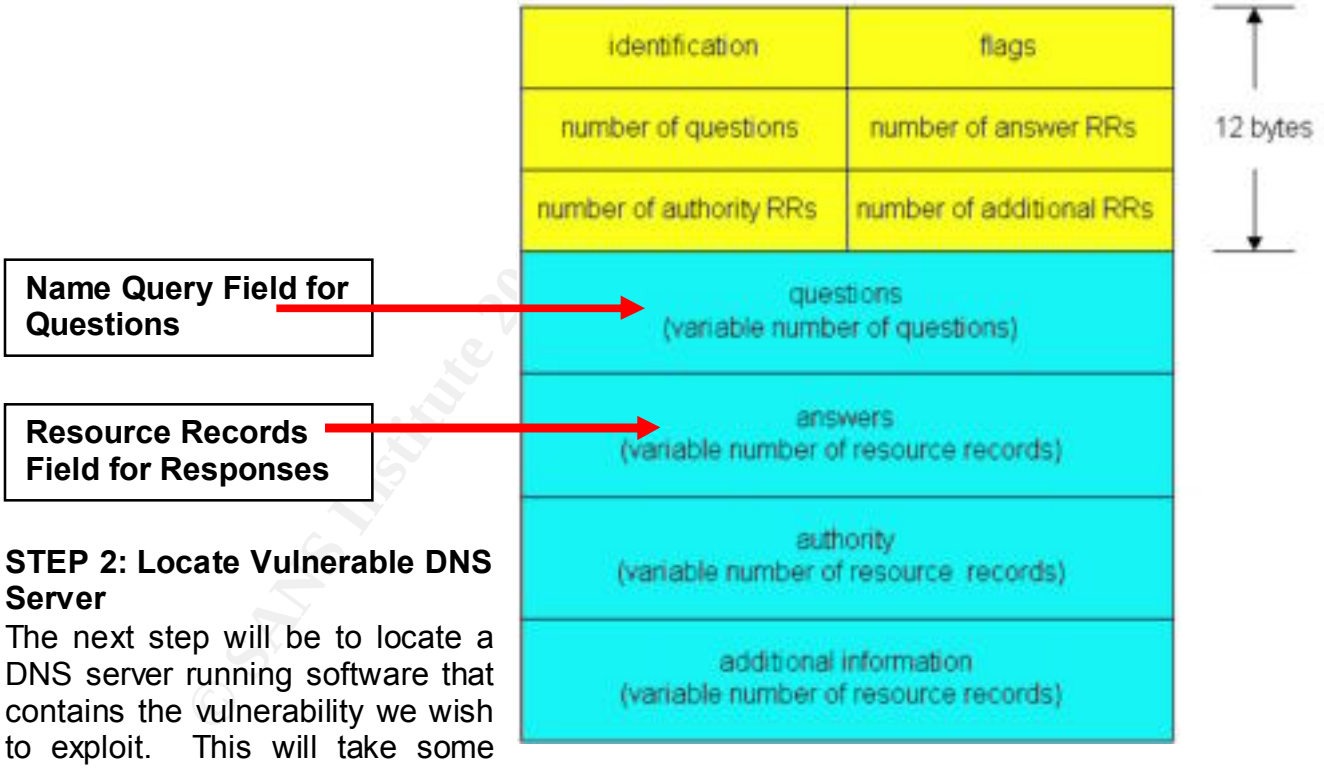
host.hacker.com > nslookup
Default Server: dns.hacker.com
Address: 192.168.0.3

> www.giac.com
Server: ns.isp.com
Address: 1.2.3.4

Name: www.giac.com
Address: 1.1.1.2

So, we have obtained the address of 1.1.1.2 for the GIAC Enterprises web site. Since Peter is using port DNAT port redirection on his primary firewall, the resolved address should theoretically be a "routable" IP address of the external interface of the firewall.

NOTE: In reality, it is not real clear from Peter's design how the actual IP address of the web site could be resolved. This may have been an oversight on his part. Because of this, I am making some assumptions.



STEP 2: Locate Vulnerable DNS Server

The next step will be to locate a DNS server running software that contains the vulnerability we wish to exploit. This will take some plain old trial-and-error footwork. Since the goal here is to compromise 50+ cable modem or DSL systems, we will start investigating the name servers for ISP that specialize in cable modem or DSL service. Again, we use nslookup and explore until we find a potentially vulnerable name server. Let's check out the ISP cable.net first:

```
host.hacker.com> nslookup
Default Server:  dns.hacker.com
Address: 192.168.0.3

> set q=ns
> cable.net
Server: ns1.cable.net
Address: 1.2.3.4

cable.net  nameserver = ns1.cable.net
ns1.cable.net internet address = 1.2.3.5
```

OK. We now have the IP address of the name server for cable.net. We can use the "dig" utility to, hopefully, determine what type and version of software this name server is running:

```
dig@cable.net version.bind txt chaos

; <<>> DiG 8.2 <<>> @cable.net version.bind txt chaos
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;;     version.bind, type = TXT, class = CHAOS
;;
;; ANSWER SECTION:
VERSION.BIND.      OS CHAOS TXT     "4.9.7-REL"
```

Bingo. Seems as though we hit pay dirt. Cable.net is using a vulnerable version of BIND.

STEP 3: Poison the Cache of cable.net

We are going to make another assumption here. We are going to assume that a large number of cable.net's users have their browser's default start page set to msn.com, which ns1.cable.net will attempt to resolve when the user logs on and opens their browser. Based on that assumption, we will want to implant false cache data in ns1.cable.net, pointing msn.com to GIAC Enterprises web site.

The tool "nemesis" will be used to perform our "cache poisoning". Nemesis is a great little UNIX command line packet injection tool, originally written by Mark Grimes. We will use "nemesis-dns" for our evil exploit as this sub-utility will allow us to send a DNS query to ns1.cable.net that also contains a resource record answer in the message. If cable.net's DNS software has not been patched, then it will not first verify that the injected record is a "response" and not a "query".

To make this work, we will place our corrupted record in a file, and call the file from nemesis-dns:

```
; file corrupt.txt
www.msn.com              IN     A      1.1.1.2

nemesis-dns –v –S 1.2.3.4 –D 1.2.3.5 –q 1 –W 1 –P corrupt.txt

DNS Packet Injection -=- The NEMESIS Project 1.32
……..
……..

Wrote 40 bytes

DNS Packet injected
```

The dirty work is now complete. If our attack was successful, the following should occur:

- Cable.net's name server should contain a host resource record for msn.com pointing to GIAC Enterprises web site.
- When cable.net users log on and open their browsers, the browser resolver should request the IP address for msn.com from the listed name server of cable.net.
- Cable.net's name server should first check its DNS cache for an entry for msn.com.
- When it finds the injected entry in its "poisoned cache", it will return the bogus IP address back to the requesting browser.
- The browser will then be directed to GIAC Enterprises web site.
- 50 cable modem users being redirected to GIAC Enterprises within the same time frame should be sufficient to overwhelm the Cisco 1720 router and possibly render it ineffective.

Prevention

The following actions may be taken to help secure a DNS configuration as well as help prevent DNS cache poisoning:

- Upgrade to a version of BIND 8.1.1 or greater, preferably to version 9.3.
- Restrict and authorize zone transfers.
- Restrict dynamic updates.
- Restrict recursive queries.

ATTACKING THE INTERNAL NETWORK

Peter generally described his use of SSH as such: "Suppliers and business partners do not differ much in their access requirements. The suppliers create new fortunes – and must upload these to the SSH system. The partners buy fortunes as described above. When they translate fortunes – these must as well be uploaded to the SSH system." Lets see what we can do with this.

Secure Shell and OpenSSH have been beset with several "authentication" related vulnerabilities over the past two years. This has somewhat tarnished its image as an impeccable replacement for the UNIX "r" utilities and telnet. Recently, in fact, the July 17, 2002, CERT Advisory CA-2002-18 "OpenSSH Vulnerabilities in Challenge Response Handling" at http://www.cert.org/advisories/CA-2002-18.html advised of two new related vulnerabilities in the handling of challenge responses in OpenSSH 2.9.9 through 3.3. We also have CERT Vulnerability Note VU#684820 at http://www.kb.cert.org/vuls/id/684820 that describes a new potential man-in-the-middle (MITM) vulnerability in the beleaguered SSH-1 that permits client authentication to be forwarded by a malicious server to another server.

However, SSH's publicly documented problems began in late 2000 when Dug Song released a new version of his *dsniff* program. "dsniff", freely available at http://www.monkey.org/~dugsong/dsniff, is a sophisticated set of utilities that allows monitoring and redirection of network traffic so it can be analyzed, or, depending on what ones motives may be, "hijacked". While most packet sniffers can monitor network data, very few are capable or rerouting traffic or analyzing encrypted data. For example, with the "dsniff" *dnsspoof* utility, one can forge DNS replies and redirect DNS names to arbitrary IP addresses that you control. The new release of dsniff included a tool called *sshmitm* that performs a man-in-the-middle attack (MITM) on the SSH-1 protocol. What we are going to do is penetrate Peter's firewall and perimeter network with "dniff" by victimizing one of GIAC Enterprises suppliers or partners as they attempt to connect to the OpenSSH server on Peter's network.

STEP 1: Select Victim

The machine running both "dnsspoof" and "sshmitm" should be situated between the SSH client and SSH server, and on the same network segment as the SSH client. So, we must first do some preliminary social engineering and detective work to gather information about the personal habits of GIAC Enterprises suppliers and partners. Ideally, we would identify one that frequently plugs into a wireless network, or uses a public cyber-café or conferencing terminal room. Once we have identified our victim, we will wait for the opportunity to plug-in with him.

STEP 2: Spoof DNS
When our GIAC Enterprises victim logs into the GIAC Enterprises SSH server, we want to direct them to our attack machine. This is where we will put "dsnspoof" to use. On the attack machine, we configure our /etc/dnsspoof.host file as such:

1.2.3.4 *.giac.com

with *.giac.com pointing to the IP address of the attack machine. We then launch the "dsnspoof" utility:

dnsspoof -i eth0 -f /etc/dnsspoof.host

Now, when our victim's resolver queries giac.com, they will be directed to the attack machine.

STEP 3: Intercept Authentication
Next we start up “sshmitm”:

sshmitm -d -l -p 22 sshserver.giac.com

The “sshmitm” program establishes a connection to the actual SSH server as if it were a normal SSH client. When the victim client machine connects to the server, the machine running "sshmitm" intercepts the packets. The sshmitm process pretends to be the actual server. It then takes the data that it receives from the real client, which is available in clear text, re-encrypts it, and sends it to the real ssh server.

Because the client sends packets to the server without knowing they are being intercepted, and the login process appears to progress normally, the victim has no concept that a hack is taking place, and that sensitive data is about to be stolen. However, every packet the client sends is readable (and modifiable) by the sshmitm program. “sshmitm” follows the SSH protocol specification by generating the host key, agreeing on cryptographic algorithms and keys, and eventually asking the client for the password. The game is then over and the hack is complete.

All of this activity does not take place without some signs of foul play however, and the “sshmitm” program makes several assumptions about normal human behavior. For instance, when the victim logs in and the “sshmitm” key is provided, they are likely to be greeted with the following message:

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY! Someone could be eavesdropping on you right now (man-in-the-middle attack)! It is also possible that the RSA1 host key has just been changed. The fingerprint for the RSA1 key sent by the remote host is 8:45:0b:e7:0b:bb:70:ac:8e:94:93:7f:33:44:10:c6. Please contact your system administrator. Add correct host key in /home/bri/.ssh/known_hosts to get rid of this message. Offending key in /home/bri/.ssh/known_hosts:17 RSA1 host key has changed and you have requested strict checking.

"sshmitm" works on the assumption that most users (victims) will ignore this warning and answer "Yes" when posed the following question by the SSH server:

The authenticity of host 'example.com (192.168.12.12)' can't be established.
RSA1 key fingerprint is 44:a9:18:b9:81:aa:c6:97:42:cd:66:44:ae:79:32:6a.
Are you sure you want to continue connecting (yes/no)?

Prevention

There are several ways to avoid the SSH Man (Monkey)-in-the-Middle attack:

1. Upgrade to SSH versions 2.x or 3.x "sshmitm" was written for SSH-1 and will not be modified for newer version, at least by Mr. Dug Song.

2. Use SSH identities for authentication. This form of authentication uses public key cryptography and the private key is never sent across the network.

3. Educate users to the dangers of the attack and how to recognize when the attack is taking place.

4. Implement lower level security protocols such as DNSSEC. This will prevent attackers from spoofing DNS replies.

5. Other options include using IPSec and making it mandatory for workstations and servers, with a strong authentication system it would be much more difficult for attackers to spoof traffic or sniff it.

FINAL THOUGHTS

My regards to the Sans Institute. This assignment has not only been a pleasure to do, but a great learning experience. Oh yes, it was tough, and it was hard work. But it forced me to explore and learn the intimate details. To all of you, a job well done.

