GIAC Certified Firewall Analyst (GCFW) Practical Assignment
Version 1.7

William S. Davis

August 28, 2002

Table of Contents:

1 GCFW Practical Assignment 1 – Security Architecture

1.1 Practical Scenario

GIAC Enterprises is an e-business that deals with the online sale of fortune cookie sayings. In its operations and business dealings, it interacts with individuals who require access to the company's computer network and resources. These individual users can be classified into five groups, with varying access requirements and restrictions. The five groups consist of the follow: Customers, companies or individuals that purchase fortune cookie sayings; Suppliers, companies that supply fortune cookie sayings; Partners, international companies that translate and resell fortune cookie sayings; GIAC Enterprise employees, who work at the office headquarters; and GIAC Enterprise employees working remotely, either from home as teleworkers, or around the globe as part of the mobile sales force.

The following defines a security architecture for GIAC Enterprises based on the needs of its customers, suppliers, partners and employees, and the security required to operate as a responsible online e-business.

1.2 Business Background

GIAC Enterprises is a start-up venture in an e-business sector that has seen surprising growth and activity over the past two to three years, namely the online sale of fortune cookie sayings. Many of the existing companies in this international market sector have converted to e-business in an attempt to streamline their operations and expand their market beyond the traditional "fortune cookie." As a result of high standards set down by brilliant pioneers early in this expansion into e-business, most companies have strived to meet these security standards required to conduct e-business, often hiring security specialists or outsourcing to security consultants. As a whole, the industry has gained a reputation for secure business transactions, as reflected by this market sectors' ability to maintain its stock valuation during a crises of confidence over corporate ethics by shareholders.

GIAC Enterprises enters the market at a time when venture capital funds are extremely tight, success in a large field of well-positioned competitors is more than uncertain, and high standards of business transactions are expected. However, continued growth and expansion of this market sector is anticipated in the age of global market economies and makes a compelling case for potential success.

The key to GIAC Enterprises' success will be its own well-positioned ties to non traditional customers, customers around the globe who are seeking to inject a positive outlook into everyday life by providing fortune cookie sayings in a variety of new methods, such as providing sayings for the open source software "fotd," which delivers personalized fortunes to computer users each day, web advertisements, email signatures, and other means of electronic communications and interactions.

As a result of these interests, the goals for the IT staff of GIAC Enterprises are:

- To provide a secure environment for conducting e-business that strives to meet industry standards
- To keep capital costs as low as possible for the near term

- Seek technical solutions that will scale as the company becomes established in the market and begins to expand its business operations
- To provide an efficient means of communicating with customers, suppliers and partners.
- If possible, within this framework, to meet the VISA standards of e-business

In pursing these goals, GIAC Enterprises was fortunate, pardon the pun, in hiring a recent graduate of the SANS Firewall Certification program whose credentials were promising and was willing, at least for the interim, to work at a lower salary and without the promise of pie-in-the-sky stock options.

The security architecture presented herein is the result of studying the proposed business operations, access requirements and system restrictions for the employees, suppliers and partners of GIAC Enterprises. (see Figure 1) Basically, the GIAC Enterprises network consists of four subnets. A perimeter network as the initial point of entry, a service network which contains the public services such as web and email systems, an internal network where GIAC Enterprises employee systems are located, and finally an internal protected network where the database is located.

1.3 Business Operations, Access Requirements and System Restrictions

The following describes the basic business operations and communications between GIAC Enterprises' customers, suppliers, partners and employees.

GIAC Enterprises commitment to security:

In order for any of the following business operations to take place between any of the four groups, customers, suppliers, partners and employees, GIAC Enterprises requires standard security precautions to be observed. The primary means by which all business interactions will take place are through email, web access, file transfers or direct connections.

Customers:

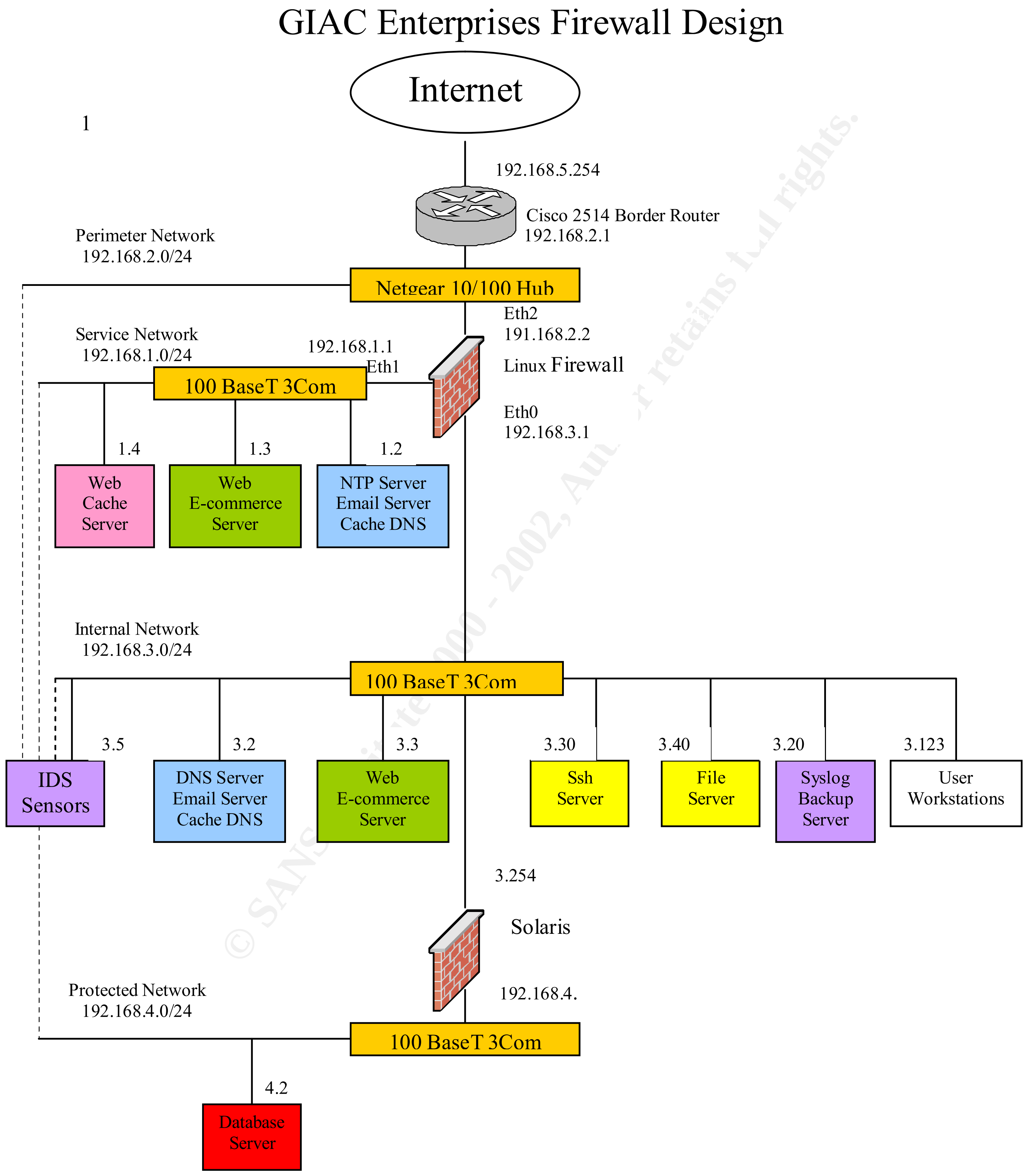
The customer base for GIAC Enterprises is international in scope, but predominately English language based, although the long-term business plan stresses the need for a more multilingual position, all interactions with customers will be in English.

The customer's needs are generally defined as follows:

- View company information and products
- Establish, review and update account profile
- Order company products
- Obtain order status
- Conduct financial transactions with the company
- Communicate with company personnel

In order to meet these needs, the customers will be able to obtain information from the company's static web site using the http protocol, conduct transactions via a secure e-commerce web site using the https protocol, and communicate by email with company personnel using the smtp protocol.

Figure 1



GIAC Enterprises Firewall Design
Internet
1
192.168.5.254
Cisco 2514 Border Router
192.168.2.1
Perimeter Network
192.168.2.0/24
Netgear 10/100 Hub
Eth2
191.168.2.2
Linux Firewall
Service Network
192.168.1.0/24
192.168.1.1
Eth1
100 BaseT 3Com
Eth0
192.168.3.1
1.4
1.3
1.2
Web Cache Server
Web E-commerce Server
NTP Server Email Server Cache DNS
Internal Network
192.168.3.0/24
100 BaseT 3Com
3.5
3.2
3.3
3.30
3.40
3.20
3.123
IDS Sensors
DNS Server Email Server Cache DNS
Web E-commerce Server
Ssh Server
File Server
Syslog Backup Server
User Workstations
3.254
Solaris
192.168.4.
Protected Network
192.168.4.0/24
100 BaseT 3Com
4.2
Database Server

All Customers will be required to authenticate any transactions with the company. This does not apply to accessing company or product information available at the static web site. The e-commerce web site will require the customer to authenticate with a username and password. Customer systems will also be required possess a client certificate in order to access the e-commerce web server. Any legal correspondence using email must contain a digital signature.

Usernames and passwords are established when the user applies for a new account. The user must supply a username, email address, mailing address before they receive a temporary password. GIAC Enterprises will create a client certificate and upon receipt, the customer can log into the e-commerce site and activate their account by supplying a new password that meets the criteria for good passwords, e.g. minimum 8 characters, with alphanumeric, upper and lower case, and at least one special character.

Customers are expected to obtain either their own company certificate or PGP key pair that will facilitate the use of digital signatures for either incoming or outgoing email.

Figure 2 shows the business flow for customer interactions. Note that customers are not allowed direct access to internal company systems, but are allowed access to systems on the service network only. In cases of account or order status information, the e-commerce server acts as a proxy in retrieving the relevant information from the protected database on the internal system, and presents it to the customer. Customers are not allowed to modify records directly. The e-commerce server forwards requested changes to internal personnel for final commitment to the database.

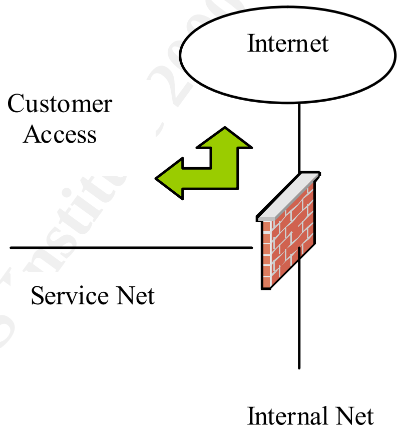


Figure 2

Suppliers:

Suppliers are companies that provide GIAC Enterprises with their fortune cookie sayings. Their needs are summarized as follows:

- Transfer fortune cookie sayings to GIAC Enterprises
- Create and access supplier account information and status

- Conduct financial transactions with GIAC Enterprises
- Communicate with GIAC Enterprises' personnel

There are several methods in which the suppliers can deliver their fortune cookie sayings to the company. One method is to include the sayings as an attachment to an email. However, the supplier is expected to encrypt and digitally sign the email. Furthermore, the maximum size of an email is three megabytes.

The second, and primary means, of delivering fortune cookie sayings will be through a secure file transfer into the internal network's file server utilizing the ssh protocol. This can be done either manually or automatically. Manual transfers are achieved by connecting to the internal ssh server and authenticating via the supplier's username and password. The ssh server has access to network directories for which the supplier is authorized. These directories are contained on a single internal file server.

Suppliers can conduct e-business transactions in the same manner as customers, creating and accessing their e-commerce account via username and password and their email must also be digitally signed.

Suppliers will have access to the service network, but will have limited access to the internal network via an ssh server for the sole purpose of uploading or downloading files. Access to the protected database will be through proxy via the e-commerce server and changes committed to the database by GIAC Enterprises personnel.

Figure 3 indicates the business flow for supplier transactions.

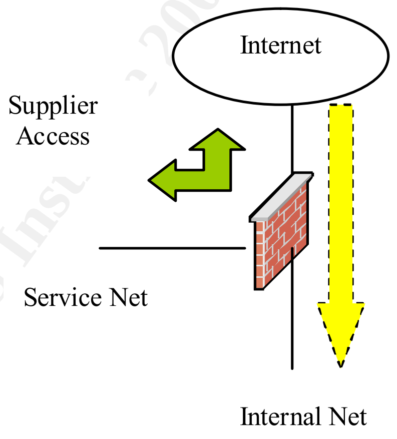


Figure 3

Partners:

Business partners with GIAC Enterprises are international companies that translate and resell fortunes, as well as those who are researching new ways to inject fortune cookie sayings into everyday life.

These partners have the following requirements:

- Transfer fortune cookie sayings to and from GIAC Enterprises
- Create and access partner account information and status
- Conduct financial transactions with GIAC Enterprises
- Communicate with GIAC Enterprises' personnel
- Participate in product research and development

Partners have the same access methods and requirements as suppliers, but with one major exception. Some partners need to have collaborative access to systems within the internal network for efforts aimed at improving translations, and new product development. Due to the wide-ranging interactions, the best method of granting access is to provide limited VPN access to the internal network. While it is anticipated partner interaction with GIAC Enterprises will increase in the future, the expense of a VPN concentrator is not warranted at present. VPN links will be created utilizing the more limited capabilities of ssh via the internal ssh server.

As an additional requirement, partners are required to obtain a security certificate from a reputable certificate authority. All ssh access and digital signatures will require the use of this certificate.

Figure 4 illustrates the business flow for partner interactions.

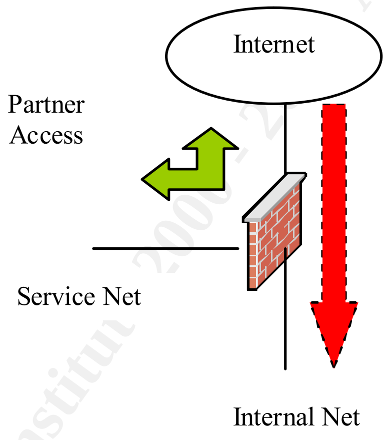


Figure 4

GIAC Enterprises Employees:

GIAC Enterprises employees have a wide variety of needs for computer resources. Functions that employees do need access to are as follows:

- Receive and send email
- Internet web access
- Update static web pages
- Update company database

Employees can generally be broken down into two major groups; those who work at the headquarters locations using workstations directly connected to the internal network, and those who work remotely, either telecommuting or as part of the mobile sales force, requiring remote access to the internal network.