APPENDIXES

Appendix A – Script router_security.conf

```
! ##################################################################
! Router Security Configuration Script
! Script Name: router_security.conf
! Revised: July 17, 2002
! Written By:  Mark E. Donaldson
! ##################################################################
!
! ##################################################################
! Set Passwords
! ##################################################################
! start in configuration mode
config t
!
! set enable password
enable password xxxxxxxxx
! set enable secret password:
enable secret xxxxxxxxx
! set vty password
line vty 0 4
login
password xxxxxxxxx
! set console and auxiliary passwords
line con 0
login
password xxxxxxxxx
line aux 0
login
password xxxxxxxxx
exit
!
! ##################################################################
! Set Router Hostname & DNS
! ##################################################################
! start in configuration mode
config t
!
hostname Router1
exit
ip domain-lookup
! set up a host table
config t
ip host Router1 172.16.0.10
ip host server2 192.168.0.1
ip host server1 192.168.0.2
ip host server3 192.168.1.2
```

```
ip host server4 192.168.2.2
!
! show host table
exit
show hosts
!
! ################################################################
! Set IP addressing
! ################################################################
! start in configuration mode
config t
! eth0
int eth0
Ip address 172.16.0.10 255.255.0.0
no shut
description GIAC Enterprises Network Gateway
! eth1
int eth1
Ip address 172.17.0.10 255.255.0.0
no shut
description GIAC Internet Gateway
exit
!
! ################################################################
! Set Banners
! ################################################################
! start in configuration mode
config t
!
! set login banner
banner login #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
! set MOTD banner:
banner MOTD #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
! set incoming terminal line banner:
banner incoming #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
! set executive process creation banner:
banner exec #
WARNING:  Access To This Network is Restricted.
You're Actions are being Monitored and Logged!!!!!
#
exit
!
! ################################################################
! Harden Router
```

```
! ################################################################
! start in configuration mode
config t
!
! force display passwords to be encrypted
service password encryption
! disable CDP (Cisco Discovery Protocol)
no cdp run
! disable finger service
no service finger
! disable echo, discard, chargen, and daytime services
no service udp-small-servers
no service tcp-small-servers
! drop source routing
no ip source-route
! disable servers
no ip bootp server
no ip http server
! disable SNMP
no snmp
!
! Add these to external interface of screening router
! run from interface config mode
int eth1
! don't send icmp for denied items in access-list
no ip unreachables
! prevent ICMP redirection
no ip redirects
! prevent layer 2 broadcast mapping and smurf amplification
no ip direct-broadcast
! disable proxy-arp
no ip proxy-arp
! disable NTP if not using external time source
! no enable ntp
exit
!
! ################################################################
! Set Default & Static Routing
! ################################################################
! We want to avoid dynamic routing protocols if at all possible
! start in configuration mode
config t
!
! disable RIP
no router rip
! disable OSPF
no ospf
!
! set default route
ip route 0.0.0.0 0.0.0.0 172.16.0.1
ip classless
!
! set static routes
```

```
! Syntax: ip route route mask next hop
! Example: 172.16.0.0 255.255.0.0 192.168.0.1
ip route 192.168.0.0 255.255.255.0 172.16.0.1
ip route 192.168.1.0 255.255.255.0 172.16.0.1
ip route 192.168.2.0 255.255.255.0 172.16.0.1
!
! ################################################################
! Configure Logging
! ################################################################
! configure loop back interface
! start in configuration mode
config t
interface loopback 0
ip address 192.168.30.1 255.255.255.255
exit
!
! define the NTP server
! start in configuration mode
config t
!
! set clock for Pacific Time Zone (offset 8 hours from GMT) and PDST
clock timezone PST -8
clock summer-time zone recurring
! set NTP server
ntp update-calendar
ntp server 192.168.1.2
ntp source loopback 0
!
! Now set the system clock:
! Example: #clock set 21:51:20 7 April 2002
! configure the timestamp options for system logging and debug messages
service timestamps log datetime localtime
service timestamps debug datetime localtime
exit
! check NTP associations
sh ntp associations
!
! disable console and buffered logging (or any form of logging)
! start in configuration mode
config t
no logging console
no logging buffered
! enable monitor logging
logging monitor debugging
!
! start in configuration mode
config t
!
! logging facility local1
! logging source-interface Loopback 0
! Log everything to syslog
! no logging console
no logging buffered
```

```
logging 192.168.2.2
logging trap debug
logging monitor debugging
logging console emergencies
!
! ################################################################
! Set Performance Related Items
! ################################################################
! start in configuration mode
config t
!
ip tcp path-mtu-discovery
ip tcp selective-ack
exit
!
! ################################################################
! Show, Save, & Backup Configuration
! ################################################################
! start in exec mode
sh run
show start
copy run start
! copy start tftp
! ################################################################
```

Appendix B – Script access-list vty.txt

```
! ################################################################
! Secure vty (telnet) and aux ports
! Script Name: access-list vty.txt
! Revised: July 17, 2002
! Written By:  Mark E. Donaldson
! ################################################################
!
! Standard access lists
! Add access-lists:                access-list 1 & access-list 2
!
! Allow only specific hosts to telnet into router protected network
access-list 1 permit 192.168.0.0 0.0.0.255
! use if SNAT working on firewall
access-list 1 permit 172.16.0.0 0.15.255.255
line vty 0 4
access-class 1 in
login
!
! Block all access to aux port
access-list 2 deny 0.0.0.0 255.255.255.255
line aux 0
access-class 2 in
login
transport input all
```

```
exit

! ##################################################################
```

Appendix C – Applied Router Security Policy Scripts

```
! ##################################################################
! Named Access List Inbound Screen
! Script Name: access-list extended inbound eth1.txt
! Revised: July 17, 2002
! Written By:  Mark E. Donaldson
! ##################################################################
!
! ##################################################################
!
ip access-list extended inbound-eth1
!
! ##################################################################
! Router Lockdown & Management
! ##################################################################
!
! ##################################################################
! Permit private address used for testing purposes
! Deny for production environment but insert actual routable IP addresses
! REMEMBER:  Using outbound SNAT to firewall external interface 172.16.0.1
! ##################################################################
! permit packets addresses to external firewall and VPN gateway ports
permit ip any host 172.16.0.1 log
permit ip any host 172.16.0.2 log
!
! ##################################################################
! Deny rfc 1918 addresses
! ##################################################################
deny ip 192.168.0.0 0.0.255.255 any log
! remove for testing
! deny ip 172.16.0.0 0.15.255.255 any log
deny ip 10.0.0.0 0.255.255.255 any log
!
! ##################################################################
! Deny packets with localhost, broadcast and multicast addresses
! ##################################################################
deny ip 127.0.0.0 0.255.255.255 any log
deny ip 255.0.0.0 0.255.255.255 any log
deny ip 224.0.0.0 7.255.255.255 any log
!
! ##################################################################
! Deny packets without ip address
! ##################################################################
deny ip host 0.0.0.0 any log
!
! ##################################################################
! Spoof Prevention (Ingress & Egress Filtering)
```

```
! ###################################################################
! Prevent spoofing
! The edge router internal interface should only accept traffic with source
! addresses belonging to the internal network. The router external interface
! should only accept traffic with source addresses other than the internal network
!
! Concern is only with public (routable) addresses as any private addresses
! have already been dropped earlier.  In our case we are only
! concerned with external firewall interface and external VPN Gateway interface
! and both router ports.
!
deny ip host 172.16.0.1 any log
deny ip host 172.16.0.2 any log
!
! router interface ip addresses
!
deny ip host 172.16.0.10 any log
deny ip host 172.17.0.10 any log
!
! ###################################################################
! Deny NetBIOS proprietary protocols from coming in
! ###################################################################
! we don't particularly want to log these
deny tcp any any range 135 139
deny udp any any range 135 139
deny tcp any any eq 445 log
!
! ###################################################################
! Deny X-Windows Traffic
! ###################################################################
deny tcp any any range 6000 6255 log
!
! ###################################################################
! Deny TFTP, Syslog, and SNMP Traffic
! ###################################################################
deny udp any any eq 69 log
deny udp any any eq 514 log
deny udp any any range 161 162 log
!
! ###################################################################
! Deny NFS & Sun RPC
! ###################################################################
deny tcp any any eq 111 log
deny udp any any eq 111 log
deny tcp any any eq 2049 log
deny udp any any eq 2049 log
!
! ###################################################################
! Reflexive Protocol Control
! ###################################################################
! reflexive list must have a matched in-out pair on each interface
!
evaluate tcp-filter
```

```
evaluate udp-filter

! allow host on internet to talk to mail relay
permit tcp any host 172.16.0.1 eq 25 reflect smtp-filter
! allow host on internet to talk to external DNS server
permit udp any host 172.16.0.1 eq 53 reflect dns-filter
permit tcp any host 172.16.0.1 eq 53 reflect dns-filter
!
!#################################################################
! Permit Traffic To Firewall External Interface and VPM Gateway
! ################################################################
! permit packets addressed to external firewall and VPN gateway ports
permit ip any host 172.16.0.1 log
permit ip any host 172.16.0.2 log
!
! ################################################################
! DNS Traffic Static
! ################################################################
!
! DNS traffic DNAT forwarded from firewall
permit tcp any host 172.16.0.1 eq 53 log
permit udp any host 172.16.0.1 eq 53 log
!
! ################################################################
! NTP Traffic Static
! ################################################################
!
! NTP traffic DNAT forwarded from firewall
permit udp any host 172.16.0.1 eq 123 log
!
! ################################################################
! SMTP Traffic Static
! ################################################################
!
! SMTP traffic DNAT forwarded from firewall
permit tcp any host 172.16.0.1 eq 25 log
!
! ################################################################
! NNTP Traffic Static
! ################################################################
!
! NNTP traffic DNAT forwarded from firewall
permit tcp any host 172.16.0.1 eq nntp log
!
! ################################################################
! ICMP Control Static
! ################################################################
!
! Allow only specific ICMP
! Again:  because we are using SNAT only firewall and VPN gateway
! external port IP addresses are of concern for all host behind the router
!
! net-unreachable
```

```
permit icmp any host 172.16.0.1 3 0
permit icmp any host 172.16.0.2 3 0
! host-unreachable
permit icmp any host 172.16.0.1 3 1
permit icmp any host 172.16.0.2 3 1
! port-unreachable
permit icmp any host 172.16.0.1 3 3
permit icmp any host 172.16.0.2 3 3
! packet-too-big (fragment)
permit icmp any host 172.16.0.1 3 4
permit icmp any host 172.16.0.2 3 4
! administratively-prohibited
permit icmp any host 172.16.0.1 3 13
permit icmp any host 172.16.0.2 3 13
! source-quench
permit icmp any host 172.16.0.1 4
permit icmp any host 172.16.0.2 4
! ttl-exceeded
permit icmp any host 172.16.0.1 10
permit icmp any host 172.16.0.2 10
!
! ################################################################
! Default
! ################################################################
! Permit and Log everything that does not meet the above rules.
!
permit ip any any log
!
! HITS CISCO IOS IMPLICIT DENY
! ################################################################

! ################################################################
! Named Access List Inbound Screen
! Script Name: access-list extended outbound eth1.txt
! Revised: July 17, 2002
! Written By:  Mark E. Donaldson
! ################################################################
! ################################################################
!
ip access-list extended outbound-eth1
!
! ! ################################################################
! Reflexive Protocol Control
! ################################################################
! reflexive list must have a matched in-out pair on each interface
! allow unrestricted access to internet from internal network
!
permit tcp host 172.16.0.1 any reflect tcp-filter
permit udp host 172.16.0.1 any reflect udp-filter
evaluate smtp-filter
evaluate dns-filter
!
! ################################################################
```

```
! Default
! ################################################################
! Permit and Log everything that does not meet the above rules.
!
permit ip any any log
!
! HITS CISCO IOS IMPLICIT DENY
! ################################################################

! ################################################################
! Named Access List Outbound Screen
! Script Name: access-list extended outbound eth0.txt
! Revised: July 17, 2002
! Written By:  Mark E. Donaldson
! ################################################################
!
!################################################################
!
ip access-list extended outbound-eth0
!
! ################################################################
! Router Lockdown & Management
! ################################################################
!
permit ip host 172.168.0.1 host 172.16.0.10 log
! use following if SNAT not up
permit ip host 192.168.0.3 host 172.16.0.10
!
! ################################################################
! Connectivity Checking
! ################################################################
! allow internal users to ping both sides of router
permit icmp host 172.16.0.1 host 172.16.0.10 8
permit icmp host 172.16.0.1 host 172.17.0.10 8
! use following if SNAT not up
permit ip 192.168.0.0 0.0.0.255 host 172.16.0.10
permit ip 192.168.0.0 0.0.0.255 host 172.16.0.10
!
! ################################################################
! Deny rfc 1918 addresses
! ################################################################
deny ip 192.168.0.0 0.0.255.255 any log
! remove this for testing purposes
! deny ip 172.16.0.0 0.15.255.255 any log
deny ip 10.0.0.0 0.255.255.255 any log
!
! ################################################################
! Deny packets with localhost, broadcast and multicast addresses
! ################################################################
deny ip 127.0.0.0 0.255.255.255 any log
deny ip 255.0.0.0 0.255.255.255 any log
deny ip 224.0.0.0 7.255.255.255 any log
!
```

```
! ##################################################################
! Deny packets without ip address
! ##################################################################
deny ip host 0.0.0.0 any log
!
! ##################################################################
! Spoof Prevention (Ingress & Egress Filtering)
! ##################################################################
! Prevent spoofing
! The edge router internal interface should only accept traffic with source
! addresses belonging to the internal network. The router external interface
! should only accept traffic with source addresses other than the internal network
!
! Concern is only with public (routable) addresses as any private addresses
! have already been dropped earlier.  In our case we are only
! concerned with external firewall interface and external VPN Gateway interface
!
permit ip host 172.16.0.1 any log
permit ip host 172.16.0.2 any log
! use following if SNAT not up
permit ip 192.168.0.0 0.0.0.255 any log
permit ip 192.168.0.0 0.0.0.255 any log
!
! ##################################################################
! Deny NetBIOS protocols from going out
! ##################################################################
! we don't particularly want to log these
deny tcp any any range 135 139
deny udp any any range 135 139
deny tcp any any eq 445 log
!
! ##################################################################
! Deny X-Windows Traffic
! ##################################################################
deny tcp any any range 6000 6255 log
!
! ##################################################################
! Deny TFTP, Syslog, and SNMP Traffic
! ##################################################################
!deny udp any any eq 69 log
!deny udp any any eq 514 log
deny udp any any range 161 162 log
!
! ##################################################################
! Deny NFS & Sun RPC
! ##################################################################
deny tcp any any eq 111 log
deny udp any any eq 111 log
deny tcp any any eq 2049 log
deny udp any any eq 2049 log
!
! ##################################################################
! Default
```

```
! ################################################################
! Deny and Log everything that does not meet the above rules.
! Use log-input to find violating machines
!
deny ip any any log-input
! HITS CISCO IOS IMPLICIT DENY
! ################################################################
```