Employees at GIAC Enterprises Headquarters:

Employees would be permitted use of various system resources based on their job function. They would be expected to sign and adhere to the company's acceptable use policy. System access would be through username and password authorization. Access to the company database would be through an internal database proxy server, or via an internal https web server. IT staff have access to all systems via ssh. File and print sharing is allowed as authorized by file permissions of the various work groups (departments).

Internet web access is allowed through a web proxy server. Email attachments are permitted, but any official correspondence to customers, suppliers or partners must be digitally signed. Furthermore, any sensitive information must be encrypted and digitally signed.

Figure 5 illustrates the business flow for partner interactions.

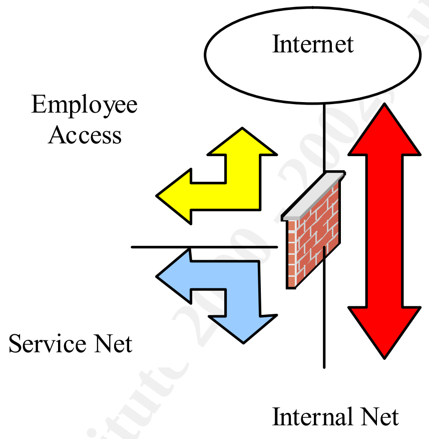


Figure 5

Employees Requiring Remote Access:

These employees need access to a wide range of internal services, depending on their job function. The sales force may need access to update or correct static web pages, check on order status, and modify the database. IT staff require full access to any system for purposes of monitoring or troubleshooting problems.

Rather than try to tailor these needs at the remote end, the access and restrictions of a particular employee should be part of the internal user's access controls as explained above. Remote systems are required to have anti-virus software and up to date virus definitions, personal firewall software, utilize a file system which allows file access controls (like NTFS or UFS), be capable of encrypting data files and creating digitally signed email. The primary means of connecting to the internal network would be through the ssh server, which would also establish VPN tunnels for receiving and sending email, as well as access to the internal web/e-commerce server and database proxy server.

Figure 6 illustrates the business flow for partner interactions.

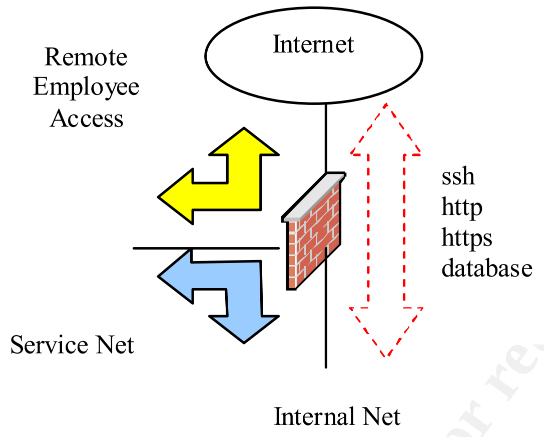


Figure 6

1.4 GIAC Enterprises Security Architecture:

The network design for GIAC Enterprises (Figure 1) is based on a common security architecture, dividing the overall network into three parts, a perimeter network, a service network and an internal network. An additional "protected" network is linked to the internal network to provide added security for the company's database. The following describes the components within this network design, their function and security role.

Border Router:

Brand/Model	Cisco 2514
Specs:	20MHz 68030, 16 MB Flash, 16MB DRAM
OS version	Cisco IOS 11.2
Interfaces:	2 serial, 2 10BaseT Ethernet, 1 Aux, 1 Console

The border router is the gateway from the GIAC Enterprises network to the Internet. A T1 link will provide a 1.5 megabit/second Internet connection. The device selected for the border router is an older model Cisco 2514 router. This model was selected because it is still well suited for a small to medium size business. Since it has two serial and two Ethernet ports, an additional T1 link could be added to provide increased capacity or a backup link via an auxiliary port utilizing a dialup modem. Only one ethernet interface will be used to connect to the perimeter network. Another major selling point was that it was cheaply available at auction.

The Cisco IOS v11.x is a commonly used operating system, is well supported, and has numerous books and documentation available. However, this software has reached end of life development.

Though bug fixes may be available for the near future, it will be advisable to upgrade to a Cisco 2600 series model when funds are available.

The function of the border router is to route legitimate traffic to and from the GIAC Enterprises network using static routes.

The security role that the border router plays is to do simple static filtering of incoming and outgoing traffic. The filtering, based on source and destination IP addresses as well as source and destination ports, will prevent some common types of attacks from occurring, and preventing illegal or spoofed addresses from being forwarded. Section 2.1 will address these issues in detail.

Perimeter Network:

Brand/Model	Netgear DS106
Specs:	6 port Auto 10/100BaseT Hub
OS version	Unmanaged
Interfaces:	1 uplink/port, 5 port

The perimeter network utilizes a Netgear dual speed 10/100 BaseT unmanaged hub. Any device attached to this hub will be exposed to the Internet, with the only protection being the border router's filtering, or the device's host security.

The function of the hub is to connect the border router to the firewall, or other attached devices.

The reason an unmanaged hub was selected was that it was purely a hardware solution, with no access to the device over the network. This avoids attacks against this device using either http or snmp protocols. A hub was selected over a switch, so that an IDS sensor could be plugged in and listens to all traffic traversing the hub (see IDS Sensors below). The decrease in speed of a hub versus a switch is mitigated by the fact that the border router ethernet interface is limited to 10BaseT. At the time when the Cisco 2514 is upgraded to a Cisco 2600, the hub should be replaced with a switch.

An additional reason for creating a perimeter network as opposed to just a strait link between the router and firewall components was to allow for expansion to a dedicated VPN device, such as a Cisco 3005 VPN concentrator. This VPN component would plug into the hub, and directly into an interface card on the firewall component. The border router would then be able to separate traffic destined for this VPN gateway from normal firewall traffic. After the VPN traffic was decrypted, the packets would then be forwarded to the firewall for filtering. This VPN component was considered an important aspect of future company capabilities, but at present, due to cost limitations, its implementation will be deferred.

Firewall:

Brand/Model	Dell 2500
Specs:	Dual Pentium III 1GHz, 1GB RAM, 32GB Disk
OS version	Redhat Linux v7.3, Kernel 2.4.18 (recompiled monolithic)
Software:	Iptables v1.2.5-3, OpenSSH v3.1p1-3

The firewall component selected is an Intel based server running the Redhat Linux 7.3 operating system and utilizing the Linux Kernel v2.4.18, and iptables v 1.2.5-3 stateful packet filtering software. The firewall component will have three interfaces. One interface connects to the perimeter network, a second connects to the service network, and a third interface connects to the internal network.

The firewall will act to filter and control connections between three network segments, the Internet, the service net, and the internal net.

The firewall is a key component to the security architecture and serves three main roles; a router, a packet filter, and intrusion detection. As a router, it directs traffic to and from the more vulnerable service network and separately, and to and from the important internal network. Its packet filtering and stateful inspection capabilities will be explained in the following firewall policy and tutorial (see sections 2.3 and 2.5). The default stance of the filtering will be to deny everything. The firewall policy, based on the business operation needs, will determine what services and systems will be able to communicate with one another. Finally, due to iptables' extensive logging capability, the log files can be an important aspect in determining the effectiveness of the security architecture, as well as providing a resource to aid in the detection of probes and illicit attacks against the network.

An additional aspect worth mentioning about the Linux Kernel is the ability to recompile the kernel to include the iptables modules as part of the bootable kernel, and to disable further loading of kernel modules. This is significant in that it increases host security on one of the key modules of the security architecture by addressing the risk of kernel root kits. Kernel root kit attacks, if successful, are extremely difficult to detect, since normal scans for modified files using programs such as tripwire, would not detect any changes, since the operating system itself would be able to return false information. (1)

Service Network:

Brand/Model	3Com Superstack Switch 3300
Specs:	24 port Auto 10/100BaseT Switch
OS version	Managed, 3Com Network Supervisor http interface
Interfaces:	24 ports total, 1 uplink/port, 1 monitor port for IDS

The service network is a screened subnet of the GIAC Enterprises network. Components of the service network are connected via a 3-Com Switch 3300 10/100 BaseT ethernet switch. This switch is a managed switch with both http and snmp access. A feature of note is the ability to define one port as a monitoring port, allowing it to see all traffic on another switch port. This older switch was selected because of its availability at auction. It is still a fully usable component and not in need of upgrade.

The switch controls connections between components, reducing traffic load by allowing full duplex connections between components.

The security role that this switch plays is in its ability to limit the traffic seen on the network. If offers some protection against packet sniffers, though recent attacks of this nature have tended toward distributed packet sniffers. It also plays a role in intrusion detection by allowing an IDS sensor to listen to all traffic on the service network by monitoring the port connected to the firewall interface. (See IDS Sensor below)

Service Network Servers:

Brand/Model	PC clone
Specs:	CPU <1GHz, 512-1024 MB RAM, <32GB Disk
OS version	Redhat Linux v7.3, Kernel 2.4.18 (recompiled monolithic)
Software:	Iptables v1.2.5-3, OpenSSH v3.1p1-3, other - varies depending on service

All servers located in the service network will be using PC clone computers running RedHat Linux 7.3.

The function of these servers will vary based on which of the network services they provide, but will include one or more of the following service protocols; dns, http, https, ntp or smtp.

These servers will run with recompiled kernels to preventing the use of loadable kernel modules (see Firewall above.) Iptables will be run on each server, providing some additional, but minimal, filtering protections and logging for these servers, due to their exposure to access from the internet. Host security on these systems will also include Tripwire file integrity checking software, McAfee Anti-virus software, ssh, for remote administration or access from internal systems, tcp wrapper, for added access control of ssh service access, and network time protocol (ntp) clients, for network system log synchronization. System logging will utilize a service network logging server.

Email Gateway (SMTP):

The email service, using the smtp protocol, runs Sendmail v 8.11.6 software.

This email server will act as a relay for email to and from the Internet, forwarding all legitimate email through the firewall component to an email server located on the internal network. This enables a tight firewall policy on email, allowing email destined for the internal network to only be permitted through if it comes from the service net email server.

The email server will serve two important security functions. One will be to filter incoming and outgoing email for viruses utilizing the AmaVis filter utilizing the McAfee virus definitions. (2) These virus definitions will be automatically updated when new definitions are posted on McAfee's web site via a perl language script written for this purpose. A second function will be to filter out spam or other obnoxious email through content filtering using the Spam Assassin software (spamassassin.sourceforge.net). Filtering out such emails can help prevent attacks based on social engineering, as well as conserve company resources.

Email is used as a common method to deliver an attack, allowing the attacker to inject a virus, worm or Trojan program into the computer system. To meet these concerns, all incoming and outgoing email will be virus scanned by the email gateway. Email may also be intercepted to obtain company secrets, or in the other extreme, denied that it was ever sent. In order to provide both protection of sensitive company information, emails containing such information must be encrypted. Additionally, all official email correspondence to or from the company involving financial transactions must be digitally signed for reasons of non-reputability.

Domain Name Service (DNS):

The DNS service runs Bind v9.2.0-8 software, using a split DNS configuration.

DNS is an essential service to all networks, translating fully qualified domain names (fqdn) into numeric IP addresses for transport layer operations. The service network DNS server is a caching and forward only DNS server, providing name to IP address resolution for service network systems. It also acts as a "proxy" for the internal DNS server. The internal DNS server (see below) splits off the internal network, providing resolution of internal names, while forwarding requests for external names to the service network DNS server.

One of the security roles this provides is to prevent internal network information from being publicly available, making it more difficult for an attacker to enumerate the internal network. Since it is not an authoritative server, the only public information will be two systems listed with the ISP's DNS server, the public email and web servers. Though this is somewhat of a security through obscurity defense, it is only one of many layers. The split DNS configuration can also provide protection against DNS cache poisoning, and zone transfer attacks.

Network Time Protocol (NTP):

NTP is accessed via the ntpd and ntpdate software provided with RedHat Linux (rpm ntp-4.1.1-1).

NTP synchronizes a computer system's clock to UTZ time broadcast across the Internet. The service network ntp server acts as the lowest stratum timeserver for both the service and internal networks, as well as the border router.

Clock synchronization is extremely important in intrusion detection and incident analysis. It allows the correlation of logged events throughout the network.

Web (Http):

The web service uses the Apache v1.3.23-11 web server software.

The web service provides a vital function for the company, providing information about the business organization and products or services that it provides. It also acts as the initial method in which customers could register to conduct e-commerce transactions with the company.

The web server utilizes static web pages and has no access direct access to the internal network. Since the company's image is projected by the web pages available on the web server, they are a target for malicious activity, defacing the web pages by replacing them with the attackers' own versions. As an added protection against such an attack, the web server is a mirror of an internal web server. The static web pages are uploaded to the external web server on a regular basis via ssh. If compromise and defacement did occur, tripwire, a file integrity checker, would detect the changes, and the pages would be reset to their original content, minimizing the damage. However, the vulnerability that allowed the compromise would still have to be determined and eliminated.

The only form input on the web server would be one allowing the customer to request an e-commerce account with GIAC Enterprises. This form would be subject to strict limitations on the character set and variable size to prevent Unicode or buffer overflow attacks against the web server.

Web e-commerce server (https):

The e-commerce server would also utilize the Apache v 1.3.23-11 web server, and the https protocol.

The e-commerce server provides a means for customers, suppliers and partners, to enter and update account profile information in a secure maner, as well as create, update or display orders and their current status.

The e-commerce server is another vital key of the company's security architecture. This service deals with sensitive information including credit card details, ordering information, or other personal details that require safeguarding. The https protocol provides a secure means of communication, using encryption, between the server and the client. Customers are required use a https capable web browser and to authenticate using a username and password. Customers, suppliers and partners must also provide a client side security certificate for authentication purposes. This provides a higher level of authentication, since customers have access to company products, and suppliers or partners have access to more sensitive company information.