Appendix D – Firewall Startup Shell Script iptables.rc

```
#!/bin/sh
#######################################################################

#######################################################################
# Script:         iptables.rc
# Description     linux iptables firewall startup script
# Author:         Mark E. Donaldson
# Date:           July 19, 2002
# Run Time:           run at boot time from:
#                 /etc/init.d/rc.local or
#                 /etc/init.d/boot.d or
#                 /etc/init.d/rc3.d
#                 ln -s /etc/init.d/iptables /etc/init.d/boot.d/S23iptables
#######################################################################

#######################################################################
# iptables.rc Script: Script Start
#######################################################################
# declare a trusted path
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/usr/local/bin
export PATH

# modify the IPT (iptables executable) variable to point to iptables binary
IPT=/usr/sbin/iptables

# set script variable name
SCRIPT_NAME="iptables.rc"

# set variable for ip_forwarding
FWD="/proc/sys/net/ipv4/ip_forward"

# set variable for controlling number of state table entries
STATE_CONNECTIONS="/proc/sys/net/ipv4/ip_conntrack_max"

# set variable for tcp connection timeout
TIMEOUT="/proc/sys/net/ipv4/tcp_fin_timeout"

# set variable for tcp keep alive interval
KEEP_ALIVE="/proc/sys/net/ipv4/tcp_keepalive_intvl"

# set variable for route verification ( built-in egress & ingress filtering)
# drop spoofed packets which cannot be replied to on the received
```

```
# interface based on routing tables
# * = all interfaces
RP_FILTER="/proc/sys/net/ipv4/conf/*/rp_filter"

# set variable for ICMP broadcasts
ICMP_ECHO_BROADCASTS ="/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts"

# set variable for source routing
# * = all interfaces
ACCEPT_SOURCE_ROUTE="/proc/sys/net/ipv4/conf/*/accept_source_route"

# set variable for accepting ICMP redirects
# * = all interfaces
REDIRECTS="/proc/sys/net/ipv4/conf/*/accept_redirects"

# set variable for sending ICMP redirects
# * = all interfaces
MAKE_REDIRECTS="/proc/sys/net/ipv4/conf/*/send_redirects"

# set variables for each firewall interface IP address
FIRE_ETH0="192.168.0.1"
FIRE_ETH1="172.16.0.1"
FIRE_ETH2="192.168.2.1"
FIRE_ETH3="192.168.1.1"

# set variables for each router interface IP address
ROUTE_ETH0="172.16.0.10"
ROUTE_ETH1="172.17.0.10"

# set variables for each vpn gateway interface IP address
VPN_ETH0="172.16.0.2"
VPN_ETH1="192.16.1.2"

# set variable for internal network
INT_NET="192.168.0.0/24"

# set variable for service network
SERVICE_NET="192.168.1.0/24"

# set variable for administrative network
ADMIN_NET="192.168.2.0/24"

# set variable for partners network
PART_NET="192.168.100.0/24"

#####################################################################
# Begin Execution
#####################################################################
# display firewall startup message
if [ -x /usr/bin/logger ];
then
    logger -p warning "Activating firewall script $SCRIPT_NAME "
    echo "Activating firewall script $SCRIPT_NAME"
```

```
fi

#####################################################################
# Link Modules As Needed
#####################################################################
# some modules we'll want to install if not compiled into kernel
# modprobe ip_tables
# modprobe ip_nat_ftp
# modprobe ip_conntrack_ftp

#####################################################################
# Start Clean - Flush and Delete Chains & Rules
#####################################################################
# The following lines will flush (-F) all rules in the chain shown as the argument to -F.
# Since no chain name was given as an argument, -F will flush all chains:

for i in filter nat mangle
do
     echo "Flushing Filter Chains . . . . . $i table"
     $IPT -t $i -F
     $IPT -t $i -X
done

#####################################################################
# Set Default and Static Routes
#####################################################################
echo "Setting default and static routes for server2 . . . . . "
logger -p warning "Setting default and static routes for server2 . . . . . "

# set default route for server2
route add default gw 172.16.0.10

# set static routes for server2 (firewall)
route add -net 172.16.0.0 netmask 255.255.0.0 dev eth1
route add -net 172.17.0.0 netmask 255.255.0.0 dev eth1
route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth3
route add -net 192.168.2.0 netmask 255.255.255.0 dev eth2

#####################################################################
# Set Critical Kernel Parameters
#####################################################################
# disable ip forwarding
echo 0 > $FWD

# set tcp connection timeout
# echo "30" > $TIMEOUT

# set tcp keep alive
# echo "1800" > $KEEP_ALIVE

# set up route verification ( built-in egress & ingress filtering)
# apply to all interfaces on machine
```

```
for f in $RP_FILTER;
do
    echo 1 > $f
done

# drop packets that are source routed
for f in $ACCEPT_SOURCE_ROUTE;
do
    echo 0 > $f
done

# drop ICMP redirect packets
for f in $REDIRECTS;
do
    echo 0 > $f
done

# deny sending of ICMP redirect packets
for f in /proc/sys/net/ipv4/conf/*/send_redirects;
do
    echo 0 > $f
done

##################################################################
# SECURITY POLICY FOR GIAC ENTERPRISES
##################################################################
# We are ready to start building our rule base and therefore need to refer to our basic written security
policy.
#
# NOTE:  All policy rules will be enforced, but enforcement will be distributed between the screening
router and the main firewall.  Some rules may be applied in both for redundancy
##################################################################
# The general policy scheme for the network is as follows:
#
# 1. Allow remote router & firewall management from single internal address (authorized administration).
#
# 2. Allow internal users to ping router and firewall interfaces to check connectivity (router &
lockdown/connectivity).
#
# 3. Drop all other internal and external traffic addressed to firewall and router (router & firewall lockdown).
#
# 4. Deny unwanted protocols, services, and ports (noise reduction, attack prevention, and Trojan
defense).
#
# 5. Deny unwanted network addresses (attack and spoof prevention).
#
# 6. Deny loop back & private addresses (reliability).
#
# 7. Deny broadcasts (smurf & DOS attack prevention).
#
# 8. Allow established internal tcp connections to pass (performance).
#
```

```
# 9. Allow internally initiated udp sessions to pass (performance).
#
# 10.    Allow internally initiated icmp queries and messages, and path MTU messages to pass
(performance).
#
# 11.    Allow inbound and outbound email and DNS services (performance).
#
# 12.    Deny internal DNS records to internet (attack prevention).
#
# 13.    Allow inbound access to web server (performance).
#
# 14.    Allow secure transactions (attack prevention).
#
# 15.    Allow encrypted communication (attack prevention).
#
# 16.    Allow internal clients unrestricted access to internet (business policy).
#
# 17.    Allow multicast (performance).
#
# 18.    Screen  hostile traffic (attack prevention).
#
# 19.    Allow centralized logging with and centralized backup (administration).
#
# 20.    Allow NTP time synchronization between network devices  over network segments
(administration).
#
# 21.    Deny everything else and log it (implicit deny)
#####################################################################

#####################################################################
# Define Default Policy
#####################################################################
# default policy for firewall will be deny all unless explicitly permitted.
echo "Setting default policies . . . . . filter table"
$IPT -t filter -P OUTPUT DROP
$IPT -t filter -P INPUT DROP
$IPT -t filter -P FORWARD DROP

echo "Setting default policies . . . . . nat table"
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT

echo "Setting default policies . . . . . mangle table"
$IPT -t mangle -P PREROUTING ACCEPT
$IPT -t mangle -P OUTPUT ACCEPT
#####################################################################

#####################################################################
# Create NAT Rules
#####################################################################
# since we are using private addressing on our internal network
# we need to SNAT all the outgoing packets to use the address of the firewall
```

```
# external interface
#####################################################################
# since SNAT is POSTROUTING we must append the POSTROUTING chain
$IPT -t nat -A POSTROUTING -o eth1 -j LOG --log-prefix "SNAT OUT ETH1  "
$IPT -t nat -A POSTROUTING -o eth1 -j SNAT --to 172.16.0.1

# send all incoming web traffic to Squid Proxy (HTTP & SHTTP)
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth1 -j LOG --log-prefix "DNAT IN 8080  "
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth1 -j DNAT --to 192.168.1.2:8080
$IPT -t nat -A PREROUTING -p tcp --destination-port 443 -i eth1 -j DNAT --to 192.168.1.2:8080

# send all incoming DNS traffic to External DNS server
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p udp --destination-port 53 -i eth1 -j LOG --log-prefix "DNAT IN 53  "
$IPT -t nat -A PREROUTING -p udp --destination-port 53 -i eth1 -j DNAT --to 192.168.1.2:53
$IPT -t nat -A PREROUTING -p tcp --destination-port 53 -i eth1 -j DNAT --to 192.168.1.2:53

# send all incoming mail traffic to sendmail relay
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 25 -i eth1 -j LOG --log-prefix "DNAT IN 25  "
$IPT -t nat -A PREROUTING -p tcp --destination-port 25 -i eth1 -j DNAT --to 192.168.1.2:25

# send all outgoing web traffic to Squid Proxy (HTTP & SHTTP)
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth0 -j LOG --log-prefix "DNAT OUT 8080  "
$IPT -t nat -A PREROUTING -p tcp --destination-port 80 -i eth0 -j DNAT --to 192.168.1.2:8080
$IPT -t nat -A PREROUTING -p tcp --destination-port 443 -i eth0 -j DNAT --to 192.168.1.2:8080

# send all outgoing FTP traffic to Squid Proxy
# since DNAT is PREROUTING we must append the PREROUTING chain
$IPT -t nat -A PREROUTING -p tcp --destination-port 21:22 -i eth0 -j LOG --log-prefix "DNAT FTP  "
$IPT -t nat -A PREROUTING -p udp --destination-port 21:22 -i eth0 -j DNAT --to 192.168.1.2:8080

#####################################################################
# Create User Defined Chains
#####################################################################
# We need identical rules in each of the FORWARD and INPUT chains.
# To prevent duplicating a lot of rules, let's create user chains and call
# them from both INPUT and FORWARD

# We also will create a special rule (Rule_X) to detect and log port scans
#####################################################################
echo "Creating User Defined Chains . . . . . "
# TCP rules
$IPT -t filter -N tcprules

# UDP rules
$IPT -t filter -N udprules

# ICMP rules
$IPT -t filter -N icmprules
```

```
# AH rules
$IPT -t filter -N ahrules

# ESP rules
$IPT -t filter -N esprules

# OSPF rules
$IPT -t filter -N ospfrules

# IGRP rules
$IPT -t filter -N igrprules

# GRE rules
$IPT -t filter -N grerules

# RULE_X is a stealth port scan detector
$IPT -t filter -N RULE_X

# RULE_Y is a trojan detector
$IPT -t filter -N RULE_Y

# RULE_Z is a spoofing detector
$IPT -t filter -N RULE_Z

#####################################################################
# Set Stateful Inspection Connection Rules
#####################################################################
# The `state' extension invokes the connection-tracking analysis of the `ip_conntrack' module.
# Specifying `-m state' allows an additional `--state' option, which is a comma-separated list
# of states to match (the `!' flag indicates not to match those states). These states are:
#
# NEW = A packet which creates a new connection.
# NEW applies to packets with only the SYN bit set (and the ACK bit unset).
#
# ESTABLISHED = A packet which belongs to an existing connection.
# ESTABLISHED permits traffic to continue where it has seen traffic before in both directions.
# ESTABLISHED not only  applies to TCP connections but to UDP traffic, such as DNS queries
# and traceroutes as well as ICMP pings.
#
# RELATED = A packet which is related to, but not part of, an existing connection.
# RELATED includes packets such as an ICMP error, or a packet establishing an ftp data connection.
# RELATED applies to active FTP, which opens a related connection on port 20,
# but also applies to ICMP traffic related to the TCP connection.
#
# INVALID = A packet which could not be identified for some reason.
# This includes ICMP errors which don't correspond to any known connection.
# INVALID applies to packets that have invalid sets of options, as in an XMAS tree scan.
#####################################################################

# basic policy is to log and drop any new and invalid connections from outside

####################
# TCP RULES INBOUND
```

```
####################
# here we INSERT at top of chain and shove down
echo "Creating TCP Rules - Inbound . . . . . "
# block IRC
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6667 -j DROP

# block NetBIOS
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 445 -j DROP

# block X-Windows Traffic
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6000:6200 -j DROP

# Block NFS & Sun RPC
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 111 -j LOG --log-prefix "SUN RPC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 111 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 2049 -j LOG --log-prefix "NFS ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 2049 -j DROP

# permit Ident to mail relay
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 113 -j LOG --log-prefix "IDENT CALLED  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 113 -j ACCEPT

# FIREWALL LOCKDOWN
# permit tcp connections from router and VPN gateway
# block everything else
$IPT -t filter -I tcprules -i eth1 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK TCP TO FIREWALL 72  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp -s 172.16.0.10 -j LOG --log-prefix "TCP FROM ROUTER  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp -s 172.16.0.10 -j ACCEPT
#$IPT -t filter -I tcprules -i eth1 -p tcp -s 172.16.0.2 -j LOG --log-prefix "TCP FROM VPN  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp -s 172.16.0.2 -j ACCEPT

# APPEND
# accept new inbound connections to Squid Proxy
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 8080 -m state --state NEW -j LOG --log-prefix
"SQUID-NEW  "
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 8080 -m state --state NEW -j ACCEPT

# permit inbound mail to mail relay
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 25 -j LOG --log-prefix "INCOMING MAIL  "
$IPT -t filter -A tcprules -i eth1 -p tcp --destination-port 25 -j ACCEPT

# accept all remaining inbound traffic established internally
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A tcprules -i eth3 -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A tcprules -i eth2 -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT

# process and drop all other inbound TCP connections
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state NEW -j LOG --log-prefix "BLOCK TCP ATTEMPT  "
```

```
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state NEW -j DROP
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state INVALID -j LOG --log-prefix "INVALID TCP
ATTEMPT "
# $IPT -t filter -A tcprules -i eth1 -p tcp -m state --state INVALID -j RULE_X
$IPT -t filter -A tcprules -i eth1 -p tcp -m state --state INVALID -j DROP

# PLACE THIS LAST (last resort)
# $IPT -t filter -A tcprules -i eth1 -j REJECT --reject-with icmp-host-unreachable

####################
# TCP RULES OUTBOUND
####################
echo "Creating TCP Rules - Outbound . . . . . "
# INSERT
# block IRC
$IPT -t filter -I tcprules -i eth0 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp --destination-port 6667 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 6667 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6667 -j LOG --log-prefix "IRC ATTEMPT  "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6667 -j DROP

# block NetBIOS
$IPT -t filter -I tcprules -i eth0 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules 2 -i eth0 -p tcp --destination-port 445 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 445 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 35:39 -j DROP
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 445 -j DROP

# block X-Windows Traffic
$IPT -t filter -I tcprules -i eth0 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp --destination-port 6000:6200 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 6000:6200 -j DROP
$IPT -t filter -I tcprules -i eth1 -p tcp --destination-port 6000:6200 -j LOG --log-prefix "XWINDOWS
ATTEMPT "
$IPT -t filter -I tcprules 2 -i eth1 -p tcp --destination-port 6000:6200 -j DROP

# FIREWALL AND ROUTER LOCKDOWN
# permit administrative connections from 192.168.0.3 only
# block everything else
$IPT -t filter -I tcprules -i eth0 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK HOME TO FIREWALL "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth0 -p tcp -d 172.16.0.10 -j LOG --log-prefix "BLOCK HOME TO ROUTER "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -d 172.16.0.10 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK SERVICE TO FIREWALL "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth3 -p tcp -d 172.16.0.10 -j LOG --log-prefix "BLOCK SERVICE TO ROUTER "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp -d 172.16.0.10 -j DROP
$IPT -t filter -I tcprules -i eth2 -p tcp -d 172.16.0.1 -j LOG --log-prefix "BLOCK ADMIN TO FIREWALL "
```

```
$IPT -t filter -I tcprules 2 -i eth2 -p tcp -d 172.16.0.1 -j DROP
$IPT -t filter -I tcprules -i eth2 -p tcp -d 172.16.0.10 -j LOG --log-prefix "BLOCK ADMIN TO ROUTER "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp -d 172.16.0.10 -j DROP
# permit
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 192.168.0.1 -j LOG --log-prefix "HOME CONNECT
FIREWALL 01 "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 192.168.0.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 192.168.1.1 -j LOG --log-prefix "HOME CONNECT
FIREWALL 11 "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 192.168.1.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 192.168.2.1 -j LOG --log-prefix "HOME CONNECT
FIREWALL 21"
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 192.168.2.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.1 -j LOG --log-prefix "HOME CONNECT
FIREWALL 72"
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.1 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.10 -j LOG --log-prefix "HOME CONNECT
TO ROUTER "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 -d 172.16.0.10 -j ACCEPT

# permit all http connections from 192.168.0.3 only
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 80 -j LOG --log-prefix "HTTP FROM
HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 80 -j ACCEPT
# for Webmin use port 10000
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 10000 -j LOG --log-prefix "WEBMIN
FROM HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 10000 -j ACCEPT

# permit all ssh connections from 192.168.0.3 only
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j LOG --log-prefix "SSH FROM
HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j ACCEPT

# permit all telnet connections from 192.168.0.3 only
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 23 -j LOG --log-prefix "TELNET
FROM HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 23 -j ACCEPT

# permit all VNC connections from 192.168.0.3 only
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 5901 -j LOG --log-prefix "VNC
FROM HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 5901 -j ACCEPT

# permit all ftp connections from 192.168.0.3 only
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 20 -j LOG --log-prefix "FTP FROM
HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 20 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -p tcp -s 192.168.0.3 --destination-port 21 -j LOG --log-prefix "FTP FROM
HOME "
$IPT -t filter -I tcprules 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 21 -j ACCEPT

# permit and log service network & administrative network access to mail
```

```
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 25 -j LOG --log-prefix "TO MAIL RELAY "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 25 -j ACCEPT
$IPT -t filter -I tcprules -i eth2 -p tcp --destination-port 25 -j LOG --log-prefix "TO MAIL RELAY "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp --destination-port 25 -j ACCEPT

# permit DNS zone transfer from primary to secondary external DNS
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 53 -j LOG --log-prefix "ZONE TRANSFER "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 53 -j ACCEPT
$IPT -t filter -I tcprules -i eth2 -p tcp --destination-port 53 -j LOG --log-prefix "ZONE TRANSFER "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp --destination-port 53 -j ACCEPT

# permit oracle service between internal and service networks
$IPT -t filter -I tcprules -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1521 -j LOG --log-prefix
"ORACLE CONNECTION  "
$IPT -t filter -I tcprules 2 -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1521 -j ACCEPT
$IPT -t filter -I tcprules -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1575 -j LOG --log-prefix
"ORACLE CONNECTION  "
$IPT -t filter -I tcprules 2 -i eth3 -d 192.168.0.0/24 -p tcp --destination-port 1575 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1521 -j LOG --log-prefix
"ORACLE CONNECTION  "
$IPT -t filter -I tcprules 2 -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1521 -j ACCEPT
$IPT -t filter -I tcprules -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1575 -j LOG --log-prefix
"ORACLE CONNECTION  "
$IPT -t filter -I tcprules 2 -i eth0 -d 192.168.1.0/24 -p tcp --destination-port 1575 -j ACCEPT

# permit and log amanda backups
$IPT -t filter -I tcprules -i eth3 -p tcp --destination-port 10080:10083 -j LOG --log-prefix "AMANDA
BACKUP OPS  "
$IPT -t filter -I tcprules 2 -i eth3 -p tcp --destination-port 10080:10083 -j ACCEPT
$IPT -t filter -I tcprules -i eth2 -p tcp --destination-port 10080:10083 -j LOG --log-prefix "AMANDA
BACKUP OPS  "
$IPT -t filter -I tcprules 2 -i eth2 -p tcp --destination-port 10080:10083 -j ACCEPT

# APPEND
# permit unrestricted access to the internet from the internal network
#$IPT -t filter -A tcprules -i eth0 -p tcp -m state --state NEW -j LOG --log-prefix "STATE TCP ACCEPT  "
$IPT -t filter -A tcprules -i eth0 -p tcp -m state --state NEW -j ACCEPT
$IPT -t filter -A tcprules -i eth0 -p tcp -j LOG --log-prefix "NO STATE TCP ACCEPT  "
$IPT -t filter -A tcprules -i eth0 -p tcp -j ACCEPT

# deny all other tcp connections from remaining devices on internal network
$IPT -t filter -A tcprules -i eth0 -p tcp -s 192.168.0.0/24 -j LOG --log-prefix "TCP DENY FROM HOME  "
$IPT -t filter -A tcprules -i eth0 -p tcp -s 192.168.0.0/24 -j DROP

# deny all other tcp connections from remaining devices on service network
$IPT -t filter -A tcprules -i eth3 -p tcp -s 192.168.1.0/24 -j LOG --log-prefix "TCP DENY FROM SERVICE
"
$IPT -t filter -A tcprules -i eth3 -p tcp -s 192.168.1.0/24 -j DROP

# deny all other tcp connections from remaining devices on admin network
$IPT -t filter -A tcprules -i eth2 -p tcp -s 192.168.2.0/24 -j LOG --log-prefix "TCP DENY FROM ADMIN  "
$IPT -t filter -A tcprules -i eth2 -p tcp -s 192.168.2.0/24 -j DROP
```

```
###################
# UDP RULES INBOUND
###################
echo "Creating UDP Rules - Inbound . . . . . "
# here we INSERT at top of chain and shove down
# block NetBIOS
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 35:39 -j DROP

# Block NFS & Sun RPC
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 111 -j LOG --log-prefix "SUN RPC ATTEMPT "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 111 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 2049 -j LOG --log-prefix "NFS ATTEMPT  "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 2049 -j DROP

# block remaining TFTP, Syslog, and SNMP Traffic
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 69 -j LOG --log-prefix "TFTP ATTEMPT "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 69 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 514 -j LOG --log-prefix "SYSLOG ATTEMPT "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 514 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 161:162 -j LOG --log-prefix "SNMP ATTEMPT  "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 161:162 -j DROP

# FIREWALL LOCKDOWN
# permit tcp connections from router and VPN gateway
# block everything else
$IPT -t filter -I udprules -i eth1 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK UDP TO FIREWALL 72  "
$IPT -t filter -I udprules 2 -i eth1 -p udp -d 172.16.0.1 -j DROP