Additionally, the e-commerce server acts as a proxy for requests made to the database via the e-commerce web interface, using access that is limited to read-only. Requested changes are forwarded internally to GIAC Enterprises employees for commitment to the database from the internal network.

The e-commerce server can be run as a second instance of the Apache web server on the same system as the http web server. They can share file resources and directories, but access control lists and directory parameters can limit access and require higher levels of encryption and authentication, effectively separating access between the two web servers. (3)

Web Cache server:

The web caching service uses Squid v2.4.STABLE6-1.7.2 and acts as an outbound proxy in handling requests from internal web browsing clients to web servers on the Internet. It additionally acts as a cache, enabling quicker retrieval of frequently requested web pages.

The role of the web cache server is to provide an additional layer of protection between the internal client and the web server it is trying to connect to. By interceding as a proxy, the squid service hides the actual internal addresses and can provide additional filtering or checks on http, ftp and gopher based requests and responses. It can filter on URL's, keywords and http commands. Additional software, add-ons, can be used to file out banner advertisements. It should be noted that the web cache server would simply pass through encrypted traffic of SSL/https sessions. It cannot filter what it cannot see.

System log server:

One system acts as the syslog server for the entire service net and the border router.

The syslog server acts as a main repository for logged events, accepting log messages from other systems. The syslog file is transferred into the internal syslog server each morning to provide a secure archive of activity in the service network.

The role of the syslog file is vital in the scheme of intrusion detection. It alerts the system administrator to unusual behavior and possible compromises. Although the syslog file might be modified or during a system compromise to cover the tracks of the attacker, often previous log files will provide some trace of their initial reconnaissance or compromise attempts.

Internal Network:

Brand/Model	3Com Superstack Switch 3300
Specs:	24 port Auto 10/100BaseT Switch
OS version	Managed, 3Com Network Supervisor http interface
Interfaces:	24 ports total, 1 uplink/port, 1 monitor port for IDS

The internal network is the second screened subnet of the GIAC Enterprises network. Components of the service network are connected via a 3-Com Switch 3300 10/100 BaseT Ethernet switch. This switch is a managed switch with both http and snmp access. A feature of note is the ability to define one port as a monitoring port, allowing it to see all traffic on the switch. It is still a fully usable component and not in need of upgrade.

The switch controls connections between components on the internal network, reducing traffic load by allowing full duplex connections between components.

The security role that this switch plays is in its ability to limit the traffic seen on the network. If offers some protection against packet sniffers. It also plays a role in intrusion detection by allowing an IDS sensor to listen to all traffic on the service network. (See IDS Sensor below)

Internal Network Components:

Brand/Model	PC clone
Specs:	CPU <1GHz, 512-1024 MB RAM, <32GB Disk
OS version	Redhat Linux v7.3, Kernel 2.4.18 (recompiled monolithic)
Software:	Iptables v1.2.5-3, OpenSSH v3.1p1-3, other - varies depending on service

The internal network is composed of PC clones running one of two operating systems, either Windows 2000 or RedHat Linux v7.3. The Linux servers are used to provide services to the internal users, or by IT staff for system administration and monitoring purposes.

Linux servers providing services are configured in the same method the service network servers are configured. The Linux workstations used by the IT staff are similarly protected, but have loaded tools and software necessary for system administration purposes.

The Windows 2000 systems are not allowed to run services, other than print and file sharing, and have Norton Anti-virus software installed. Since incoming email is scanned using McAfee virus definitions, Norton Anti-virus was selected to provide a second layer of defense against virus infection, since Windows systems are a common target. IP is the only supported network protocol allowed. Additionally, the EventReporter (www.eventreporter.com) software is used to provide logging of events to the Unix based log server.

Internal Email gateway (SMTP):

The internal email gateway runs Sendmail v8.11.6 on a Linux server.

The internal email server is the primary mail server for all internal user accounts. Any incoming mail is delivered to the user in a shared mail directory on the file server. Outgoing email is relayed to the external email server for delivery if the email address is outside the GIAC Enterprises domain. Mail destined for another internal email address is delivered directly by the internal email server.

As with the external email server, all email is virus checked, and scanned for spam. This redundancy allows for checking of internal only emails that might spread a virus that infected the system through other means. It also allows email sent from remote systems, via an ssh tunnel (see section 2.4), to be checked. In both these cases, the external email would never see the traffic. As mentioned above, the last line of defense would be the host anti-virus software.

Internal DNS:

The internal DNS server runs Bind v9.2.0-8 on a Linux server.

The internal DNS server resolves names for internal network systems only. Internal names are stored in its zone file, but zone transfers are not allowed. Recursive lookups are allowed, but the only server it can forward requests to is the external DNS server.

By splitting the internal name resolutions from all other lookups, this keeps the internal names secret, preventing enumeration activity by attackers. This also limits DNS traffic to traffic between the two DNS servers, making for simpler firewall rules to handle this protocol. Since DNS is a critical service, it is often used either to attack networks through vulnerabilities in Bind, or as a covert channel of communication by Trojan programs, since DNS is normally allowed through the firewall.

Internal NTP:

NTP is accessed via the ntpd and ntpdate software provided with RedHat Linux (rpm ntp-4.1.1-1).

The internal NTP server looks to the external NTP server for it's time synchronization, and in turn acts as the timeserver for only systems on the internal network.

As mentioned above, time synchronization is important in correlating log files for purposes of intrusion detection and incident analysis.

Internal Web server (http):

The web service uses the Apache v1.3.23-11 web server software.

The internal web server is the primary production web and e-commerce server. Its static web pages and scripts are mirrored out to the external web server on a regular basis. Internal only pages and scripts are also allowed, but not mirrored. The internal web server is accessible from any internal system or remotely linked host. Its files are accessible by only those whose job function requires making changes or modifications to the web site.

The advantage of having an internal web server that it is partly mirrored to the service network web server and e-commerce server is twofold. As mentioned above, it can help in quickly recovering from a defacement of the public web site. Another advantage is that it provides a means by which remote users logged into the internal network can access and modify the database directly if they are authorized to do so. Authentication is by username and password, and uses ssl for all sessions, making remote access safe. Partners using the server must supply a client certificate as part of their authentication.

Web e-commerce server (https):

The e-commerce server would also utilize the Apache v 1.3.23-11 web server, and the https protocol.

The internal e-commerce server provides GIAC employees and partners read/write access to the internal database. The server also acts as a database proxy, making queries to the database as specified from forms or scripts on the e-commerce server.

This internal server separates any write activity to the database from anywhere other than the internal network or remote VPN tunnels.

Ssh VPN server:

The internal ssh server runs OpenSSH v3.1.1p1-3 on a Linux server.

The ssh server acts as a VPN server, allowing remote users running the ssh client to establish tunnels for receiving (Imap) and sending (smtp) mail, accessing the internal web server (https), or directly accessing the database via the database proxy listener. File access is allowed using secure file transfer (scp) from files located on the internal file server. Finally, partners and employees are able to further connect to other internal systems running OpenSSH server using an ssh client located on the ssh VPN server.

The ssh server is key to allowing supplies, partners, and employees access to internal resources in a limited and controlled manner. Suppliers are allowed access only to files located on the internal file server, they can access and make changes to their account profile via the internal e-commerce web site, but are not allowed to receive or send email via the internal email server. Partners have the same privileges and restrictions as suppliers, except they are allowed access to other internal systems and to access the database via internal web server scripts. Employees have the same access as partners, but are allowed to send and receive email as well as run specialized database applications using a database proxy tunnel. It should be noted that direct access to the database from the ssh server is not permitted by the internal firewall component (see below.)

Internal file server:

Linux Server using the Unix File System (UFS)

The file server functions as a limited file sharing system for interacting with suppliers, partners and employees. The file system contains common directories into which may be uploaded or downloaded information and products required to carry out business functions. These directories can only be access via ssh and scp from internal system only.

System log server:

The system log server makes use of the syslogd software on a Linux server.

The system log server acts as the central receptacle for all logging activity. Only systems in the internal network log directly to the log server.

The system log server plays an important role in detecting malicious activity as well as normal service and system failures. By concentrating all logging to one server, it is possible to correlate system responses to attacks or failures. The service network syslog files are transferred daily using ssh to the internal log server.

Backup Server:

Backup software using ufsdump v 0.4b27-3 on a Linux server.

The backup server is instrumental in creating full and incremental backups of all UFS directories. These backups enable the restoration of the various network components to a known safe state in the event of catastrophic failure or attack.

Backups are the last line of defense in the event of total security failure. If made correctly, they can provide a snapshot of a component in a known safe state. If the system must be rebuilt, this provides a quicker procedure than rebuilding the component from scratch.

Since company policy is to have all critical files located on the file server, or other UFS servers, the Windows systems are not backed up on a daily basis. However, disk images are created using Norton

Ghost to preserve user profiles and the installed software base to provide for quick recovery should a system fail or be compromised.

Note: Because of the more static nature of the service network systems, the configuration files are maintained on the administrative workstation which is backed up by this server. Service network systems are rebuilt rather than backed up because of their simple nature, and limited function.

User Workstations:

The primary platform for user workstations is Windows 2000.

Window 2000 systems are located either on the internal network or used as remote access systems, such as laptops, or home computers, though Linux systems are encouraged. These workstations allow the majority of the employees to engage in the business operations assigned to them.

Host security plays an important role for these systems, as Windows oriented software is often the target of virus, worm and Trojans programs. As mentioned above, Windows systems are not allowed to run Internet services such as IIS, and are limited to IP transport protocols but including NetBUIE over TCP. Group membership and file shares are to be strictly controlled.

The few IT staff linux workstations serve a similar role, but are required to have greater security as mentioned above in Internal Components due to their role as remote administrative consoles.

Protected Network:

Brand/Model	3Com Superstack Switch 3300
Specs:	24 port Auto 10/100BaseT Switch
OS version	Managed, 3Com Network Supervisor http interface
Interfaces:	24 ports total, 1 uplink/port, 1 monitor port for IDS

Although there is limited use of this network, an managed 10/100BaseT hub is used to provide the full speed access database queries can require, and the desire to access a monitor port for intrusion detection.

The switch provides links between the firewall component and the database server.

The switch plays a role in intrusion detection, allowing an IDS sensor to be plugged into a port that monitors all traffic on the firewall's protected network interface. This can be used to verify that the strict rules of the firewall component are succeeding in blocking unauthorized traffic.

Internal protected firewall:

Brand/Model	Sun Sunfire V100
Specs:	CPU 500MHz, 256K eCache, 512MB RAM, 40MB disk
OS version	Solaris 8
Software:	IP Filter v3.4.29, ssh v3.1

The internal firewall runs IP Filter v3.4.29 packet filtering software on a Sun SunFire V100 system using Solaris 8 as its operating system. Sunfire V100 are not current models and are available at closeout prices or at auction.

The internal firewall isolates the jewels of the company, the production database, from the internal network. It filters the traffic and limits access to only those systems that need access to the database, and provides a log of the connections made.

Isolating the database provides a greater level of control of access than if the database server were located directly on the internal network. The additional filtering can be extremely strict due to the limited nature of the traffic this system will see. Only the ssh, ntp, and database protocols will be allowed through. By logging all initial connections, the system can provide a ledger of access activity.

The reason IP Filter (coombs.anu.edu.au/~avalon/), a stateful packet filtering software package, running on a Sun hardware system was selected was to prevent a single vulnerability in the firewall software to allow access throughout the network. By utilizing a different platform, running a different set of software for filtering, an attacker would have to penetrate two distinct operating systems using different kinds of filtering software. Since the database contains the most important information and files, this is a prudent additional step to take.

Another interesting aspect is a fail over capability using IP Filter on two identical SunFire systems by uploading the state tables to the hot spare system. This component could be added in the future to provide redundancy in case of system failure.

Database server:

A PC clone running Redhat Linux 7.3 and MySQL v3.23.52 open source database software.

The MySQL database (www.mysql.com) stores the company's most important data and files, including customer, supplier, partner and employee information, as well as company products and financial transactions.

Host security is still critical. No other services other than NTP, ssh and the MySQL listener protocol should be allowed to run. The host and MySQL software should be securely locked down. Carefully configured MySQL users limit the access to the various database tables available on the system.

By allowing through only the MySQL protocol from specific systems and users, access to the database can be severely limited and strictly controlled.

IDS Sensors:

IDS sensors are simple Redhat Linux 7.3 servers running Shadow (www.nswc.navy.mil/ISSEC/CID/) or Snort (www.snort.org) IDS software and have at least two Ethernet interfaces and located on the internal network.

An IDS sensor is used to watch the traffic on the network, forwarding what it sees to the system log server for analysis.

The IDS sensor logs provide an important means of detecting unusual, unexpected or illicit activity on a network segment. While they are often located behind the firewall component, it is helpful to have them placed in the various segments of the overall network when trouble shooting or analyzing attacks against the network. It is not necessary to have a sensor at every point, it is only necessary to be able to plug to each network segment as desired. It is advisable to maintain one sensor on the internal network at all times, using an additional NIC as a means of verifying firewall rulesets.

Monitoring traffic is possible by adding an additional network interface card (NIC) that connects to a port that can see all passing traffic, such as the monitoring ports on the 3Com 3300 switches. The interface card itself is not assigned an IP address and is put into promiscuous mode, enabling it to listen, but not respond. Since there is no assigned IP address, the NIC can be connected to any segment at any time as desired.

2. Assignment 2 – Security Policy and Tutorial

2.1 Network addresses

GIAC Enterprises has obtained access to a class B network address space. The class B network space will be sub divided into the four subnets as follows:

- Border router external IP address 192.168.5.254/32
- Perimeter net 192.168.2.0/24
- Service net 192.168.1.0/24
- Internal net 192.168.3.0/24
- Protected net 192.168.4.0/24

NOTE: These networks use a private Class B address space and would not be used under normal conditions. These network addresses are purely for the purposes of this assignment. It is unlikely that a Class B address space would be allocated to a startup company. It would be necessary to use network address translation (NAT) to maintain this addressing scheme, subnet a smaller block of ip addresses using networks with 29 bit masks (6 useable IP addresses) for the perimeter and protected networks, and either 28 or 27 bit masks (14 or 30 useable IP addresses) for the service and internal networks, or some combination of NAT and subnetting.