# permit syslog and tftp from router and VPN
$IPT -t filter -I udprules -i eth1 -p udp -s 172.16.0.10 --destination-port 514 -j LOG --log-prefix "SYSLOG
FROM ROUTER  "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 172.16.0.10 --destination-port 514 -j ACCEPT
# from router loop back logging facility
$IPT -t filter -I udprules -i eth1 -p udp -s 192.168.3.1 --destination-port 514 -j LOG --log-prefix "SYSLOG
FROM ROUTER  "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 192.168.3.1 --destination-port 514 -j ACCEPT

$IPT -t filter -I udprules -i eth1 -p udp -s 172.16.0.10 --destination-port 69 -j LOG --log-prefix "TFTP
FROM ROUTER "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 172.16.0.10 --destination-port 69 -j ACCEPT
#$IPT -t filter -I udprules -i eth1 -p udp -s 172.16.0.2 --destination-port 514 -j LOG --log-prefix "SYSLOG
FROM VPN  "
$IPT -t filter -I udprules 2 -i eth1 -p udp -s 172.16.0.2 --destination-port 514 -j ACCEPT

# permit DNS inbound to external DNS server
#$IPT -t filter -I udprules -i eth1 -p udp --destination-port 53 -j LOG --log-prefix "EXTERNAL DNS CALL "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 53 -j ACCEPT

# permit communication with external NTP stratum 2 server
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 1233 -j LOG --log-prefix "INTERNET NTP CALL "
$IPT -t filter -I udprules 2 -i eth1 -p udp --destination-port 1233 -j ACCEPT

# APPEND
# accept connections established internally
```

```
$IPT -t filter -A udprules -i eth1 -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A udprules -i eth3 -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A udprules -i eth2 -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

# process and drop all other UDP connections
#$IPT -t filter -A udprules -i eth1 -p udp -m state --state NEW -j LOG --log-prefix "BLOCK EXTERNAL
UDP   "
$IPT -t filter -A udprules -i eth1 -p udp -m state --state NEW -j DROP
$IPT -t filter -A udprules -i eth1 -p udp -m state --state INVALID -j LOG --log-prefix "BLOCK INVALID UDP
"
$IPT -t filter -A udprules -i eth1 -p udp -m state --state INVALID -j DROP

####################
# UDP RULES OUTBOUND
####################
echo "Creating UDP Rules - Outbound . . . . . "
# here we INSERT at top of chain and shove down
# block NetBIOS
$IPT -t filter -I udprules -i eth0 -p udp --destination-port 35:39 -j DROP
$IPT -t filter -I udprules -i eth3 -p udp --destination-port 35:39 -j DROP
$IPT -t filter -I udprules -i eth1 -p udp --destination-port 35:39 -j DROP

# firewall and router lockdown
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.3 -d 192.168.0.1 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.3 -d 192.168.0.1 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.2 -d 192.168.0.1 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.2 -d 192.168.0.1 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.3 -d 172.16.0.10 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.3 -d 172.16.0.10 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.2 -d 172.16.0.10 -j LOG --log-prefix "HOME
CONNECT FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.2 -d 172.16.0.10 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK HOME TO FIREWALL "
$IPT -t filter -I udprules 2 -i eth0 -p udp -d 172.16.0.1 -j DROP
$IPT -t filter -I udprules -i eth0 -p udp -d 172.16.0.10 -j LOG --log-prefix "BLOCK HOME TO ROUTER "
$IPT -t filter -I udprules 2 -i eth0 -p udp -d 172.16.0.10 -j DROP
$IPT -t filter -I udprules -i eth3 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK SERVICE TO FIREWALL
"
$IPT -t filter -I udprules 2 -i eth3 -p udp -d 172.16.0.1 -j DROP
$IPT -t filter -I udprules -i eth3 -p udp -d 172.16.0.10 -j LOG --log-prefix "BLOCK SERVICE TO ROUTER
"
$IPT -t filter -I udprules 2 -i eth3 -p udp -d 172.16.0.10 -j DROP
$IPT -t filter -I udprules -i eth2 -p udp -d 172.16.0.1 -j LOG --log-prefix "BLOCK ADMIN TO FIREWALL "
$IPT -t filter -I udprules 2 -i eth2 -p udp -d 172.16.0.1 -j DROP
$IPT -t filter -I udprules -i eth2 -p udp -d 172.16.0.10 -j LOG --log-prefix "BLOCK ADMIN TO ROUTER "
$IPT -t filter -I udprules 2 -i eth2 -p udp -d 172.16.0.10 -j DROP

# permit all ftp connections from single workstation on internal network
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.3 --destination-port 20 -j LOG --log-prefix "FTP FROM
HOME "
```

```
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.3 --destination-port 20 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -s 192.168.0.3 --destination-port 21 -j LOG --log-prefix "FTP FROM
HOME "
$IPT -t filter -I udprules 2 -i eth0 -p udp -s 192.168.0.3 --destination-port 21 -j ACCEPT

# permit and log network access to DNS
#$IPT -t filter -I udprules -i eth3 -p udp --destination-port 53 -j LOG --log-prefix "SERVICE DNS CALL  "
$IPT -t filter -I udprules 2 -i eth3 -p udp --destination-port 53 -j ACCEPT
#$IPT -t filter -I udprules -i eth2 -p udp --destination-port 53 -j LOG --log-prefix "ADMIN DNS CALL  "
$IPT -t filter -I udprules 2 -i eth2 -p udp --destination-port 53 -j ACCEPT
#$IPT -t filter -I udprules -i eth0 -p udp --destination-port 53 -j LOG --log-prefix "INTERNAL DNS CALL  "
$IPT -t filter -I udprules 2 -i eth0 -p udp --destination-port 53 -j ACCEPT

# permit and log all internal network access to logging
$IPT -t filter -I udprules -i eth3 -p udp -d 192.168.2.2 --destination-port 514 -j LOG --log-prefix "SYSLOG
OPERATION  "
$IPT -t filter -I udprules 2 -i eth3 -p udp -d 192.168.2.2 --destination-port 514 -j ACCEPT
$IPT -t filter -I udprules -i eth2 -p udp -d 192.168.2.2 --destination-port 514 -j LOG --log-prefix "SYSLOG
OPERATION  "
$IPT -t filter -I udprules 2 -i eth2 -p udp -d 192.168.2.2 --destination-port 514 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp -d 192.168.2.2 --destination-port 514 -j LOG --log-prefix "SYSLOG
OPERATION  "
$IPT -t filter -I udprules 2 -i eth0 -p udp -d 192.168.2.2 --destination-port 514 -j ACCEPT

# permit and log all internal NTP activity
$IPT -t filter -I udprules -i eth3 -p udp --destination-port 123 -j LOG --log-prefix "NTP OPERATION  "
$IPT -t filter -I udprules 2 -i eth3 -p udp --destination-port 123 -j ACCEPT
$IPT -t filter -I udprules -i eth2 -p udp --destination-port 123 -j LOG --log-prefix "NTP OPERATION  "
$IPT -t filter -I udprules 2 -i eth2 -p udp --destination-port 123 -j ACCEPT
$IPT -t filter -I udprules -i eth0 -p udp --destination-port 123 -j LOG --log-prefix "NTP OPERATION  "
$IPT -t filter -I udprules 2 -i eth0 -p udp --destination-port 123 -j ACCEPT

# APPEND
# allow unrestricted access to the internet from the internal network
# $IPT -t filter -A udprules -i eth0 -p udp -m state --state NEW -j LOG --log-prefix "STATE UDP ACCEPT
"
$IPT -t filter -A udprules -i eth0 -p udp -m state --state NEW -j ACCEPT

#$IPT -t filter -A udprules -i eth0 -p udp -j LOG --log-prefix " NO STATE UDP ACCEPT  "
$IPT -t filter -A udprules -i eth0 -p udp -j ACCEPT

# deny all remaining udp connections from network
#$IPT -t filter -A udprules -i eth0 -p udp -s 192.168.0.0/24 -j LOG --log-prefix "UDP DENY FROM HOME "
$IPT -t filter -A udprules  -i eth0 -p udp -s 192.168.0.0/24 -j DROP

# deny all remaining udp connections from remaining devices on service network
#$IPT -t filter -A udprules -i eth3 -p udp -s 192.168.1.0/24 -j LOG --log-prefix "UDP DENY FROM SERVIE
"
$IPT -t filter -A udprules  -i eth3 -p udp -s 192.168.1.0/24 -j DROP

# deny all remaining udp connections from remaining devices on administrative network
#$IPT -t filter -A udprules -i eth2 -p udp -s 192.168.0.0/24 -j LOG --log-prefix "UDP DENY FROM ADMIN
"
```

```
$IPT -t filter -A udprules  -i eth2 -p udp -s 192.168.0.0/24 -j DROP

####################
# ICMP RULES INBOUND
####################
echo "Creating ICMP Rules - Inbound . . . . . "
# here we INSERT at top of chain and shove down
# do not allow firewall to be pinged from internet
iptables -t filter -I icmprules -i eth1 -p icmp --icmp-type echo-request -j DROP

# APPEND
# permit icmp
$IPT -t filter -A icmprules -i eth1 -p icmp -j ACCEPT

#####################
# ICMP RULES OUTBOUND
#####################
echo "Creating ICMP Rules - Outbound . . . . . "
# APPEND
# permit icmp
$IPT -t filter -A icmprules -i eth0 -p icmp -j ACCEPT
$IPT -t filter -A icmprules -i eth2 -p icmp -j ACCEPT
$IPT -t filter -A icmprules -i eth3 -p icmp -j ACCEPT

#######################
# AH RULES: PROTOCOL 51
#######################
echo "Creating AH Rules . . . . . "
# deny all AH inbound
$IPT -t filter -A ahrules -i eth1 -p 51 -j LOG --log-prefix "AH DENIED  "
$IPT -t filter -A ahrules -i eth1 -p 51 -j DROP

#######################
# ESP RULES: PROTOCOL 50
#######################
echo "Creating ESP Rules . . . . . "
# deny all ESP inbound
$IPT -t filter -A esprules -i eth1 -p 50 -j LOG --log-prefix "ESP DENIED  "
$IPT -t filter -A esprules -i eth1 -p 50 -j DROP

#######################
# OSPF RULES: PROTOCOL 89
#######################
echo "Creating OSPF Rules . . . . . "
# deny all OSPF inbound
$IPT -t filter -A ospfrules -i eth1 -p 89 -j LOG --log-prefix "OSPF DENIED  "
$IPT -t filter -A ospfrules -i eth1 -p 89 -j DROP

####################
# IGRP RULES: PROTOCOL 9
####################
echo "Creating IGRP Rules . . . . . "
# deny all IGRP inbound
```

```
$IPT -t filter -A igrprules -i eth1 -p 9 -j LOG --log-prefix "IGRP DENIED  "
$IPT -t filter -A igrprules -i eth1 -p 9 -j DROP

#####################
# GRE RULES: PROTOCOL 47
#####################
echo "Creating GRE Rules . . . . . "
# deny all IGRP inbound
$IPT -t filter -A grerules -i eth1 -p 47 -j LOG --log-prefix "GRE DENIED  "
$IPT -t filter -A grerules -i eth1 -p 47 -j DROP

#####################
# OUTPUT RULES
#####################
echo "Creating OUTPUT Rules  . . . . . "
# here we insert at top of chain and shove down
# allow MTU discovery
$IPT -t filter -I OUTPUT -p icmp --icmp-type 3/4 -j LOG --log-prefix "ICMP DON'T FRAGMENT  "
$IPT -t filter -I OUTPUT 2 -p icmp --icmp-type 3/4 -j ACCEPT

# allow echo replies to single workstation on internal network
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p icmp --icmp-type 0 -j ACCEPT

# permit ftp to single workstation on internal network
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p tcp --destination-port 20 -j LOG --log-prefix "FIREWALL FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p tcp --destination-port 20 -j ACCEPT
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p tcp --destination-port 21 -j LOG --log-prefix "FIREWALL FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p tcp --destination-port 21 -j ACCEPT
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p udp --destination-port 21 -j LOG --log-prefix "FIREWALL FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p udp --destination-port 21 -j ACCEPT
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p udp --destination-port 20 -j LOG --log-prefix "FIREWALL FTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p udp --destination-port 20 -j ACCEPT

# permit telnet connections to single workstation on internal network
$IPT -t filter -I OUTPUT -d 192.168.0.3 -p tcp --destination-port 23 -j LOG --log-prefix "FIREWALL
TELNET  "
$IPT -t filter -I OUTPUT 2 -d 192.168.0.3 -p tcp --destination-port 23 -j ACCEPT

# permit syslog connections to syslog server on network
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p udp --destination-port 123 -j LOG --log-prefix "FIREWALL NTP
"
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p udp --destination-port 123 -j ACCEPT

# permit syslog connections to NTP server on network
#$IPT -t filter -I OUTPUT -d 192.168.2.2 -p udp --destination-port 514 -j LOG --log-prefix "FIREWALL
SYSLOG  "
$IPT -t filter -I OUTPUT 2 -d 192.168.2.2 -p udp --destination-port 514 -j ACCEPT

# permit connections to mail relay
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p tcp --destination-port 25 -j LOG --log-prefix "FIREWALL MAIL  "
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p tcp --destination-port 25 -j ACCEPT

# APPEND
```

```
# permit established
$IPT -t filter -A OUTPUT -p tcp -m state --state ESTABLISHED,RELATED -j LOG --log-prefix "FIREWALL
PERMIT EST-TCP  "
$IPT -t filter -A OUTPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -t filter -A OUTPUT -p udp -m state --state ESTABLISHED,RELATED -j LOG --log-prefix
"FIREWALL PERMIT EST-UDP  "
$IPT -t filter -A OUTPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

# permit access to all of firewall own interfaces
#$IPT -t filter -A OUTPUT -p tcp -d 192.168.0.1 -j LOG --log-prefix "FIREWALL TO SELF TCP01  "
$IPT -t filter -A OUTPUT -p tcp -d 192.168.1.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p tcp -d 192.168.0.1 -j LOG --log-prefix "FIREWALL TO SELF TCP11  "
$IPT -t filter -A OUTPUT -p tcp -d 192.168.1.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p tcp -d 192.168.2.1 -j LOG --log-prefix "FIREWALL TO SELF TCP21  "
$IPT -t filter -A OUTPUT -p tcp -d 192.168.2.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p tcp -d 172.16.0.1 -j LOG --log-prefix "FIREWALL TO SELF TCP72  "
$IPT -t filter -A OUTPUT -p tcp -d 172.16.0.1 --j ACCEPT

#$IPT -t filter -A OUTPUT -p udp -d 192.168.0.1 -j LOG --log-prefix "FIREWALL TO SELF UDP01  "
$IPT -t filter -A OUTPUT -p udp -d 192.168.1.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p udp -d 192.168.0.1 -j LOG --log-prefix "FIREWALL TO SELF UDP11  "
$IPT -t filter -A OUTPUT -p udp -d 192.168.1.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p udp -d 192.168.2.1 -j LOG --log-prefix "FIREWALL TO SELF UDP21  "
$IPT -t filter -A OUTPUT -p udp -d 192.168.2.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p udp -d 172.16.0.1 -j LOG --log-prefix "FIREWALL TO SELF UDP72  "
$IPT -t filter -A OUTPUT -p udp -d 172.16.0.1 --j ACCEPT

#$IPT -t filter -A OUTPUT -p icmp -d 192.168.0.1 -j LOG --log-prefix "FIREWALL TO SELF ICMP01  "
$IPT -t filter -A OUTPUT -p icmp -d 192.168.1.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p icmp -d 192.168.0.1 -j LOG --log-prefix "FIREWALL TO SELF ICMP11  "
$IPT -t filter -A OUTPUT -p icmp -d 192.168.1.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p icmp -d 192.168.2.1 -j LOG --log-prefix "FIREWALL TO SELF ICMP21  "
$IPT -t filter -A OUTPUT -p icmp -d 192.168.2.1 --j ACCEPT
#$IPT -t filter -A OUTPUT -p icmp -d 172.16.0.1 -j LOG --log-prefix "FIREWALL TO SELF ICMP72  "
$IPT -t filter -A OUTPUT -p icmp -d 172.16.0.1 --j ACCEPT

# firewall lockdown
#$IPT -t filter -A OUTPUT -p tcp -j LOG --log-prefix "BLOCKED TCP FROM FIREWALL  "
$IPT -t filter -A OUTPUT -p tcp -j DROP
#$IPT -t filter -A OUTPUT -p udp -j LOG --log-prefix "BLOCKED UDP FROM FIREWALL  "
$IPT -t filter -A OUTPUT -p udp -j DROP

#####################################################################
# RULE_X: Attack & Stealth Port Scan Detection
#####################################################################
echo "Creating RULE_X: Stealth Scan Detection . . . . . "
# Block & Log SYN/FIN scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,FIN -j LOG --log-prefix "RULE_X DENY: SYN/FIN
Scan  "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,FIN -j DROP

# Block & Log FIN scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL FIN -j LOG --log-prefix "RULE_X DENY: FIN Scan  "
```

```
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL FIN -j DROP

# Block & Log ACK scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL ACK -j LOG --log-prefix "RULE_X DENY: ACK Scan"
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL ACK -j DROP

# Block & Log SYN/ACK scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK -j LOG --log-prefix "RULE_X DENY: SNY/ACK
Scan  "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK -j DROP

# Block & Log XMAS TREE scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK,URG,PSH,RST,FIN -j LOG --log-prefix "RULE_X
DENY: XMAS TREE Scan  "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL SYN,ACK,URG,PSH,RST,FIN -j DROP

# Block & Log NULL scans.
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL NONE -j LOG --log-prefix "RULE_X DENY: NULL Scan  "
$IPT -t filter -A RULE_X -p tcp --tcp-flags ALL NONE -j DROP
#####################################################################

#####################################################################
# RULE_Y: Trojan Rules
#####################################################################
echo "Creating RULE_Y: Trojan Detection . . . . . "
# block netbus, subseven, and backorifice trojans
# here we insert at top of chain and shove down
# insert in FORWARD chain as needed
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 12345:12346 -j LOG --log-prefix "NETBUS
DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 12345:12346 -j DROP
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 6666:6667 -j LOG --log-prefix "NETBUS
DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 6666:6667 -j DROP
#$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 27374 -j LOG --log-prefix "SUBSEVEN
DENIED"
#$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 27374 -j DROP
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 2773:2774 -j LOG --log-prefix "SUBSEVEN
DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 2773:2774 -j DROP
#$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 7000 -j LOG --log-prefix "SUBSEVEN
DENIED"
#$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 7000 -j DROP
#$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 54320:54321 -j LOG --log-prefix "BACK
ORIFICE DENIED  "
#$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 54320:54321 -j DROP
$IPT -t filter -I RULE_Y  -i eth1 -d 0/0 -p tcp --destination-port 31337:31338 -j LOG --log-prefix "BACK
ORIFICE DENIED  "
$IPT -t filter -I RULE_Y 2 -i eth1 -d 0/0 -p tcp --destination-port 31337:31338 -j DROP

#####################################################################
# RULE_Z: Spoofing Rules (egress-filtering & ingress-filtering)
#####################################################################
```

```
echo "Creating RULE_Z: Spoofing Rules . . . . . egress & ingress filtering"
# block all routable addresses with source address of network

# EGRESS FILTERING
# block any packets coming from the network with spoofed addresses (egress filtering)
$IPT -t filter -I RULE_Z -i eth0 ! -s 192.168.0.0/24 -j LOG --log-prefix "INTERNAL SPOOF ATTEMPT  "
$IPT -t filter -I RULE_Z 2 -i eth0 ! -s 192.168.0.0/24 -j DROP
$IPT -t filter -I RULE_Z -i eth3 ! -s 192.168.1.0/24 -j LOG --log-prefix "INTERNAL SPOOF ATTEMPT  "
$IPT -t filter -I RULE_Z 2 -i eth3 ! -s 192.168.1.0/24 -j DROP
$IPT -t filter -I RULE_Z -i eth2 ! -s 192.168.2.0/24 -j LOG --log-prefix "INTERNAL SPOOF ATTEMPT  "
$IPT -t filter -I RULE_Z 2 -i eth2 ! -s 192.168.2.0/24 -j DROP

# INGRESS FILTERING
# block any packets coming into the network with spoofed addresses (ingress filtering)
$IPT -t filter -A RULE_Z -i eth1 -s 10.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 10.0.0.0/8 -j DROP
# comment out 172.16 for testing
#$IPT -t filter -A RULE_Z -i eth1 -s 172.16.0.0/12 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
#$IPT -t filter -A RULE_Z -i eth1 -s 172.16.0.0/12 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.0.0/16 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.0.0/16 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.4.0/22 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.4.0/22 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.8.0/21 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.8.0/21 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.16.0/20 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.16.0/20 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.32.0/19 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.32.0/19 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.64.0/18 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.64.0/18 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.128.0/17 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 192.168.128.0/17 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 127.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 127.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 0.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 0.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 2.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 2.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 5.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 5.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 7.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 7.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 23.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 23.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 27.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 27.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 31.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 31.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 36.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 36.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 37.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 37.0.0.0/8 -j DROP
```

```
$IPT -t filter -A RULE_Z -i eth1 -s 39.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 39.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 41.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 41.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 42.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 42.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 58.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 58.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 59.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 59.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 60.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 60.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 72.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 72.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 82.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 82.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 83.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 83.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 84.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 84.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 88.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 88.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 96.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 96.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 197.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 197.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 221.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 221.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 222.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 222.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 223.0.0.0/8 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 223.0.0.0/8 -j DROP
$IPT -t filter -A RULE_Z -i eth1 -s 224.0.0.0/3 -j LOG --log-prefix "EXTERNAL SPOOF ATTEMPT  "
$IPT -t filter -A RULE_Z -i eth1 -s 224.0.0.0/3 -j DROP

#####################################################################
# Specialized Rules
#####################################################################
echo "Creating Specialized Rules . . . . . "
# Protecting Data
# It's very difficult to make sure someone on the inside isn't passing
# data through your firewall that shouldn't go out. We can try to prevent
# certain data from leaving by marking that data, then looking for that
# mark using the string match. It may be a good idea to implement a policy of
# putting a string, such as "Copyright: GIAC Enterprises - not for publication"
# at the top of those files you don't want sent through the firewall.
# Then, on the inner firewall, you might want something like:
iptables -t filter -I FORWARD -i eth2 -m string --string="Copyright: GIAC Enterprises - not for publication" -
j DROP
#####################################################################

#####################################################################
# Implement All Rules
```

```
#####################################################################
echo "Implementing All Firewall Rules . . . . . "
echo "Configuring so RULE_X_Y_Z can be turned ON and OFF easily."
#direct to appropriate rule chain
# call RULE_X only for new connection attempts

$IPT -t filter -A INPUT -p tcp -m state --state NEW -j RULE_X
$IPT -t filter -A INPUT -p tcp -j RULE_Y
$IPT -t filter -A INPUT -j RULE_Z
$IPT -t filter -A FORWARD -p tcp -m state --state NEW -j RULE_X
$IPT -t filter -A FORWARD -p tcp -j RULE_Y
$IPT -t filter -A FORWARD -j RULE_Z

$IPT -t filter -A INPUT -p tcp -j tcprules
$IPT -t filter -A INPUT -p udp -j udprules
$IPT -t filter -A INPUT -p icmp -j icmprules
$IPT -t filter -A INPUT -p 50 -j esprules
$IPT -t filter -A INPUT -p 51 -j ahrules
$IPT -t filter -A INPUT -p 89 -j ospfrules
$IPT -t filter -A INPUT -p 9 -j igrprules
$IPT -t filter -A INPUT -p 47 -j grerules

$IPT -t filter -A FORWARD -p tcp -j tcprules
$IPT -t filter -A FORWARD -p udp -j udprules
$IPT -t filter -A FORWARD -p icmp -j icmprules
$IPT -t filter -A FORWARD -p 50 -j esprules
$IPT -t filter -A FORWARD -p 51 -j ahrules
$IPT -t filter -A FORWARD -p 89 -j ospfrules
$IPT -t filter -A FORWARD -p 9 -j igrprules
$IPT -t filter -A FORWARD -p 47 -j grerules

# turn on ip_forarding
echo 1 > $FWD
#####################################################################
```

Appendix E – Ipsec Configuration File ipsec.conf

```
#################################################
# /etc/ipsec.conf - FreeS/WAN IPsec configuration file
#################################################

# basic configuration
config setup
    # THIS SETTING MUST BE CORRECT or almost nothing will work.
    # this section applies to all connections
    # external interface of VPN gateway is eth1
    interfaces="ipsec0=eth1"
    # Debug-logging controls:  "none" for (almost) none, "all" for lots.
    klipsdebug=none
    plutodebug=none
    # Use auto= parameters in conn descriptions to control startup actions.
    plutoload=%search
```

```
    plutostart=%search

# defaults for subsequent connection descriptions
conn %default
    # define items that need to be applied to all connections
    # general defaults here can be subject to override later
    # How persistent to be in (re)keying negotiations (0 means very).
    keyingtries=0
    # means to authenticate gateways
    # Set up autokeying using IKE and RSA
    keyexchange=ike
    keylife=8h
    # authentication can be by shared secrets or digital signatures
    # digital signatures are superior in every way to shared secrets
    authby=rsasig
    # load all connection descriptions by default
    # can override this later with auto=start
    auto=add

    # PARTNERS
    # vpn tunnel for partners network
    # Tunnel mode, static IP, RSA key authentication
    # Here we just use ESP for both encryption and authentication
conn partners
    type=tunnel
    # declare identify for authentication
    leftid=182.16.0.2
    leftrsasigkey=0x0f985uf8w35w4t98qr...
    # optionally may place keys in dns
    leftrsasigkey=%dns
    # left security gateway (GIAC Enterprises) (public-network address)
    left=172.16.0.2
    # next hop to reach right (GIAC Enterprises router)
    leftnexthop=172.16.0.10
    # subnet behind left
    leftsubnet=192.168.1.0/24
    # declare identify for authentication
    rightid=172.18.0.2
    rightrsasigkey=0x05j4535o6drw76cf34ca...
    # optionally may place keys in dns
    rightrsasigkey=%dns
    # right security gateway (partners network) (public-network address)
    right=172.18.0.2
    next hop to reach left
    rightnexthop=172.18.0.10
    rightsubnet=192.168.100.0/24
    # SPI number
    spi=0x200
    # (manual) encryption/authentication algorithm and parameters to it
    esp=3des-md5-96
    espenckey=[192 bits]
    espauthkey=[128 bits]
    auto=start
```

```
# ROAD WARRIORS
# host on one end and a subnet (behind a security gateway) on the other
# This case is also called "road warrior"
# the road warrior is getting a dynamic ip address from an isp
conn road-warrior
    type=transport
    # declare identify for authentication
    leftid=182.16.0.2
    leftrsasigkey=0x0f985uf8w35w4t98qr...
    # optionally may place keys in dns
    leftrsasigkey=%dns
    # left security gateway (GIAC Enterprises) (public-network address)
    left=172.16.0.2
    # next hop to reach right (GIAC Enterprises router)
    leftnexthop=172.16.0.10
    # subnet behind left
    leftsubnet=192.168.1.0/24
    # accept any address for right
    right=%any
    # any address, provided authentication works
    rightid=0.0.0.0
    rightrsasigkey=0xd9a24765fe...
    #
    # no subnet for a typical road warrior
    # so the rightsubnet= parameter is omitted
    # SPI number
    spi=0x300
    # (manual) encryption/authentication algorithm and parameters to it
    esp=3des-md5-96
    espenckey=[192 bits]
    espauthkey=[128 bits]
    # let the road warrior start the connection
    auto=add
    # override the default retry for road warriors
    # we don't want to retry if IP connectivity is gone
    keyingtries=1
```