2.2 Border Router Policy

The border router policy is based on the Cisco guidelines for securing IOS software (4) and the NSA Router Security Configuration Guide (5). Additional useful references are listed below. The configuration removes unneeded risky services and provides access control lists that limit access to the router itself, as well as filtering packets based on source and destination ip addresses or ports. The full configuration file is listed in Appendix A.

www.pasadena.net/cisco/secure.html
www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scoverv.htm
www.cisco.com/univercd/cc/td/doc/product/software/ios11/index.htm
http://nsa2.www.conxion.com/cisco/guides/cis-2.pdf
http://www.cert.org/tech_tips/packet_filtering.html

Securing Access:

To begin with, password encryption should be enabled so that passwords stored on the router are encrypted. If anyone were able to obtain a configuration listing, it would not display the passwords in plain text. In addition, the enable password should use the "secret" encryption scheme that uses a better encryption scheme. (6)

```
service password-encryption
enable secret 5 $1$Z43R$9GbRR8BewJ.bTwAbExlm54
```

The console login can also be configured to require a password to gain interactive access to the router. Remote access via the terminal lines (0 – 4) should also require a password. It is wise to limit what system can telnet into the terminal lines by using an access control list (see below).

```
line con 0
 password 7 018A44D2195430B1284
 login
line aux 0
 no exec
line vty 0 4
 access-class 10 in
 password 7 09374B62534A32472C
 login
```

Finally, all logins should encounter a message indicating unauthorized access is not allowed:

```
banner motd ^C WARNING: Authorized Access Only ^C
```

Disable unneeded services or limit use:

The services that should be disabled are the Cisco Discovery protocol (CDP), the tcp-small-servers (echo, discard, chargen and daytime), the udp-small-server (echo, discard, and chargen), finger, pad, bootp, http, snmp and ip classless or source routing. These services are often utilized in Denial of Service attacks or information gathering queries. (6) Finally, if domain name lookups are not required, disable DNS for the router.

```
no cdp run
no service tcp-small-servers
no service udp-small-servers
no service finger
no service pad
no service bootp
no service http
no service snmp
no ip classless
no ip source-route

no ip domain-lookup
```

The specific interfaces of the router should also have certain services or abilities disabled. These include icmp redirects, unreachable and mask reply response messages, directed broadcasts, and proxy-arp. If the interface is not being used to synchronize to an NTP server, it should disable NTP.

```
no ip redirects
no ip unreachable
no ip direct-broadcasts
no ip mask-reply
no ip proxy-arp
ntp disable
```

NTP can be used to help synchronize logging, but should be limited to access a trusted NTP server on a query basis only from a single interface. Use the following commands to prevent the router being used as a timeserver for other systems (master server) and specify the server to synchronize to, as well as which interface will be used. Specifically disable ntp in the other active interface as mentioned above. Note: the router creates the ntp clock-period entry when it synchronizes to the NTP server.

```
no ntp master
ntp clock-period 17179873
ntp server 192.168.1.2 source Ethernet1
```

Logging:

The router should be permitted to log to another syslog server, since memory is often limited. This service (UDP 514) will need to be permitted in the firewall policy for just the border router to the syslog server.

```
logging 192.168.1.2
```

Access Control Lists:

There are three ACLs in the border router configuration. One is a standard ACL, filtering on just source or destination address. The others are extended ACLs, filtering on addresses and port numbers.

Access list 10 is a standard ACL, and defaults to an implicit deny everything. This ACL is used to limit remote access to the router for telnet logins. As a result, the only permit statement in the list is to permit traffic from the system administrator's system on the internal network.

```
access-list 10 permit 192.168.3.20
```

Access list 101 is an extended ACL that is applied to filter incoming traffic on the external interface of the router. The first five statements deny incoming traffic from private or illegal addresses. It should be pointed out that the statement to deny network 192.168.0.0 was not included here due to the use of the 192.168 network for the purposes of this assignment.

```
access-list 101 deny   ip 10.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 172.16.0.0 0.15.255.255 any log
access-list 101 deny   ip 169.254.0.0 0.0.255.255 any log
access-list 101 deny   ip 127.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 0.0.0.0 0.255.255.255 any log
```

The next two lines deny incoming traffic from Class D multicast addresses and Class E reserved addresses.

```
access-list 101 deny   ip 240.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 224.0.0.0 0.255.255.255 any log
```

Access list 10 is applied to the terminal lines with the statement:

```
access-class 10 in
```

The next four statements deny incoming traffic to the external interface that is supposedly coming from GIAC Enterprises networks, or the router's external IP address. This is most likely spoofed source addresses and should be blocked.

```
access-list 101 deny   ip 192.168.1.0 0.0.0.255 any log
access-list 101 deny   ip 192.168.2.0 0.0.0.255 any log
access-list 101 deny   ip 192.168.3.0 0.0.0.255 any log
access-list 101 deny   ip 192.168.4.0 0.0.0.255 any log
access-list 101 deny   ip host 192.168.5.254 any log
```

The next series of statements deny specific traffic on known ports that you want to protect or are known problem ports. Port 1523 is used as a database listener port on the internal database. There is no reason there should be systems accessing this port from outside the GIAC Enterprises networks. It is specifically blocked to add a layer of protection. Port 69 is the tftp protocol, used to update the router's operating system or configuration. There should be no access from external systems. There may be occasional need to upload patches, but this would be done using the internal interface. This would not be routine and would require temporary modification of the firewall policy to allow such traffic. Tftp should be disabled on the router at all other times. Ports 135-139 and port 445 are Windows NetBios related traffic. No windows protocols are allowed between the GIAC Enterprises networks and the Internet. This also blocks a large percentage of the network scanning that goes on as a result of malicious activity. Note that these connection attempts are not logged. The default firewall policy also denies and logs this activity. It is further recommended by CERT that UDP 87 (link) be blocked because of its common use by intruders, as well as UPD/TCP 111 & 2049 (SunRPC & NFS), TCP 512, 513 & 514 (Unix "r" cmds), TCP 515 (lpd), TCP 540 (uucpd), TCP/UDP 2000 (openwindows), and TCP/UDP 6000 (X windows) be blocked because of chronic problems with these services. (7) These rules are implemented here partly to reduce the load on the firewall, as well as protect the firewall server. Finally, snmp traffic is blocked as an added caution, even though the snmp service has been disabled on the router.

```
access-list 101 deny   ip any any eq 1523 log
access-list 101 deny   ip any any eq 69 log
access-list 101 deny   udp any any eq 87 log
access-list 101 deny   ip any any eq 111 log
access-list 101 deny   ip any any eq 135
access-list 101 deny   ip any any eq 136
access-list 101 deny   ip any any eq 137
access-list 101 deny   ip any any eq 138
access-list 101 deny   ip any any eq 139
access-list 101 deny   ip any any eq 445
access-list 101 deny   tcp any any eq 512 log
access-list 101 deny   tcp any any eq 513 log
access-list 101 deny   tcp any any eq 514 log
access-list 101 deny   tcp any any eq 515 log
access-list 101 deny   tcp any any eq 540 log
access-list 101 deny   ip any any eq 2000 log
access-list 101 deny   ip any any eq 2049 log
access-list 101 deny   ip any any eq 6000 log
access-list 101 deny   ip any any eq 6001 log
access-list 101 deny   udp any any eq snmp log
access-list 101 deny   udp any any eq snmptrap log
```

Finally, all other traffic to GIAC Enterprises networks are permitted, before the final catch all statement explicitly denies all other traffic even though this is the default policy of extended lists.

```
access-list 101 permit ip any 192.168.1.0 0.0.0.255
access-list 101 permit ip any 192.168.2.0 0.0.0.255
access-list 101 permit ip any 192.168.3.0 0.0.0.255
access-list 101 permit ip any 192.168.4.0 0.0.0.255
access-list 101 deny   ip any any log
```

Access list 101 is applied to the external serial interface #0 with the line:

```
ip access-group 101 in
```

The last list, access list 102 is applied to traffic outgoing from the external interface. The primary purpose here is to prohibit traffic originating from the GIAC Enterprises networks from sending out illegal source addresses, preventing spoofing. Only traffic with source addresses from GIAC Enterprises is permitted out. All other traffic is denied and logged.

```
access-list 102 permit ip 192.168.1.0 0.0.0.255 any
access-list 102 permit ip 192.168.2.0 0.0.0.255 any
access-list 102 permit ip 192.168.3.0 0.0.0.255 any
access-list 102 permit ip 192.168.4.0 0.0.0.255 any
access-list 102 deny   ip any any log
```

Access list 102 is applied to the external serial interface #0 with the line:

```
ip access-group 102 out
```

2.3 Firewall Policy

Netfilter is packet filtering software capable of maintaining state, e.g. connection tracking, and stateful inspection. The program, which sets up, maintains, and views the filter rules, is called iptables. "Netfilter" and "iptables" are often used interchangeably to refer to the Linux firewall software and from this point forward will simply be referred to as "iptables." However, it is worth pointing out the relations between the various components.

According to the home page of the netfilter/iptables project (www.netfilter.org):

> *netfilter is a set of hooks inside the linux 2.4.x kernel's network stack, which allows kernel modules to register callback functions called every time a network packet traverses one of those hooks.*
>
> *iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists out of a number of classifiers (matches) and one connected action (target).*
>
> *netfilter, iptables and the connection tracking as well as the NAT subsystems together build the whole framework.*

The rules implementing the firewall policy are installed using the iptables command line interface. Together, these commands can be aggregated into a single script, often called rc.firewall. When this script is executed, each iptables command loads a firewall rule into the filter table.

Iptables is based on the concept of "chains" of rules. This is similar to the concept of a computer programming language with a main program and subroutines. A chain is a set of rules that are tested for matching packets from top to bottom. If a packet matches the rule's conditions, then the rule's action will dictate what happens to that packet. If no rules match, than either the default policy applies, or examination of the packet returns to the next rule of the "calling" chain.

Iptables has three built-in chains that are the starting point for all traffic. These are the INPUT, OUTPUT and FORWARD chains. The INPUT chain refers to packets that are destined for an interface installed in the firewall server. The OUTPUT chain deals with packets originating from an interface installed in the firewall server. Computers with only one network interface would only deal with packets incoming or outgoing from that interface. If more than one interface is present, than the third chain, FORWARD, filters traffic that is forwarded between the interfaces. This assumes that IP forwarding has been enabled on the system, allowing the computer to act as a router.

An additional type of chain is called a "user-defined" chain. These chains provide a means of organizing filtering rules. If the built-in chains are considered as a "main program," than the user-defined chains are the subroutines called by these built-in chains.

Collectively, these chains provide the rules that implement the firewall policy.

2.3.1 Ruleset optimization

The intent of optimizing the firewall ruleset is to increase the speed at which packets in the data stream are processed by the filter. Not only does greater efficiency increase the throughput, but also decreases the chance that the firewall will become overburdened and simply drop or forward packets without subjecting them to the firewall rules. There are three factors which effect the efficiency of the filter processing; the number of rules installed in the kernel, the chain traversal length, and the total number of match tests that must be performed. To address these issues, it is best to begin with rules that block specific traffic, such as problem ports associated with specific vulnerabilities, and anti-spoofing rules. (8)

The use of "state" allows a connections' traffic, once established, to bypass the filtering rules, dramatically increasing throughput. Iptable's stateful inspection ability allows traffic related to an established connection, but using a different protocol, to also quickly exit the rule checking process.

In general, it helps to place the most frequently used TCP service rules near the top of the chains followed by UDP rules. Although testing UDP packets against TCP rules first may slow down UDP processing, it should not be significant, with a possible exception being streaming media. Icmp traffic is infrequent and can be placed toward the bottom of the chains. Using iptable's multiport mode, a method for including several ports for different services in one rule, reduces the number of rules needed. (8)

Greater organization can be achieved through user-defined chains. Rather than simply testing a packet against each rule from top to bottom, until a match is finally made, different traffic flow patterns and protocols can be quickly discerned, and the checked against rules in a specialized chain for a quicker match.

The ruleset defined below uses the above principles in an attempt to optimize the filtering process.

2.3.2 Iptables syntax

Before starting in on the description of the firewall ruleset, it is necessary to provide the rudiments of the iptables syntax. For a complete description of the syntax, refer to either the iptables man page, or the excellent reference Linux Firewalls.(8) Additionally, there are a number of web sites that deal specifically with iptables. As mentioned above, the URL http://www.netfilter.org is the home page of the netfilter/iptables project. It contains current status, security alerts and links to numerous tutorials, HOW-TO's and FAQ's relating to Netfilter. It is a site worth visiting.

The iptables command line has four basic part and has the form:

```
 iptables [-t table] command [match] [target/jump]
```

The table field indicates which one of three available tables will be used for the rule. The default table is the "filter" table, used for general packet filtering. Two additional tables are the "nat" table, used for network address translation, and the "mangle" table, used for changing packet headers and advanced filtering. In the ruleset defined below, only the filter table will be used, so none of the rules will show the –t field.

The command field tells iptables what to do with the rest of the information on the command line. An entry of "-A FORWARD", appends this rule to the FORWARD chain. Some frequently used commands are append (-A), delete (-D), flush (-F), insert (-I), list (-L), create new chain (-N), policy (-P) and replace (-R). The chain name affected follows the command.

Most of these command, or the following match flags have a single character or full text identifier. For example, the append command is denoted by either "-A" or "--append". A single dash precedes the single character, while a double dash precedes the full text identifier. (See below.)

The match section of the iptables command line is used to define the criteria by which a packet will be matched to the rule. These match operations can be further broken down into five categories as follows (9):

- Generic matches available to all rules
- TCP matches applied against tcp protocol packets only
- UDP matches applied against udp protocol packets only
- ICMP matches applied against icmp protocol packets only
- Explicit matches which require the –m (or –match) option, such as state, or multiport

These matches allow you to identify and isolate specific kinds of packets that are either acted upon immediately or directed to more specialized chains. Some of the more commonly used matches are:

-s	or --source	the source ip of the packet
-d	or --destination	the destination ip of the packet
--sport	or --source-port	the source port of the packet
--dport	or --destination-port	the destination port of the packet
-p	or --protocol	specific protocol e.g. tcp, udp, icmp
--syn		SYN flag set in tcp packet
-i	or --in-interface	incoming packets on specified interface
-o	or --out-interface	outgoing packets on specified interface

For a complete listing of these match options, refer to the iptables man page, or to Chapter 3 of Linux Firewalls (8).