Appendix F – VPN Gateway Shell Script ipsec.rc

```
#!/bin/sh
#####################################################################

#####################################################################
# Script:       ipsec.rc
# Description linux iptables VPN startup script
# Author:       Mark E. Donaldson
# Date:         July 19, 2002
# Run Time:         run at boot time from:
#               /etc/init.d/rc.local or
#               /etc/init.d/boot.d or
#               /etc/init.d/rc3.d
```

```
#                ln -s /etc/init.d/iptables /etc/init.d/boot.d/S23ipsec
#####################################################################

#####################################################################
# ipsec.rc Script: Script Start
#####################################################################
# declare a trusted path
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/usr/local/bin
export PATH

# modify the IPT (iptables executable) variable to point to iptables binary
IPT=/usr/sbin/iptables

# set script variable name
SCRIPT_NAME="ipsec.rc"

# set variable for ip_forwarding
FWD="/proc/sys/net/ipv4/ip_forward"

# set variable for controlling number of state table entries
STATE_CONNECTIONS="/proc/sys/net/ipv4/ip_conntrack_max"

# set variable for tcp connection timeout
TIMEOUT="/proc/sys/net/ipv4/tcp_fin_timeout"

# set variable for tcp keep alive interval
KEEP_ALIVE="/proc/sys/net/ipv4/tcp_keepalive_intvl"

# set variable for route verification ( built-in egress & ingress filtering)
# drop spoofed packets which cannot be replied to on the received
# interface based on routing tables
# * = all interfaces
RP_FILTER="/proc/sys/net/ipv4/conf/*/rp_filter"

# set variable for ICMP broadcasts
ICMP_ECHO_BROADCASTS ="/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts"

# set variable for source routing
# * = all interfaces
ACCEPT_SOURCE_ROUTE="/proc/sys/net/ipv4/conf/*/accept_source_route"

# set variable for accepting ICMP redirects
# * = all interfaces
REDIRECTS="/proc/sys/net/ipv4/conf/*/accept_redirects"

# set variable for sending ICMP redirects
# * = all interfaces
MAKE_REDIRECTS="/proc/sys/net/ipv4/conf/*/send_redirects"

# set variables for each router interface IP address
ROUTE_ETH0="172.16.0.10"
ROUTE_ETH1="172.17.0.10"
```

```
# set variables for each vpn gateway interface IP address
VPN_ETH1="172.16.0.2"
VPN_ETH0="192.16.1.2"

# set variable for internal network
INT_NET="192.168.0.0/24"

# set variable for service network
SERVICE_NET="192.168.1.0/24"

# set variable for administrative network
ADMIN_NET="192.168.2.0/24"

# set variable for partners network
PART_NET="192.168.100.0/24"

#####################################################################
# Begin Execution
#####################################################################
# display firewall startup message
if [ -x /usr/bin/logger ];
then
    logger -p warning "Activating VPN gateway firewall script $SCRIPT_NAME "
    echo "Activating VPN gateway firewall script $SCRIPT_NAME "
fi

#####################################################################
# Link Modules As Needed
#####################################################################
# some modules we'll want to install if not compiled into kernel
# modprobe ip_tables
# Insert connection-tracking modules
# insmod ip_conntrack
# insmod ip_conntrack_ftp

#####################################################################
# Start Clean - Flush and Delete Chains & Rules
#####################################################################
# The following lines will flush (-F) all rules in the chain shown as the argument to -F.
# Since no chain name was given as an argument, -F will flush all chains:

for i in filter nat mangle
do
    echo "Flushing Filter Chains . . . . . $i table"
    logger -p warning "Flushing Filter Chains . . . . . "
    $IPT -t $i -F
    $IPT -t $i -X
done

#####################################################################
# Set Default and Static Routes
#####################################################################
echo "Setting default and static routes for server3 . . . . . "
```

```
logger -p warning "Setting default and static routes for server2 . . . . . "

# set default route for server3
route add default gw 192.168.1.1

# set static routes for vpn gateway
route add -net 172.16.0.0 netmask 255.255.0.0 dev eth1
route add -net 172.17.0.0 netmask 255.255.0.0 dev eth1
route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
route add -net 192.168.2.0 netmask 255.255.255.0 dev eth0

#####################################################################
# Set Critical Kernel Parameters
#####################################################################
# disable ip forwarding
echo 0 > $FWD

# set tcp connection timeout
# echo "30" > $TIMEOUT

# set tcp keep alive
# echo "1800" > $KEEP_ALIVE

# set up route verification ( built-in egress & ingress filtering)
# apply to all interfaces on machine

for f in $RP_FILTER;
do
    echo 1 > $f
done

# drop packets that are source routed
for f in $ACCEPT_SOURCE_ROUTE;
do
    echo 0 > $f
done

# drop ICMP redirect packets
for f in $REDIRECTS;
do
    echo 0 > $f
done

# deny sending of ICMP redirect packets
for f in /proc/sys/net/ipv4/conf/*/send_redirects;
do
    echo 0 > $f
done

#####################################################################
# Define Default Policy
#####################################################################
```

```
# default policy for firewall will be deny all unless explicitly permitted.
echo "Setting default policies . . . . . filter table"
logger-p warning "Setting default policies . . . . . filter table"
$IPT -t filter -P OUTPUT DROP
$IPT -t filter -P INPUT DROP
$IPT -t filter -P FORWARD DROP

echo "Setting default policies . . . . . nat table"
logger -p warning "Setting default policies . . . . . nat table"
$IPT -t nat -P PREROUTING DROP
$IPT -t nat -P OUTPUT DROP
$IPT -t nat -P POSTROUTING DROP

echo "Setting default policies . . . . . mangle table"
logger -p warning "Setting default policies . . . . . mangle table"
$IPT -t mangle -P PREROUTING DROP
$IPT -t mangle -P OUTPUT DROP
#####################################################################

#####################################################################
# Create IPsec Rules
#####################################################################
# allow IPsec
#
# allow IKE negotiations
$IPT -A INPUT  -p udp --sport 500 --dport 500 -j ACCEPT
$IPT -A OUTPUT -p udp --sport 500 --dport 500 -j ACCEPT

# allow ESP encryption and authentication
$IPT -A INPUT  -p 50 -j ACCEPT
$IPT -A OUTPUT -p 50 -j ACCEPT

# uncomment if AH authentication header used
# $IPT -A INPUT  -p 51 -j ACCEPT
# $IPT -A OUTPUT -p 51 -j ACCEPT

#####################################################################
# Create User Defined Chains
#####################################################################
echo "Creating User Defined Chains . . . . . "
logger -p warning "Creating User Defined Chains . . . . . "

# create a new chain that permits new connections on ipsec0
# but blocks everything else
$IPT -N block
$IPT -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A block -m state --state NEW -i eth1 -j ACCEPT
$IPT -A block -j DROP

# Jump to that chain from INPUT and FORWARD chains.
# $IPT -A INPUT -j block
# $IPT -A FORWARD -j block
```

```
#####################################################################
# VPN LOCKDOWN
#####################################################################
# permit all ssh connections from 192.168.0.3 only
$IPT -t filter -I INPUT -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j ACCEPT
$IPT -t filter -I FORWARD 2 -i eth0 -p tcp -s 192.168.0.3 --destination-port 22 -j ACCEPT

# permit syslog connections to syslog server on network
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p udp --destination-port 123 -j LOG --log-prefix "VPN NTP  "
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p udp --destination-port 123 -j ACCEPT

# permit syslog connections to NTP server on network
#$IPT -t filter -I OUTPUT -d 192.168.2.2 -p udp --destination-port 514 -j LOG --log-prefix "VPN
SYSLOG  "
$IPT -t filter -I OUTPUT 2 -d 192.168.2.2 -p udp --destination-port 514 -j ACCEPT

# permit connections to mail relay
$IPT -t filter -I OUTPUT -d 192.168.1.2 -p tcp --destination-port 25 -j LOG --log-prefix "VPN MAIL  "
$IPT -t filter -I OUTPUT 2 -d 192.168.1.2 -p tcp --destination-port 25 -j ACCEPT

# deny all other connections from remaining devices on internal network
$IPT -t filter -A INPUT -i eth0 -p tcp -s 192.168.0.0/24 -j DROP
$IPT -t filter -A FORWARD -i eth0 -p tcp -s 192.168.0.0/24 -j DROP

# deny all other tcp connections from remaining devices on service network
$IPT -t filter -A INPUT -i eth0 -p tcp -s 192.168.1.0/24 -j DROP
$IPT -t filter -A FORWARD -i eth0 -p tcp -s 192.168.1.0/24 -j DROP

# deny all other tcp connections from remaining devices on admin network
$IPT -t filter -A INPUT -i eth0 -p tcp -s 192.168.2.0/24 -j DROP
$IPT -t filter -A FORWARD -i eth0 -p tcp -s 192.168.2.0/24 -j DROP

# permit connections from router and block everything else
$IPT -t filter -I INPUT -i eth1 -p tcp -s 172.16.0.10 -j ACCEPT
$IPT -t filter -I FORWARD -i eth1 -p tcp -s 172.16.0.10 -j ACCEPT
#$IPT -t filter -A OUTPUT -p tcp -j LOG --log-prefix "BLOCKED TCP FROM VPN  "
$IPT -t filter -A OUTPUT -p tcp -j DROP
#$IPT -t filter -A OUTPUT -p udp -j LOG --log-prefix "BLOCKED UDP FROM VPN  "
$IPT -t filter -A OUTPUT -p udp -j DROP

# turn on ip_forwarding
echo "Turning on ip_forwarding . . . . . "
logger -p warning "Turning on ip_forwarding . . . . . "
echo 1 > $FWD
#####################################################################
```