The final section of the command line determines what action is to take place on the matched packet. This action or "jump" (as indicated by the "–j") can be either a direct action, such as ACCEPT, DROP, or REJECT, or can be a call to a user-defined chain or logging module for further filter processing.

2.3.3 Ruleset Definition and Policy Explanation

The following discusses the firewall policy and it's implementation in the ruleset. The full rc.firewall ruleset script is listed in Appendix B. It should be noted that this ruleset was loosely based on firewall script detailed in Chapter 5 Firewall Optimization of Linux Firewalls (r8), which was written for a simple host with one Ethernet interface.

The rc.firewall script can be divided into three parts. The first part defines variables used in the script, initializes iptables and creates the users-defined chains. The second part populates the user-defined chains with rules. The third part is the logical starting point for the filtering process and contains the rules for the built-in chains and the initial calls to the user-defined chains.

TIP: Remember that this is an executable script. In some ways it is similar to JavaScript, in that before a subroutine can be called, it must first be defined. So the user-defined chains must be created and populated first, before the built-in chains can reference them. Also, you may add in additional commands, unrelated to iptables commands that may assist in debugging. For example, you can use the echo command to indicate what portion of the script has been executed before an error occurs, and include extensive comments describing the purpose of each rule and chain.

2.3.3.1 Iptables initialization

The first section of the rc.firewall script defines constants that equate ip addresses or ports numbers to a name. The constants are used throughout the script and make it easy to modify the script as changes occur to the ip address assignments. For example, if the ip address of the DNS nameserver changes, rather than having to locate all occurrences of that ip address, only the constant definition needs to be changed. Refer to Appendix B for a listing of these constants. As noted above, this listing uses the private Class B lan address range of 192.168.x.x. This is solely for the purposes of this paper.

The last constant, USER_CHAINS, is an array of the names of the user-defined chains that will be used in the rc.firewall script and is listed below:

```
USER_CHAINS="INT-input              INT-output \
            EXT-input              EXT-output \
            SVC-input              SVC-output \
            ext-to-int             ext-to-svc \
            int-to-ext             svc-to-ext \
            int-to-svc             svc-to-int \
            tcp-state-flags        log-tcp-state \
            conn-track             ssh-connect \
            ntp-query              auth-query \
            dns-query              squid-query \
            source-address-check   ext-source-check \
            int-source-check       svc-source-check \
            dest-address-check     db-query \
            www-query              svc-mail \
            syslog-messages        \
            icmp-in                icmp-out"
```

These user-defined chains and their purpose will be explained in the following pages.

The next portion of the initialization section enables various kernel modules support for packet checks. Some of these may be redundant when compared with the border router ACLs, but the impact on system performance will be minimal if the router is working correctly. If the router is compromised, this provides a secondary line of defense.

The first command enables the system to act as a router and forward packets between interfaces.

```
# Enable ip forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Next, the kernel is told to ignore icmp echo-requests messages sent to the network broadcast address. This helps protect against smurf attacks or network scanning which seeks to elicit a response from all systems on the network by using the network's broadcast address.

```
# Enable broadcast echo Protection
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Source routing is a method by which a packet can be told a specific path, or route, to take from one point to another on the Internet. The reply will follow the reverse of the indicated route. An attacker can use this technique to snoop traffic as it passes through a network on a route he selected. Additionally, source routing may be implemented separately from the IP forwarding module. In the case of a proxy firewall, which turns off forwarding, source routing may enable traffic to be passed from one interface to another, allowing a connection to be established. In general, it is best to turn off source routed packets. (10)

```
# Disable Source Routed Packets
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
```

```
done
```

TCP SYN cookies provide a measure of defense against a denial of service attack from a SYN flood. As the connection queue reaches capacity, rather than sending a SYN-ACK response, the system sends a SYN cookie and clears the queue slot. A SYN cookie is a cryptographic challenge protocol that enables legitimate users to continue to connect. (11) If the system receives a valid response to the SYN cookie in a short period of time, the request is valid and a connection is established.

```
# Enable TCP SYN Cookie Protection
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

ICMP redirect messages inform a host to change its routing table to a more optimum gateway. This can be used as a denial of service attack or a means of sniffing rerouted traffic and should be disabled.

```
# Disable ICMP Redirect Acceptance
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done

# Don¹t send Redirect Messages
for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
    echo 0 > $f
done
```

Packets can be crafted to have a source address that is different from the actual source address. This is known as spoofing. The following prevents spoofing from being used to elicit response that would go out on an interface that differed from the interface that received the request.

```
# Drop Spoofed Packets coming in on an interface, which if replied to,
# would result in the reply going out a different interface.
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done
```

Finally, the ability to log packets that contain impossible, non-routable addresses is activated. It is very useful for tracking down misconfigured systems, as well as detecting illicit behavior.

```
# Log packets with impossible addresses.
for f in /proc/sys/net/ipv4/conf/*/log_martians; do
    echo 1 > $f
```

The final portion of the first section initializes the three tables, namely the filter, nat and mangle tables. Note the filter table is not named because it is the default. Although the nat and mangle tables are not used in this firewall design, it does not hurt to make sure they are cleared. By flushing the chains, all current rules are removed. If this were not done, the rules added by this script would be appended to the chains causing potential rules conflicts or logic errors.

```
# Remove any existing rules from all chains
iptables --flush
iptables -t nat --flush
iptables -t mangle --flush
```

Since flushing the rules does not change the default policy (see below), it is important to allow the local loopback interface to accept all traffic, allowing local services to function, in case the default policy is drop everything.

```
# Activate traffic on loopback interface
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

The chain policy sets the default stance that the firewall ruleset takes. It is a best practice to maintain a default policy of drop everything and accept only that that is specifically allowed(8). By setting this policy on the three built-in chains, INPUT, OUTPUT and FORWARD, you insure that if a certain service is not covered by the rules, or worse, the ruleset does not load correctly, the traffic will not go through.

```
# Set the default policy to drop
iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP
```

Finally, any user-defined chains are deleted in case they already existed, before creating them, otherwise an error would occur.

```
# Remove any pre-existing user-defined chains
iptables --delete-chain
iptables -t nat --delete-chain
iptables -t mangle --delete-chain

#Create the user-defined chains
for i in $USER_CHAINS; do
    iptables -N $i
done
```

2.3.3.2 Starting the filtering process

Rather than continue to Part 2 of the rc.firewall script and explain the purpose of each of the user-defined chains, it helps to see how the filtering process starts and how the initial built-in chains pass the packets on to the user-defined chains for more specific filtering. Thus, the following section explains the rules defined in the third part of the rc.firewall script. (See Part 3 of Appendix B)

Checks common to all traffic:

Before passing off the filtering process to more specialized user-defined chains, it makes sense to check for conditions that are either critical, or common to a large portion of the packets. If they can be quickly handled, then throughput can be increased.

The very first check performed by the filtering process looks for illegal TCP flag conditions in packets using the TCP protocol. Each of the built-in chains, INPUT, OUTPUT and FORWARD are listed since the arriving packet is automatically assigned to a built-in chain based on its traffic flow. Traffic coming into, out of, or forwarded to/from any interface is checked. If the traffic does not adhere to the

TCP protocol rules, then there is no reason to process them. They are most likely crafted packets with malicious intent and are dropped.

These first three checks test all tcp packets against the rules in the user-defined chain named tcp-state-flags. If there are no matches found, then the packet is considered normal, and filtering proceeds with the next rule in the built-in chain.

```
# If TCP: Check for stealth scans or illegal flag combinations
iptables -A INPUT -p tcp -j tcp-state-flags
iptables -A OUTPUT -p tcp -j tcp-state-flags
iptables -A FORWARD -p tcp -j tcp-state-flags
```

If a packet can be determined to be part of, or related to, an existing connection that has already been accepted by the ruleset, there is no need to check it again. This is the advantage of a state inspection capable firewall. By maintaining a connection state, the remaining traffic that is part of/related to the connection, once established, can be quickly accepted, and forego additional filter processing. This greatly speeds up throughput. Again, each type of traffic flow must be checked via the built-in chains. The filter processing is passed to the user-defined conn-track chain, which identifies it as part of an existing connection, or traffic related to a connection, and accepts the packet. If it does not match, the packet returns for further rule testing.

```
# Check for established connections
iptables -A INPUT -j conn-track
iptables -A OUTPUT -j conn-track
iptables -A FORWARD -j conn-track
```

At this point, the packet is considered to be either a new connection request, or a type of protocol that cannot be tracked and must be subjected to the rigorous rules that will allow it to be accepted, or dropped.

Since the traffic appears normal, the source and destination addresses are checked to determine if they are invalid non-routable private addresses, or spoofed addresses. If the addresses are invalid, the packet is dropped. Otherwise, it continues with further filer processing.

Each built-in chain, INPUT, OUTPUT and FORWARD, must test source addresses in a slightly different way for each of the firewall interfaces. As a result, packets are checked in special user-defined chains tailored to their specific traffic flows. In the section below, each FORWARD chain is further identified by the input and output interfaces. The "-A FORWARD -i $EXTNIC –o $INTNIC" matches traffic being forwarded that comes in on the external interface and is sent out on the internal interface. This is traffic coming in from the Internet and destined for the internal network. This traffic is checked against filter rules in the ext-source-check chain. The rules are repeated, matching on non-tcp traffic (! tcp) and tcp traffic with the SYN flag set.

This filter process is done for each of the possible combination of traffic flow, e.g. external to internal, internal to external, external to the service net, service to external, service to internal and internal to service, as well as the standard input and output on the firewall server's interfaces themselves.

```
# Test for illegal source and destination addresses in incoming packets
iptables -A INPUT -p ! tcp -j source-address-check
iptables -A INPUT -p tcp --syn -j source-address-check
iptables -A OUTPUT -p ! tcp -j source-address-check
iptables -A OUTPUT -p tcp --syn -j source-address-check
iptables -A FORWARD -i $EXTNIC -o $INTNIC -p ! tcp -j ext-source-check
iptables -A FORWARD -i $EXTNIC -o $INTNIC -p tcp --syn -j ext-source-check
iptables -A FORWARD -i $EXTNIC -o $SVCNIC -p ! tcp -j ext-source-check
iptables -A FORWARD -i $EXTNIC -o $SVCNIC -p tcp --syn -j ext-source-check
iptables -A FORWARD -i $SVCNIC -o $EXTNIC -p ! tcp -j svc-source-check
iptables -A FORWARD -i $SVCNIC -o $EXTNIC -p tcp --syn -j svc-source-check
iptables -A FORWARD -i $SVCNIC -o $INTNIC -p ! tcp -j svc-source-check
iptables -A FORWARD -i $SVCNIC -o $INTNIC -p tcp --syn -j svc-source-check
iptables -A FORWARD -i $INTNIC -o $EXTNIC -p ! tcp -j int-source-check
iptables -A FORWARD -i $INTNIC -o $EXTNIC -p tcp --syn -j int-source-check
iptables -A FORWARD -i $INTNIC -o $SVCNIC -p ! tcp -j int-source-check
iptables -A FORWARD -i $INTNIC -o $SVCNIC -p tcp --syn -j int-source-check
```

Destination address checks are more strait forward, checking only for illegal or private addresses, and thus do not depend on the interface involved. Three rules call the same dest-address-check chain, once for each of the built-in chains.

```
#
iptables -A INPUT -j dest-address-check
iptables -A OUTPUT -j dest-address-check
iptables –A FORWARD –j dest-address-check
```

Needless to say, if all the filter checks were applied in the above manner, it would be difficult to follow the logic and debug the ruleset. This is where the organizational strength of the user-defined chains can be applied. By separating these traffic flows into user-defined chains, it is easier to follow the logic and implement the firewall policy.

Primary user-defined chains:

So, it is at this juncture that the primary user-defined chains are first utilized. These primary level chains are defined based on the traffic flow, and are similar in concept to the built-in chains. For example, a chain named ext-to-int filters the traffic that is forwarded from the external interface to the internal network interface. A chain named int-to-ext filters traffic flowing in the reverse direction. Rules are placed in these chains for services that the firewall policy specifies for that traffic flow. As a simple example, the int-to-ext chain could include a rule to check for ssh connections originating from the internal network, whereas the ext-to-int chain would not even check for ssh connections, disallowing inbound ssh connections by default.

To take this one step further, rules relating to a specific service, such as ssh or a protocol such as icmp, can be aggregated into a specialized user-defined chain. The primary chain may call on this secondary chain to perform additional checks specific to a service. This enables similar rules to be concentrated in one chain, simplifying the logic and debugging. In the example above, the int-to-ext chain would call upon the ssh-connect chain to make the final determination of accepting or dropping the ssh traffic.