Appendix G – Secure Network Backup Script

```
#################################################
# secure remote backup using tar and ssh
# secure_backup.sh
```

```
# script date July 8, 2002
##################################################

#!/bin/bash
# backup server2 to server3

cd /

list="home boot etc var opt/oracle"

for file in $list
do
# keep the previous backup
tgz=`echo $file | sed -e 's/\//_/g'`;
old="${tgz}.old";

# Store the old backup
ssh server3 "mv -f /storage/backup/server2/$tgz /storage/backup/server2/$old";
sleep 5m

# perform current backup
tar -cz $file/* | ssh server3 "cat - > /storage/backup/server2/$tgz";
sleep 5m
done
```

REFERENCES

Auditing & Policy Testing

1. Bauer, Mick. "Checking Your Work With Scanners Part I: namp." Linux Journal. May 2001.

2. Bauer, Mick. "Checking Your Work With Scanners Part II: nessus." Linux Journal. June 2001.

3. Fyodor. "The Art of Port Scanning." URL: http://www.insecure.org.

4. Goldsmith, David and Michael Schiffman. "Firewalking." Cambridge Technology Partners. October 1998.

5. Komarnitsky, Alek. Namp-web: "Port Scanning Made Easy." Sys Admin Magazine. October 2000.

6. McCarty, Ron. "A Look At ngrep." Sys Admin Magazine. May 2001.

7. Olson, Adam. "Working With SAINT." Sys Admin Magazine. March 2001.

8. Osser, William and Alex Noordegraaf. "Auditing in the Solaris 8 Operating Environment." Sun Microsystems, August 2001. URL: http://www.sun.com/blueprints.

9. Russel, Christopher. "Penetration Testing With dsniff." Sans Institute Reading Room. February 18, 2001.

10. Spitzner, Lance. "Auditing Your Firewall Setup." December 12, 2000. URL: http://www.enteract.com/~lspitz/rules.html.

Backup & Recovery

1. Curley, Charles. "Bare Metal Recovery." Linux Journal. November 2000.

2. Curley, Charles. "Linux Complete Backup and Recovery HOWTO." Revision 0.02. January 27, 2002.
URL: http://www.ibiblio.org/mdw/HOWTO/Linux-Complete-Backup-and-Recovery-HOWTO/index.html.

3. Preston, Curtis W. UNIX Backup and Recovery. Sebastipol, CA: O'Reilly & Associates, 1999.

Cisco Routers

1. Antoine, Vanessa et al. Router Security Configuration Guide. Ft. Meade, MD. National Security Agency. November 2001.

2. Brenton, Chris. "What Is Egress Filtering and How Can I Implement It?" Sans Institute Reading Room. February 29, 2000. URL: http://rr.sans.org/firewall/egress.php.

3. McQuerry, Steve. Interconnecting Cisco Network Devices. Indianapolis, IN: Cisco Press, 2000.

4. Hucaby, David and Steve McQuerry. Cisco Field Manual: Router Configuration. Indianapolis, IN. Cisco Press, 2002.

5. Sammut, Tim. "Little Known Cisco IOS Security Features." Sys Admin Magazine. December 2001.

6. Winters, Scott. "Top Ten Blocking Recommendations Using Cisco ACLs and Securing the Perimeter With Cisco IOS 12 Routers." Sans Institute Reading Room. URL: http://rr.sans.org/firewall/blocking_cisco.php.

DNS, BIND, Sendmail & Microsoft Exchange

1. Albitz, Paul and Cricket Liu. DNS and Bind. Sebastipol, CA: O'Reilly & Associates, 1998.

2. Bauer, Mick. "Hardening Sendmail." Linux Journal. April 2002. URL: http://linuxjournal.com/print.php?sid=5753.

3. Benedict, Stew. "Providing E-mail Services For A Small Office." Linux Journal. April 2001. URL: http://linuxjournal.com/article.php?sid=4539.

4. Blum, Richard. Sendmail For Linux. Indianapolis, Indiana: Sams Macmillan, 2000.

5. Costales, Bryan. sendmail. Sebastipol, CA: O'Reilly & Associates, 1997.

6. Fennelly, Carole. "Setting Up Sendmail On A Firewall Part 1." Itworld.com. April 1999. URL: http://www.itworld.com/Net/3314/swol-0499-security.

7. Fennelly, Carole. "Setting Up Sendmail On A Firewall Part 2." Itworld.com. May 1999. URL: http://www.itworld.com/Net/3314/swol-0599-security.

8. Fennelly, Carole. "Setting Up Sendmail On A Firewall Part 2." Itworld.com. June 1999. URL: http://www.itworld.com/Net/3314/swol-0599-security.

9. Jones, Dave. "Building An E-mail Virus Detection System For Your Network." Linux Journal. December 2001.
URL: http://linuxjournal.com/article.php?sid=4882.

10. Seneca, Eric Jorn. "sendmail: Introduction and Configuration." Linux Journal. November 2001.
URL: http://linuxjournal.com/article.php?sid=5507.

11. Stephens, Robin G. Guide To Securing Windows 2000 DNS. Ft. Meade, MD. National Security Agency. April 2001.

12. Walther, Jonathan. "Setting Up E-mail." Linux Journal. February 1998. URL: http://linuxjournal.com/article.php?sid=2516.