The result of this design is the use of six chains to handle incoming packets, three for forwarded traffic and three for input on the firewall server's interfaces. An additional six chains handle the outgoing packets, again, three for forwarded traffic and three for the firewall's interfaces.

```
# Standard Checks for incoming packets
iptables -A FORWARD -i $EXTNIC -o $SVCNIC -j ext-to-svc
iptables -A FORWARD -i $SVCNIC -o $INTNIC -j svc-to-int
iptables -A FORWARD -i $EXTNIC -o $INTNIC -j ext-to-int
iptables -A INPUT -i $EXTNIC -d $EXTADDR -j EXT-input
iptables -A INPUT -i $SVCNIC -d $SVCADDR -j SVC-input
iptables -A INPUT -i $INTNIC -d $INTADDR -j INT-input

# Standard Checks for outgoing packets
iptables -A FORWARD -i $SVCNIC -o $EXTNIC -j svc-to-ext
iptables -A FORWARD -i $INTNIC -o $SVCNIC -j int-to-svc
iptables -A FORWARD -i $INTNIC -o $EXTNIC -j int-to-ext
iptables -A OUTPUT -o $EXTNIC -s $EXTADDR -j EXT-output
iptables -A OUTPUT -o $SVCNIC -s $SVCADDR -j SVC-output
iptables -A OUTPUT -o $INTNIC -s $INTADDR -j INT-output
```

The rules in the previous section dealing with address checking could be simplified and placed into each of the above primary chains. Rather than having the following rules located before the primary user-defined chains,

```
iptables -A FORWARD -i $EXTNIC -o $INTNIC -p ! tcp -j ext-source-check
iptables -A FORWARD -i $EXTNIC -o $INTNIC -p tcp --syn -j ext-source-check
```

the rules would look as follows and be located within the ext-to-int chain.

```
iptables -A ext-to-int -p ! tcp -j ext-source-check
iptables -A ext-to-int -p tcp --syn -j ext-source-check
```

Logging dropped packets:

The final portion deals with logging and dropping of all remaining traffic. Initially, it is helpful to log all packets that are dropped to help you determine that the ruleset is working as expected. As the ruleset becomes stable, certain types of traffic may be commonly dropped. If the traffic is well known and fully understood, such as NetBIOS broadcasts from the internal network, it may be advisable to remove it from the firewall log file. Reducing the volume of logged events is helpful in spotting traffic that is truly unusual, as well as an inducement to regular log file reviews.

The first rules below, select specific known traffic that is silently dropped.

```
# Specific drops of known traffic that you do not want to include in the log file
# Internal network NetBIOS broadcasts and telnet scans
iptables -A INPUT -i $INTNIC -p tcp --dport 139 -j DROP
iptables -A FORWARD -p tcp --dport 23 -j DROP
iptables -A INPUT -p tcp --dport 23 -j DROP
```

The next rules first log, then drop any remaining traffic that has not matched any prior rule. In essence, this is the default drop policy that was installed at the beginning of this script. It may be redundant, but provides that extra level of security to make certain the firewall policy is upheld.

```
# Log anything else that defaults to drop
iptables -A INPUT -j LOG --log-prefix "Default drop: "
iptables -A INPUT -j DROP
iptables -A OUTPUT -j LOG --log-prefix "Default drop: "
iptables -A OUTPUT -j DROP
iptables -A FORWARD -j LOG --log-prefix "Default drop: "
iptables -A FORWARD -j DROP
```

TIP: A comment on the syntax of the logging rule above is in order. One of the nice features of iptables is the ability to tag log entries with a prefix, in this case "Default drop." Any packet logged by this rule will show up in the syslog message file (/var/log/messages) with the specified prefix embedded at the beginning of the log line (see section 2.5l). This prefix can be used as a search string to locate similar types of dropped packets in an otherwise large log file. Patterns that might otherwise go unnoticed may be detected by searching for a prefix such as "Bad Guy #1" from the firewall log files for the past month. A "low and slow" scan might show up that has been ongoing from Bad Guy #1 for a long period of time.

2.3.3.3 User-defined Chains

In part 2 of the rc.firewall script, the rules are loaded into the various user-defined chains. As mentioned above, these chains are either referenced from the built-in chains, or from the "primary" user defined chains based on traffic flow.

User-defined chains common to all traffic:

The first two user defined chains, tcp-state-flags and log-tcp-state, are used to check for illegal TCP flag combinations.

A commonly seen port scans often use the SYN/FIN combination. It was initially used to slip by static packet filters that checked and logged packets with only the SYN flag set, hoping to elicit a response. Since the FIN is intended to close a connection, the connection attempt may not even get logged. (12)

Other odd combinations of flags may be used to profile an operating system. Based on the responses to these packets, an attacker could determine what operating system is being used, and thus know what kinds of vulnerabilities to try to exploit. This is known as OS profiling or OS fingerprinting.

If a match occurs, a jump is made to a second user-defined chain named log-tcp-state, which logs the event with the prefix "Illegal TCP state" and then drops the packet. If a packet does not match any conditions, then filter processing returns to the next rule in the built-in chain.

```
#
# tcp-state-flags
#
# Check for Stealth Scans and illegal TCP State Flags
# All of the bits are cleared
iptables -A tcp-state-flags   -p tcp --tcp-flags ALL NONE -j log-tcp-state

# SYN and FIN are both set
iptables -A tcp-state-flags   -p tcp --tcp-flags SYN,FIN SYN,FIN -j log-tcp-state

# SYN and RST are both set
iptables -A tcp-state-flags   -p tcp --tcp-flags SYN,RST SYN,RST -j log-tcp-state

# FIN and RST are both set
iptables -A tcp-state-flags   -p tcp --tcp-flags FIN,RST FIN,RST -j log-tcp-state

# FIN is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags   -p tcp --tcp-flags ACK,FIN FIN -j log-tcp-state

# PSH is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags   -p tcp --tcp-flags ACK,PSH PSH -j log-tcp-state

# URG is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags   -p tcp --tcp-flags ACK,URG URG -j log-tcp-state

#--------------------------------------------------------------
#
# log-tcp-state
#
# Log TCP packets with bad state combinations
#
iptables -A log-tcp-state -p tcp -j LOG \
         --log-prefix "Illegal TCP state: " \
         --log-ip-options --log-tcp-options

iptables -A log-tcp-state -j DROP
```

The second user-defined chain called by all built-in chains is used to check for connection state. If the packet is found to be part of and existing connection, or as traffic related to an existing connection, it is accepted, and no further checks are done. Checking for invalid state, allows the detection of packets acting as if they were part of a legitimate established connection, or a response that wasn't solicited, such as an echo reply when no echo request was made.

```
#--------------------------------------------------------------
#
# conn-track
#
# Use Connection State to By-pass Rule Checking
#
iptables -A conn-track -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A conn-track -m state --state INVALID -j LOG \
         --log-prefix "INVALID packet: "
iptables -A conn-track -m state --state INVALID -j DROP
```

Primary user-defined chains based on traffic flow:

The next portion of rules defines the primary user-defined chains based on traffic flow. These are the core of the firewall policy and the jumping off point for all further checks on traffic. As explained above in section 2.3.3.2, these chains provide the logic to implement the firewall policy, specifying which packets will be examined further based on their protocol and service. If the packet is not checked against a secondary chain, then it is dropped by default and the connection will not be allowed. These primary chains are the basis by which the policies for each of the user groups are implemented.

Incoming traffic from the Internet to the service network:

The service network contains the critical services which allow GIAC Enterprises to communicate with is customers, suppliers and partners. All of the services in the service network are available to these three classes of users.

Since it is anticipated that this will constitute a large portion of the traffic, it is important to consider the order in which the service rules are placed. The more quickly a match is made, the higher the throughput.

The rule for checking queries to the web server is at the top, since the main interface to GIAC Enterprises is through a web interface. This rule checks for both http and https (see section 2.4) traffic. The second rule checks email connections to the service network. NTP queries are checked by the ntp-query chain to see if they come from the border router. Syslog messages also are checked by the syslog-message chain to see if they come from the border router. Finally, certain types of icmp messages are allowed inbound. These are the only inbound connections that are allowed from the Internet to service net systems. For each of these protocols, further checks are made in user-defined chains that relate specifically to the protocol being used. Specifically not allowed are Internet initiated ssh connections to any service net systems. Although a rule is placed matching this protocol, it is left up to the user-defined chain ssh-connect to log and drop the packet. The authentication protocol, usually associated with email exchanges is also checked further so that it can be rejected with a TCP reset.

```
#-------------------------------------------------------------
#
# ext-to-svc
#
# Packets inbound from Internet to the service network

# Allow inbound queries to the web server
iptables -A ext-to-svc -p tcp -m multiport \
  --destination-port 80,443 --syn -j www-query
# Allow inbound email delivery to the email server
iptables -A ext-to-svc -p tcp --dport 25 -j svc-mail
# Allow inbound time synchronization from border router to service ntp server
iptables -A ext-to-svc -p udp --dport 123 -j ntp-query
# Allow inbound syslog message to service net syslog server
iptables -A ext-to-svc -p udp --dport 514 -j syslog-message
# Allow certain types of icmp messages inbound
iptables -A ext-to-svc -p icmp -j icmp-in

# Reject auth queries
iptables -A ext-to-svc -p tcp --dport 113 -j auth-query
# Do not allow ssh connections, log and drop them via chain ssh-connect
```

```
iptables -A ext-to-svc -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
```

Outgoing traffic from the service network to the Internet:

The outgoing traffic on the service network provides a means by which GIAC employees can interact with customers, suppliers and partners by visiting their web sites and sending email to them. Other traffic relates to the simple mechanics of the network such as DNS and NTP.

Services that require connections to Internet servers that originate from the service network include email, name server, web proxy, network time, and authentication. Traffic related to these protocols is further checked to meet stricter conditions. The web proxy traffic is placed first due to the expected traffic volume. Some additional comments about the web proxy rule are in order. This rule makes use of the multiport matching feature. Rather than having a rule to match each port number, using the multiport facility, you can simplify the filtering to just one rule listing the multiple ports.

TIP: This rule is rather sensitive to syntax constraints. Note that the --sport appears after the destination ports. Multiport uses its own option for destination port as seen by the --destination-port rather than the --dport. If you place the --sport 1024:65535 ($UNPRIV) before the –m multiport, it causes a syntax error. Even if placed directly after the --destination-port, it still causes a syntax error. However, by placing the --syn between them, it appears to denote an end to the multiport options and allows the use of the normal matching options.

The smtp (email), DNS, NTP, ICMP and auth protocols are also sent to protocol specific chains for further testing. Ssh connections are not permitted, and are logged and dropped.

```
#-----------------------------------------------------------------
#
# svc-to-ext
#
# Packets outbound from service network to the Internet

# Allow web proxy queries of "safe" well-known web services
iptables -A svc-to-ext -p tcp -m multiport \
  --destination-port 21,80,443,563,591,70,210 \
  --syn --sport $UNPRIV -j squid-query
# Allow email delivery to Internet email servers
iptables -A svc-to-ext -p tcp --dport 25 -j svc-mail
# Allow dns server queries to the Internet
iptables -A svc-to-ext -p udp --dport 53 -j dns-query
iptables -A svc-to-ext -p tcp --dport 53 -j dns-query
# Allow ntp client queries
iptables -A svc-to-ext -p udp --dport 123 -j ntp-query
# Allow certain icmp messages out, as needed for normal service operations
iptables -A svc-to-ext -p icmp -j icmp-out
# Allow auth queries in special cases
iptables -A svc-to-ext -p tcp --dport 113 -j auth-query

# Do not allow ssh connection out, drop and log
iptables -A svc-to-ext -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
```

Inbound traffic from the service network to the internal network:

The only critical services that require the service network to originate connections to the internal network are email and database queries. The email service relays email from the service network email server to the internal network email server. The service network web server, based on user input via the web interface, initiates the database queries. The database queries are completed via the database's own protocol using limited read-only privileges. The icmp-in chain allows some ICMP messages in. Authentication is rejected and requires special handling by auth-query chain. Ssh connections are matched, but are not allowed and are logged and dropped by the ssh-connect chain.

```
#---------------------------------------------------------------
#
# svc-to-int
#
# Packets inbound from service network to internal network

# Allow email delivery to internal email server
iptables -A svc-to-int -p tcp --dport 25 -j svc-mail
# Allow database queries to internal database server
iptables -A svc-to-int -p tcp --dport $DBPORT -j db-query
# Allow certain types of icmp
iptables -A svc-to-int -p icmp -j icmp-in

# Reject auth queries
iptables -A svc-to-int -p tcp --dport 113 -j auth-query
# Do not allow ssh connections, log and drop them via chain ssh-connect
iptables -A svc-to-int -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
```

Outgoing traffic from the internal network to the service network:

In some ways, the service network acts as a proxy to the internal network. Network services such as email, web and ntp are passed to servers on the service network, which in turn query the appropriate Internet server and return the results to the internal network. The firewall policy can thus dictate tight controls over traffic between the internal network and the Internet.

The most significant volume of traffic, requests to the web proxy server, is handled by the first rule, allowing connections from internal network systems to the service net web proxy server. Next, the firewall policy allows ssh connections from the internal network to the service net systems via the ssh-connect chain. Email is further checked by the svc-mail chain, which allows the internal email server to forward email only to the service net email server. DNS and NTP queries are also checked to allow connections between the corresponding internal and service net servers only. Some ICMP messages types are allowed out to the service network and auth packets will be rejected.

```
#---------------------------------------------------------------
#
# int-to-svc
#
# Packets outbound from internal network to service network

# Allow client requests to the web proxy server
iptables -A int-to-svc -p tcp --sport $UNPRIV --dport 8008 -j squid-query
# Allow ssh connections
```

```
iptables -A int-to-svc -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow email forwarding from internal email server to service net email server
iptables -A int-to-svc -p tcp --dport 25 -j svc-mail
# Allow DNS recursive queries from internal dns server to service net dns server
iptables -A int-to-svc -p udp --dport 53 -j dns-query
iptables -A int-to-svc -p tcp --dport 53 -j dns-query
# Allow NTP client queries to the service net NTP server
iptables -A int-to-svc -p udp --dport 123 -j ntp-query
# Allow certain icmp messages out, as needed for normal service operations
iptables -A int-to-svc -p icmp -j icmp-out


# Reject auth queries
iptables -A int-to-svc -p tcp --dport 113 -j auth-query
```

Inbound traffic from the Internet to the internal network:

This traffic flow is the most dangerous traffic, as it deals with connections originating from the Internet to your protected internal network. The firewall policy dictates that all traffic be limited to secure connections. As a result, the only connection traffic allowed through is via the ssh protocol and is checked via the ssh-connect chain. Some ICMP messages are permitted in for the proper functioning of ssh. The remaining traffic is logged, and dropped or rejected.

The reason ssh is allowed in is to enable remote logins by employees of GIAC Enterprises and partners, or the ability to upload download products by customers, suppliers and partners. Since these products are of value, they lie within the internal network and access is subject to strict authorization and monitoring as specified by the perimeter design.

Note that in this case, no auth protocol is even checked for, causing it to be dropped by default. Since ssh, the only service run between the internal net and the Internet, does not use auth, there is no reason to even check this traffic, reducing the number of rules needed.

```
#---------------------------------------------------------------
#
# ext-to-int
#
# Packets inbound from Internet to the internal network

# Allow incoming ssh connections
iptables -A ext-to-int -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow certain types of icmp
iptables -A ext-to-int -p icmp -j icmp-in
```

Outbound traffic from the internal network to the Internet:

The only connections that can be initiated by the internal network directly to the Internet are ssh connections. This allows employees of GIAC Enterprises to connect to supplier and partner sites to download or upload products. Again, certain ICMP messages are checked by ICMP-out, to insure proper functioning of ssh. One service that is specifically not allowed is connections to Internet web server. This traffic is logged and dropped. This is useful in debugging internal systems that use automatic update services such as anti-virus software obtaining updated virus signature files. This is a service you want to make sure is working properly and such log entries will put you on notice that this software's configuration needs to be corrected to use the web proxy server to obtain its updates.

```
#---------------------------------------------------------------
#
# int-to-ext
#
# Packets only outbound from internal network to the Internet

# Allow outbound ssh
iptables -A int-to-ext -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow certain icmp messages out, as needed for normal service operations
iptables -A int-to-ext -p icmp -j icmp-out

# Do not allow outbound queries to the Internet web servers
iptables -A int-to-ext -p tcp --dport 80 -j www-query
```

Firewall Server interface traffic:

The remaining traffic is limited to traffic that is destined for or originating from the three interfaces of the firewall server. There is no need for any connections on these interfaces for services used by customers, suppliers or partners. The only connections required would be related to system administration via ssh or network mechanics, namely DNS and NTP.

Six user-defined chains are defined as EXT-input, EXT-output, SVC-input, SVC-output, INT-input and INT-output. Only two chains explicitly allow traffic.

The SVC-output chain allows the firewall server to conduct DNS and NTP client queries while logging and dropping ssh traffic from the firewall to the service network.

```
#---------------------------------------------------------------
#
# SVC-output
#
# Packets outbound on the service network interface intended for the service net

# Allow DNS queries to service net DNS server
iptables -A SVC-output -p udp --dport 53 -j dns-query
iptables -A SVC-output -p tcp --dport 53 -j dns-query
# Allow ntp client queries
iptables -A SVC-output -p udp --dport 123 -j ntp-query

# Do not allow outbound ssh, log and drop
iptables -A SVC-output -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
```

The INT-input chain is the only chain that allows an ssh connection to the firewall server, and is limited to specific internal systems (system administrator consoles).

```
#---------------------------------------------------------------
#
# INT-input
#
```

```
# Packets inbound on the internal interface intended for this server

# Allow ssh connections for remote admin
iptables -A INT-input -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow certain types of icmp
iptables -A INT-input -p icmp -j icmp-in
```

The final chain listed checks outbound ICMP messages and will log and drop ssh connections.

```
#------------------------------------------------------------
#
# INT-output
#
# Packets outbound on the internal interface intended for the internal net
# Allow certain icmp messages out, as needed for normal service operations
iptables -A INT-output -p icmp -j icmp-out

# Do not allow outbound ssh to internal net, log and drop
iptables -A INT-output -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
```

The remainder of the interface chains simply log and block ssh traffic to or from their interface (see Appendix B).

Secondary User-defined Chains:

The remainder of the user-defined chains creates the specific rules that implement the firewall policy for each allowed service. These chains may also log any dropped traffic utilizing its own unique log prefix to assist in debugging and firewall log analysis. The remainder of this section will deal with these individual service chains. The order in which these chains are listed is not significant in the filtering process. It is the primary user-defined chains that determine the order the services are checked (see above). The rule order within the following chains only effects the protocol the chain is dealing with.

Email - SMTP

The email service uses the smtp protocol on TCP port 25. In this user-defined chain, the source port is used to match the traffic, constricting the flow to only individual email servers. First, mail is allowed to be sent, or received from Internet mail servers as long as it is directed to/from the service net mail server. Note in the second rule, the syntac "-s ! $INTNET". This matches all traffic NOT from the internal network. This inverse logic is used to exclude internal traffic from the match and block it by default, rather than having to add a second rule that would specifically block this traffic. The fewer the rules, the faster the throughput.

The next rules allow for email exchanges between just the internal and service net email servers. No other email traffic is allowed, and is logged and dropped.

These rules implement the firewall policy that is summarized as follows. Only the service net email server can communicate directly with other Internet mail servers. The only mail server that the internal mail server can communicate with is the service network email server. Internal network users can only communicate with the internal mail server.

```
#--------------------------------------------------------------
# svc-mail
#
# Mail server (destination tcp port 25)

# Allow outgoing mail to the Internet from service net mail server
iptables -A svc-mail -p tcp  -s $SVCMAIL --sport $UNPRIV \
      -d ! $INTNET -m state --state NEW -j ACCEPT

# Allow incoming mail from the Internet to service net mail server
iptables -A svc-mail -p tcp -s ! $INTNET --sport $UNPRIV -d $SVCMAIL \
      -m state --state NEW -j ACCEPT

# Allow outgoing mail to the internal mail server from service net mail server
iptables -A svc-mail -p tcp  -s $SVCMAIL --sport $UNPRIV \
      -d $INTMAIL -m state --state NEW -j ACCEPT

# Allow incoming mail from the internal mail server to service net mail server
iptables -A svc-mail -p tcp -s $INTMAIL --sport $UNPRIV -d $SVCMAIL \
      -m state --state NEW -j ACCEPT

# Deny any other email connections
iptables -A svc-mail -j LOG --log-prefix "Illegal Mail connect: "
iptables -A svc-mail -j DROP
```

Web – HTTP, HTTPS

The web requests on TCP port 80 and 443 are accepted from anywhere as long they are directed to the service net web server. Otherwise, they are logged and dropped. It should be noted that scanning for web servers is a common occurrence, and may not provide much useful information. However, it may be helpful at first to determine the level of active scanning of the network and at some later point dropped silently by commenting out the logging rule.

```
#--------------------------------------------------------------
#
# www-query
#
# Web server (tcp port 80, 443)

iptables -A www-query -d $WWWSERVER \
         -m state --state NEW -j ACCEPT

# Deny any other squid connections
iptables -A www-query -j LOG --log-prefix "Illegal WWW query: "
iptables -A www-query -j DROP
```

Web Proxy – FTP,HTTP,AUTH,HTTPS,NNTPS,GOPHER,WAIS

The squid-query chain deals with the requests made to the service net web proxy server. This server runs the Squid software and can handle requests for ftp, http, auth, https, nntps, gopher and wais. These are listed as the "safe ports" in the squid configuration file and were used to match traffic to this chain. The squid configuration file also enables limiting requests to specific networks. This provides additional protection backing up the following rules, which allow connections from the internal network to the service net web proxy server. The second rule allows the web proxy server to establish connections to anywhere on the Internet, as long as it is not the internal network. (Another use of reverse logic.) This also illustrates the importance of connection state. If an internal network connection was established by the first rule, but did not maintain state, the return traffic from the squid server would not match either rule and be blocked by the fianl rule.

```
#--------------------------------------------------------------
#
# squid-query
#
# Web cache proxy server (tcp port 8008)

iptables -A squid-query -s $INTNET -d $SQUIDSERVER \
         -m state --state NEW -j ACCEPT

iptables -A squid-query -s $SQUIDSERVER –d ! $INTNET\
         -m state --state NEW -j ACCEPT

# Deny any other squid connections
iptables -A squid-query -j LOG --log-prefix "Illegal Squid query: "
iptables -A squid-query -j DROP
```

Domain Name Service - DNS

DNS is a required service for any network connected to the Internet. This service is best deployed with a firewall using two servers, one in the service network and one in the internal network. (13) This is often referred to as "split-DNS."

The internal DNS server is a forward-only server. It answers queries from its authoritative data as well as cached data. (14) However, it acts as an authoritative server for just the internal network and as a cache and forward DNS server for requests from internal systems only. The only other DNS server it communicates with is another cache and forward server located in the service network.

The service net DNS server is also a forward only DNS server, answering requests from the internal DNS server, other service net systems and the firewall server itself. In this way, the service net DNS server acts as a proxy for internal DNS queries.

It is possible for the service network DNS server to be an authoritative DNS server, either as a primary or secondary to the ISP's DNS server. Since there are only two public services, web and mail, only two systems need DNS entries, which could more easily be maintained on the ISP's DNS server. This would reduce the need for DNS zone transfers to the service network DNS server, simplifying installation and maintenance of the DNS servers.

For DNS to work properly, it needs to be able to communicate using both UDP and TCP. For client to server queries, the client uses unprivileged ports to query UPD port 53 on the server. If the returned information is too large for a UDP packet, the client retries using a TCP connection to port 53. In server to server queries, the query is made from UDP port 53 to UDP port 53. This has changed in newer version of bind, which uses unprivileged source ports, but can be configured to use just port 53. However, if the return message is too large for a UPD packet, the server must resort back to a client to server request using TCP. (14)

The firewall policy for DNS is summarized as follows. The only DNS server allowed to make queries to Internet DNS servers is the service net DNS server. Only the service net DNS server can answer queries from the internal DNS server. Internal users are only allowed to query the internal DNS server. Additionally, the Firewall server is allowed to query the service net DNS server, as are the other service net systems. The border router disables DNS, so client queries from outside the firewall need not be allowed.

```
#--------------------------------------------------------------
#
# dns-query
#
# dns (UDP/TCP Port 53)

# Allow service net DNS server queries to an external DNS server via UDP 53
iptables -A dns-query -p udp -s $SVCDNSSERVER --sport 53 -d ! $INTNET \
       -m state --state NEW -j ACCEPT

# Allow service DNS client queries to external DNS server via UDP/TCP UNPRIV
iptables -A dns-query -p udp -s $SVCDNSSERVER --sport $UNPRIV -d ! $INTNET \
       -m state --state NEW -j ACCEPT
iptables -A dns-query -p tcp -s $SVCDNSSERVER --sport $UNPRIV -d ! $INTNET \
       -m state --state NEW -j ACCEPT

# Allow internal DNS server queries to service net DNS server via UDP 53
iptables -A dns-query -p udp -s $INTDNSSERVER --sport 53 -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT

# Allow internal DNS client queries to service DNS server via UDP/TCP UNPRIV
iptables -A dns-query -p udp -s $INTDNSSERVER --sport $UNPRIV -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT
iptables -A dns-query -p tcp -s $INTDNSSERVER --sport $UNPRIV -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT

# Allow firewall service interface queries to service net server via tcp or udp
iptables -A dns-query -s $SVCADDR -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT

# Deny any other DNS connections
iptables -A dns-query -j LOG --log-prefix "Illegal DNS query: "
iptables -A dns-query -j DROP
```

Secure Shell – SSH, SCP

The ssh protocol plays a key role in the VPN policy of the perimeter design. It is used for remote logins, VPN tunneling of the imap and smtp protocols for remote email access, and is the primary means by which products can be transferred between GIAC Enterprises and their customers, suppliers and partners.

Even though ssh and scp (secure copy) provides a safer means for communication through use of encrypted communications and certificate exchange, it still must be tightly controlled. The firewall policy states that suppliers be allowed to download files using scp, while partners have the ability to log into the internal ssh server to exchange information or further interact on a cooperative basis with GIAC Enterprises employees. The rules below provide for these kinds of access.

The first rule allows connections from the internal network to the service network. This allows GIAC Enterprises employees to maintain and modify the service network systems and web server files. The reason it is placed first is that ssh is used to copy across the system log files, which can be rather large. The next rule allows ssh connections to the Internet initiated from the internal network so that employees can access supplier and partner systems to upload/download products or otherwise access their systems in a secure manner. A rule then allows connections from the outside to an internal ssh server. This server allows remote logins by employees and partners, and the establishment of VPN tunnels. The last rule allows an ssh connection to the firewall server on its internal interface from a system administration console. All other connections are logged and dropped.

```
#--------------------------------------------------------------
#
# ssh_connect
#
# ssh (TCP Port 22)

# Allow ssh connections to service network only from internal network
iptables -A ssh-connect -s $INTNET -d $SVCNET -m state --state NEW -j ACCEPT

# Allow ssh connections initialed by internal network to any on the Internet
iptables -A ssh-connect -s $INTNET -d ! $SVCNET -m state --state NEW -j ACCEPT

# Allow ssh connections to specific internal ssh server from Internet
iptables -A ssh-connect -s ! $SVCNET -d $SSHSERVER -m state --state NEW -j ACCEPT

# Allow ssh connections to external interface of firewall for remote management
iptables -A ssh-connect -s $SYSADM -d $INTADDR -m state --state NEW -j ACCEPT

# Deny any other ssh connections
iptables -A ssh-connect -j LOG --log-prefix "Illegal ssh connect: "
iptables -A ssh-connect -j DROP
```

Network Time Protocol – NTP

NTP is used to synchronize log files across all GIAC Enterprises networks. This is a vital service that assists in intrusion detection. Logged events from multiple systems throughout the networks can be compared using the same time frame to search for a pattern of attack.

The service network ntp server acts as a “proxy” for the internal network. The service network ntp server is allowed to query Internet ntp servers. The internal network NTP server can only query the service net NTP server. All internal network systems can only query the internal NTP server. Additionally, service network systems and the border router are only allowed to query the service network NTP server.

In the rules below, internal NTP server requests to the service net server are allowed. The service net NTP server is allowed to contact Internet NTP servers (not internal net). All other NTP connection attempts are logged and dropped.