Firewalls & Proxy Servers

1. Bandel, David A. "A NATural Progression." Linux Journal. June 2002. URL: http://www.linuxjournal.com/article.php?sid=5839.

2. Bandel, David A. "Taming The Wild Netfilter." Linux Journal. September 2001.
URL: http://interactive.linuxjournal.com/Magazines/LJ89/4815.html.

3. Bandel, David A. "Netfilter 2: In the POM of Your Hands." Linux Journal. May 2002. URL: http://interactive.linuxjournal.com/Magazines/LJ97/5660.html.

4. Bauer, Mick. "Designing and Using DMZ Networks To Protect Internet Servers." Linux Journal. March 2001.

5. Berk, Vincent. "Improving Security, Accessibility and Maintainability of Websites Using Modified Reverse Proxy Servers: A Design and Implementation." Institute For Security Technology Studies. Dartmouth College. August 7, 2001.

6. Dean, Jeff. "Deploying Squid Part 1." O'Reilly Network. February 7, 2000.
URL: http://linux.oreillynet.com/pub/a/linux/2000/02/07/tutorial.html.

7. Dean, Jeff. "Deploying Squid Part 2." O'Reilly Network. March, 2000.
URL: http://linux.oreillynet.com/pub/a/linux/2000/03/10/tutorial.html.

8. Ferguson, P. and D. Senie. "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing." Network Working Group. RFC 2267.

9. Grennan, Mark. "Firewall and Proxy Server HOWTO." Revision 0.80. February 2000. URL: http://www.ibiblio.org/mdw/HOWTO/Firewall-HOWTO.html.

10. Hardin, John D. "Linux VPN Masquerade HOWTO." Revision 2.19. October 2000. URL: http://www.ibiblio.org/mdw/HOWTO/VPN-Masquerade-HOWTO.html.

11. Hasenstein, Michael. "IP Network Address Translation." 1997. URL: http://www.suse.de/~mha/linux-ip-nat/diplom/nat.html.

12. Hubert, Bert, Gregory Maxwell, Remco van Mook, Martin van Oosterhout, Paul B Schroeder, and Jasper Spaans. "Linux Advanced Routing & Traffic Control HOWTO." Revision 0.9.0. December 2001.URL: http://www.ibiblio.org/mdw/HOWTO/Adv-Routing-HOWTO.html.

13. Napier, Duncan. "IPTables/Netfilter – Linux's Next-Generation Stateful Packet Filter." Sys Admin Magazine. December 2001.

14. Pearson, Oskar. "Squid: A Users Manual." Sourceforge.net. URL: http://squid-docs.sourceforge.net/latest/html/book1.htm.

15. Ranch, David A. "Linux IP Masquerade HOWTO." Revision 2.00. April 2002. URL: http://www.ibiblio.org/mdw/HOWTO/IP-Masquerade-HOWTO/index.html.

16. Russell, Rusty. "Linux 2.4 NAT HOWTO." Revision 1.16. November 2001. URL: http://netfilter.samba.org/unreliable-guides.

17. Russell, Rusty. "Linux Networking Concepts HOWTO." Revision 1.13. July 2001. URL: http://netfilter.samba.org/unreliable-guides.

18. Russell, Rusty. "Linux 2.4 Packet Filtering HOWTO." Revision 1.22. November 2001. URL: http://netfilter.samba.org/unreliable-guides.

19. Russell, Rusty. "Linux Netfilter Hacking HOWTO." Revision 1.11. October 2001. URL: http://netfilter.samba.org/unreliable-guides.

20. Tapsell, John, Thomas Spellman, and Matthias Grimm. "Masquerading Made Simple HOWTO." Revision 0/07. February 2002. URL: http://www.ibiblio.org/mdw/HOWTO/Masquerading-Simple-HOWTO/index.html.

21. Vesperman, Jenifer. "Installing and Configuring Squid." O'Reilly Network. July 26, 2001. URL: http://linux.oreillynet.com/pub/a/linux/2001/07/26/squid.html.

22. Vesperman, Jenifer. “Using Squid On Intermittent Connections.” O’Reilly Network. August 9, 2001.
URL: http://linux.oreillynet.com/pub/a/linux/2001/08/09/authen_squid.html.

23. Vesperman, Jenifer. “Authentication and Squid.” O’Reilly Network. July 26, 2001.
URL: http://linux.oreillynet.com/pub/a/linux/2001/07/26/squid.html.

24. Vesperman, Jenifer. “Transparent Proxying with Squid.” O’Reilly Network. October 25, 2001. URL:
http://linux.oreillynet.com/pub/a/linux/2001/10/25/transparent_proxy.html.

25. Vesperman, Jenifer. “Peering Squid Caches.” O’Reilly Network. September 17, 2001.
URL: http://linux.oreillynet.com/pub/a/linux/2001/09/17/squidpeering.html.

26. Zwicky, Elizabeth D., Simon Cooper, and D. Brent Chapman. Building Internet Firewalls. Sebastipol, CA: O’Reilly & Associates, 2000.

Hacking

1. Russell, Ryan. et al. Hack Proofing Your Network. Rockland, MA: Syngress Press. 2002.

2. Silverman, Richard E. “Dsniff and SSH Reports of My Demise are Greatly Exaggerated.” O’Reilly Network.
URL: http://sysadmin.oreilly.com/news/silverman_1200.html.

3. CERT Advisory CA-2002-18 “OpenSSH Vulnerabilities in Challenge Response Handling.” June 26, 2002.
URL: http://www.cert.org//advisories/CA-2002-18.html.

4. Cert Advisory CA-1997-2 “BIND – The Internet Named Daemon.” May 26, 1998.
URL: http://www.cert.org/advisories/CA-1997-22.html.

5. Scambray, Joel, Stuart McClure, and George Kurtz. Hacking Exposed. Second Edition. Berkeley, CA: Osborne/McGraw-Hill. 2001.

6. Sloan, Joseph D. Network Troubleshooting Tools. Sebastopol, CA: O’Reilly Press. 2001.

Host Hardening

1. Bauer, Mick. “Battening Down The Hatches With Bastille.” Linux Journal. April 2001. URL: http://www.linuxjournal.com/article.php?sid=4547.

2. Noordegraaf, Alex and Keith Watson. "Solaris Operating Environment Security." Sun Microsystems, January 2000.
URL: http://www.sun.com/blueprints.

3. Spitzner, Lance. "Armoring Linux." September 19, 2000.
URL: http://www.enteract.com/~lspitz/linux.html.

4. Spitzner, Lance. "Armoring NT." April 16, 2000.
URL: http://www.enteract.com/~lspitz/nt.html.

Intrusion Detection

1. Bauer, Mick. "Intrusion Detection For The Masses." Linux Journal. July 2001.

2. Bush, Christopher. "Enhancing Network Security With tcp_wrapper." Sys Admin Magazine. September, 1999.

3. Chaubal, Ameet. "LIDS and Mandatory Access Control (MAC) On Linux." UNIX Review. June 2001.

4. Cinelli, Anthorny. "Port Sentry." Linux Journal. July 2001.

5. Dubrawsky, Ido. "Securing Solaris." Sys Admin Magazine. November 2000.

6. Dubrawsky, Ido. "PortSentry For Attack Detection Part One." SecurityFocus Online. May 15, 2002.
URL: http://online.securityfocus.com/infocus/1580.

7. Dubrawsky, Ido. "PortSentry For Attack Detection Part Two." SecurityFocus Online. May 19, 2002.
URL: http://online.securityfocus.com/infocus/1586.

8. Gaur, Nalneesh. "Snort: Planning IDS For Your Enterprise." Linux Journal. July 2001.

9. McCarty, Ron. "FCheck: A Solution To Host Based Intrusion Detection." Sys Admin Magazine. December 2000.

10. McCarty, Ron. "Intrusion Detection Strategies and Design Considerations." Sys Admin Magazine. September 1999.

11. McCarty, Ron. "Snort." Sys Admin Magazine. February 2000.

12. Olson, Adam. "Scaring Crackers Away With TCP Wrapper." Sys Admin Magazine. October 2000.

13. Ranum, Marcus J. "Coverage In Intrusion Detection Systems." NFR Security, Inc.

14. Ranum, Marcus J. "Experience Benchmarking Intrusion Detection Systems." NFR Security, Inc.

15. Roesch, Martin. "Snort – Lightweight intrusion Detection For Networks." URL: http://www.snort.org.

16. Roesch, Martin. "Snort Users Manual Snort Release: 1.8.2." October 16, 2002. URL: http://www.snort.org.

17. Spitzner, Lance. "Intrusion Detection." URL: http://www.enteract.com/~lspitz/ids.html.

18. Westphal, Kristy. "Snort – A Look Inside An Intrusion Detection System." Sys Admin Magazine. September 2000.

Logging and Syslog

1. Bauer, Mick. "swatch: Automated Log monitoring For The Vigilant But Lazy." Linux Journal. August 2001.

2. Cinelli, Anthorny. "Using Port Sentry and LogCheck." Sys Admin Magazine. March 2002.

3. Liberti, Leo. "Automating Firewall Log Scanning." Linux Journal. July 2001.

4. Martin, Michael J. "Using Cisco IOS Logging Part I." Serachnetworking.com. April 10, 2002. URL: http://searchnetworking.techtarget.com.

5. Martin, Michael J. "Using Cisco IOS Logging Part II." Serachnetworking.com. May 8, 2002. URL: http://searchnetworking.techtarget.com.

6. Spitzner, Lance. "Watching Your Logs." July 19, 2000. URL: http://www.enteract.com/~lspitz/swatch.html.

Linux/UNIX

1. Frisch, AEleen. Essential System Administration. Sebastipol, CA: O'Reilly & Associates, 1995.

2. Garfinkle, Simson and George Spafford. Practical UNIX & Internet Security. Sebastipol, CA: O'Reilly & Associates, 1996.

3. Nemeth, Evi et al. UNIX System Administration Handbook. Upper Saddle River, NJ: Prentice Hall. 2000.

NTP

1. Deeths, David and Glenn Brunette. "Using NTP To Control and Synchronize System Clocks Part I: Introduction To NTP." Sun Microsystems, July 2001. URL: http://www.sun.com/blueprints.

2. Deeths, David and Glenn Brunette. "Using NTP To Control and Synchronize System Clocks Part II: Basic NTP Administration and Architecture." Sun Microsystems, August 2001. URL: http://www.sun.com/blueprints.

3. Deeths, David and Glenn Brunette. "Using NTP To Control and Synchronize System Clocks Part III: NTP Monitoring and Troubleshooting." Sun Microsystems, September 2001. URL: http://www.sun.com/blueprints.

Security Policy

1. Spitzner, Lance. "Building Your Firewall Rule Base." January 26, 2000. URL: http://www.enteract.com/~lspitz/rules.html.

2. Weise, Joel and Charles Martin. "Developing A Security Policy." Sun Microsystems, December 2001. URL: http://www.sun.com/blueprints.

TCP/IP Protocols

1. Comer, Douglas E. Internetworking With TCP/IP Volume I: Principles, Protocols, and Architecture. Upper Saddle River, NJ: Prentice Hall. 2000.

2. Hall, Eric A. Internet Core Protocols: The Definitive Guide. Sebastipol, CA: O'Reilly & Associates, 2000.

3. Hunt, Craig. Linux Network Servers. Alameda, CA: Sybex, 1999.

4. Hunt, Craig. TCP/IP Network Administration. Sebastipol, CA: O'Reilly & Associates, 1994.

5. Shinder, Debra Littlejohn and Thomas W. Shinder. Troubleshooting Windows 2000 TCP/IP. Rockland, MA: Syngress Media.

6. Stevens, W. Richard. TCP/IP Illustrated Volume I: The Protocols." Reading, MA: Addison-Wesley. January 1999.

Virtual Private Networks, SSH, & IPsec

1. Barrett, Daniel J. and Richard E. Silverman. SSH The Secure Shell. Sebastipol, CA: O'Reilly & Associates, 2001.

2. Bauer, Mick. "The 101 Uses of OpenSSH: Part I." Linux Journal. January 2001. URL: http://www.linuxjournal.com/article.php?sid=4412.

3. Bauer, Mick. "The 101 Uses of OpenSSH: Part II." Linux Journal. February 2001. URL: http://www.linuxjournal.com/article.php?sid=4413.

4. Denker, John S. "Routing For Linux-IPSec." URL: http://www.quintillion.com/fdis/moat/ipsec+routing.

5. Galvin, Peter. "Enter the secure shell." Itworld.com. February 1998. URL: http://www.itworld.co.

6. Galvin, Peter. "More On Mastering the secure shell." Itworld.com. March 1998. URL: http://www.itworld.co.

7. Lasser, Jon. "Make SSH Do More." LinuxWorld.com. October 2001. URL: http://www.itworld.com.

8. Lesko, Matt. "Installing and Configuring OpenSSH." Sys Admin Magazine. October 2000.

9. Napier, Duncan. "Administering Linux IPSec Virtual Private Networks." Sys Admin Magazine. March 2002.

10. Napier, Duncan. "Introducing FreeS/WAN and IPSec." Sys Admin Magazine. November 2000.

11. Scott, Charlie, Paul Wolfe, and Mike Erwin. Virtual Private Networks. Sebastipol, CA: O'Reilly & Associates, 1999.