The NTP server configuration file /etc/ntp.conf has methods of limiting and controlling the actions of the server. For example, you can permit time synchronization with an NTP server based on the network address, but not allow them to query or modify the server. Again, this is defense in depth, and should be followed even for a simple service like NTP. You might ask yourself, what would happen if your systems clock were suddenly set to a time in the past? Would this effect database operations or entries? How would the operating system respond? It might just be a nuisance, but it also might prove to be a denial of service.

```
#--------------------------------------------------------------
#
# ntp_query
#
# ntp (UDP Port 123)

# Allow ntp synchronization from internal ntp server to service net ntp server
iptables -A ntp-query -s $INTNTPSERVER -d $SVCNTPSERVER -m state --state NEW -j
ACCEPT

# Allow ntp synchronization from border router to service net ntp server
iptables -A ntp-query -s $BDRADDR -d $SVCNTPSERVER -m state --state NEW -j ACCEPT

# Allow ntp synchronization from service net ntp server to external ntp server
iptables -A ntp-query -s $SVCNTPSERVER –d ! $INTNET -m state --state NEW -j ACCEPT

# Deny any other ntp connections
iptables -A ntp-query -j LOG --log-prefix "Illegal ntp query: "
iptables -A ntp-query -j DROP
```

Authentication – identd

This protocol is used to determine the "real name" of the user contacting your system and is most often seen in services such as email or ftp. When a server wants to confirm who is connecting to it, it sends a auth request back to the source of the connection. The source returns a text message containing the "real name" extracted from the password file. This assumes that the reply is telling the truth, or that the user account has not been hacked. (15)

As such, it is not a very useful service, but sometimes it is necessary to respond to it, even if it is a rejection rather than a reply. Sendmail is an email delivery service that uses auth. It will send an auth request when a message is sent to it. Sendmail will send a request, then wait for a reply multiple times before it gives up and accepts the email anyway. The length of this timeout depends on the system OS or Sendmail configuration, and may significantly slow down delivery. If the packet is rejected, a TCP packet with the RESET flag set is sent back in reply, closing the connecting attempt and essentially telling the other system that auth is not available. Sendmail then will immediately accept the email that was sent to it.

The only auth traffic allowed is from the service net email server to Internet servers. Incoming requests are rejected with a TCP reset packet. All other traffic is dropped and logged.

```
#--------------------------------------------------------------
#
# auth-query
```

```
#
# auth/identd (TCP Port 113)

# Allow auth queries from service network email server to the Internet
iptables -A auth-query -p tcp --syn -s $SVCMAIL -d ! $INTNET \
       -m state --state NEW -j ACCEPT

# Reject, with response, any auth queries
iptables -A auth-query -p tcp --syn \
       -j REJECT --reject-with tcp-reset

# Deny any other squid connections
iptables -A auth-query -j LOG --log-prefix "Illegal auth query: "
iptables -A auth-query -j DROP
```

Syslog – syslogd

This chain checks for syslog messages coming from the border router to the service network syslog server. This is the only logging passed between networks because the border router has limited memory and is unable to extensively log for any length of time. The service net systems' logs are copied to the internal system administrator system for further analysis via an automated script using scp.

```
#-------------------------------------------------------------
#
# syslog-message
#
# database listener port (UDP Port 514)
iptables -A syslog-message -s $BDRADDR -d $SVCSYSLOG -j ACCEPT

# Log and Deny any other database connection connections
iptables -A syslog-message -j LOG --log-prefix "Illegal syslog message: "
iptables -A syslog-message -j DROP
```

Database queries

The database used by GIAC Enterprises has a "listener" service on an internal database proxy service, which in turn contacts the internal protected network database server. This allows systems to contact the database server and make queries. The firewall policy permits only the service network web server to contact the database proxy server.

Two additional security systems provide protection for the database server. One is that the database server grants the web server readonly access. Secondly, the database server is located behind another firewall on the internal network which also has rules which limit access from systems outside the internal network to just the internal database proxy server.

```
#-------------------------------------------------------------
#
# db-query
#
# database listener port (TCP Port 1523)
iptables -A db-query -s $WWWSERVER -d $DBSERVER -m state --state NEW -j ACCEPT
```

```
# Log and Deny any other database connection connections
iptables -A db-query -j LOG --log-prefix "Illegal db query: "
iptables -A db-query -j DROP
```

ICMP outgoing traffic:

There are only four ICMP message types that are actually needed for normal network operations. These are source quench, parameter problem, incoming destination unreachable and outgoing destination unreachable with a subtype of fragmentation needed. Four others are optional, echo request, echo reply, time exceeded, and the remaining outgoing destination unreachable subtypes. All other icmp messages types can be dropped. (8 – p. 171)

The firewall policy allows the necessary four ICMP messages. Before allowing any legitimate traffic out, packets are checked for fragmented icmp packets. This should not happen for normal traffic. This rule is placed before the allowed ICMP traffic, to prevent fragmentation of necessary messages types. Outgoing source quench is allowed in order for the system to slow down the rate of transmission for established connections. Parameter problem messages are allowed out when there are illegal or unexpected values in the protocol header or checksum errors.

Fragmentation needed messages are permitted and are necessary for normal network operations. A TCP message is sent with the DF (do not fragment) flag set in an attempt to discover the smallest MTU (maximum transmission unit) in the path to the destination. If a network receives such a packet with a smaller MTU than the packet size, it sends an ICMP fragmentation needed message. The source system creates all further TCP packets in that connection with a size that meets this requirement. (12 – p. 59)

One optional icmp message type is allowed because of iptables' connection tracking ability and limits on the source. Outgoing echo requests (ping) are allowed if they originate from the internal network system administration console. Ping is still a useful tool in diagnosing network problems. All other icmp traffic is blocked. Note that this includes dropping outgoing destination unreachable messages.

```
#--------------------------------------------------------------
#
# icmp-out
#
# ICMP traffic


# Drop outgoing ICMP fragments
iptables -A icmp-out --fragment -j LOG --log-prefix "Frag outgoing ICMP: "
iptables -A icmp-out --fragment -j DROP

# Allow source quench
iptables -A icmp-out -p icmp --icmp-type source-quench -j ACCEPT

# Allow parameter-problem
iptables -A icmp-out -p icmp --icmp-type parameter-problem -j ACCEPT

# Allow fragmentation needed
iptables -A icmp-out -p icmp --icmp-type fragmentation-needed -j ACCEPT
```

```
# Allow outgoing ping from internal network
iptables -A icmp-out -p icmp --icmp-type echo-request \
      -s $SYSADM -m state --state NEW -j ACCEPT

#Drop all other incoming ICMP traffic
iptables -A icmp-out -p icmp -j LOG --log-prefix "Illegal outgoing ICMP"
iptables -A icmp-out -p icmp -j DROP
```

ICMP incoming messages:

Incoming ICMP traffic is also very limited and at first blocks ICMP fragmented packets. The firewall policy allows incoming destination unreachable messages that indicate a requested service port or host on the Internet is unavailable. This speeds up applications like web browsers, as they avoid waiting for lengthy timeouts on connection attempts. Also allowed in are parameter-problem and source quench for similar reasons mentioned above.

This does allow one potential Denial of Service attack referred to as a "Super source quench." The ICMP source quench message has an associated redirect control message that tells the host to redirect traffic to a different router. The super source quench attack tells the host to redirect all traffic to its own loopback address, which can overload the system. (16)

```
#-------------------------------------------------------------
#
# icmp-in
#
# ICMP traffic

# Drop ICMP fragments
iptables -A icmp-in --fragment -j LOG --log-prefix "Frag ICMP: "
iptables -A icmp-in --fragment -j DROP

# Allow destination unreachable
iptables -A icmp-in -p icmp --icmp-type destination-unreachable -j ACCEPT

# Allow parameter-problem
iptables -A icmp-in -p icmp --icmp-type parameter-problem -j ACCEPT

# Allow source quench
iptables -A icmp-in -p icmp --icmp-type source-quench -j ACCEPT

#Drop all other incoming ICMP traffic
iptables -A icmp-in -p icmp -j LOG --log-prefix "Illegal incoming ICMP"
iptables -A icmp-in -p icmp -j DROP
```

This concludes the description of the user-defined chains. In summary, the primary chains are called by the built-in rules, while the secondary chains are called by the primary chains. If no match is made against a packet, it reaches the final rules on the built-in chains to log and drop the packet.

2.4 VPN Policy

The VPN capability of the GIAC Enterprises network is at this time fairly limited to using ssh and the secure sockets layer (ssl). Primarily, ssh is used for two purposes. It is used to permit remote access to email for GIAC Enterprises employees by tunneling the imapd and smtp protocols through using the ssh protocol. It is also used to allow partners to have limited access to the internal network for collaborative projects and exchange of information. The ssl protocol provides the means for e-commerce transactions and retrieval of products by customers via the e-commerce web server. Finally, employees are allowed to tunnel the database listener protocol to the internal database proxy server to allow for remote updating of the database.

Remote Email Access:

Employees need remote access to GIAC Enterprises email system. It not only provides a consistent means of communications, but also enforces better security by virus checking all email sent or received from a remote system.

When an employee is attempting remote access, they are required to first connect to the internal ssh server and authenticate with their username/password. The ssh client software on their remote system must be configured. This is done as follows:

```
Start the SSH program, the select from the menu
  Edit->Settings
Click on the Host Settings->Tunneling->Outgoing entry
Click on "Add" to add the IMAP entry for incoming email
In the dialog box enter:
Name:         IMAP
Listen Port:  143
Uncheck the "Allow Local Connections Only"
Destination Host: 192.168.3.30
Destination Port: 143
Click OK

Click on "Add" to add an entry to allow for outgoing email
In the dialog box enter:
Name:         SMTP
Listen Port:  25
Uncheck the "Allow Local Connections Only"
Destination Host: 192.168.3.30
Destination Port: 25
Click OK

Click OK
Then exit and click on "save" the new settings when prompted.
```

Similar actions are taken to create tunnels for the https server on port 443 and the database proxy server on port 1523, named HTTPS and MYSQL respectively.

After ssh has been initialized, the setting for the web browser must be altered to use the localhost as its setting for both the incoming mail server using the imap protocol, and as the outgoing email server.

There are vulnerabilities associated with this technique. Any remote system that is scanned or attacked while it is connect will subject the internal email server to the same scan or attack since the local port is actually a direct tunnel to the internal server. For this reason, all remote systems that access the email server must run Zone Alarm, a personal firewall, to prevent external access to these four ports. All remote systems must also have current anti-virus software and up-to-date virus definitions when they access the system.

Database access:

As mentioned above, write access to the database is enabled by creating tunnels to the https (internal e-commerce server), and to the internal database proxy listener port. Partners are limited to only access via the https port, while employees are allowed access to both. Since the firewall policy cannot distinguish between these users, the database proxy server must implement access controls.

Secure socket layer:

The service network web server is configured to use ssl on tcp port 443 for secure connections when utilizing portions of the web site that deal with e-commerce and the delivery of customer products. This web server is actually a second instance of the Apache web server running on the public http web server. Customers, suppliers and partners must not only authenticate with a username and password, but the server will also request a client certificate when establishing the connection to verify the user is actually connecting from a registered system.

In the httpd.conf file, the following entries must be made for the area of text and scripts to be accessed only with an SSL connection. This will require a client certificate and the use of strong encryption ciphers. This certificate must first be created for the customer, supplier or partner, and signed with GIAC Enterprises' own certificate that was obtained from a valid Certificate Authority (CA). (3)

```
<Directory /usr/local/apache/htdocs/secure/ecommerce>
          SSLVerifyClient      require
          SSLVerifyDepth       5
          SSLCACertificateFile conf/ssl.crt/ca.crt
          SSLCACertificatePath conf/ssl.crt
          SSLOptions           +FakeBasicAuth +StrictRequire
          SSLRequireSSL
          AuthName             "GIAC Enterprises Authentication"
          AuthType             Basic
          AuthUserFile         /usr/local/apache/conf/httpd.passwd
          require              valid-user
</Directory>
```

The user is also required to be a valid user, and must be included in a user database /usr/local/apache/conf/httpd.passwd. Examples of entries in the file are:

```
/C=CE/L=Geneva/O=Fortunes Swiss/OU=Staff/CN=Hans:xxj31ZMTZzkVA
/C=US/L=N.Y/O=Make Your Fortunes Today/OU=CA/CN=Matty:xxj31ZMTZzkVA
/C=US/L=O.R./O=Fortunes Forever/OU=Dev/CN=Charles:xxj31ZMTZzkVA
```

The letter C indicates country, L the locations, O the company (organization) name, OU the department (organizational unit), and CN the common name, followed by the hash of the password.

Requiring strong encryption ciphers by the web browsers, a user certificate, and a username/password authentication can thus make secure access for these sensitive files. Separate directories with tighter access control can be used to separate customers, suppliers and partners. Allowing them to retrieve products and access critical information about their accounts via scripts using sql queries to the internal protected database via the internal database proxy server.

While this is a limited VPN capability, it has the advantage of being extremely cheap. As business needs increase usage, system latency may become a problem and the use of a hardware VPN device such as the Cisco 3005 may be warranted.

2.5 Firewall Tutorial

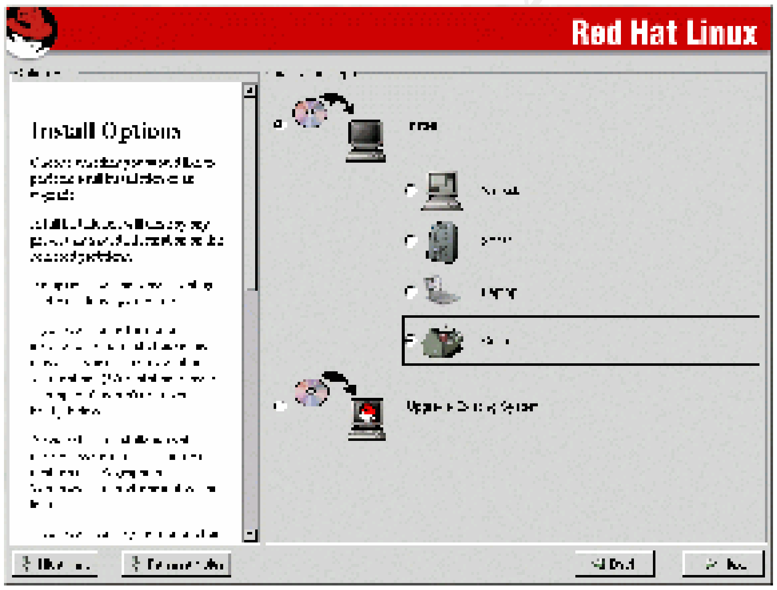
2.5.1 Installing and Hardening RedHat Linux v7.3

The RedHat installation GUI, allows for extensive customization of the software installed on the firewall server. When hardening a server, it is a best practice to remove, or not install, as much unneeded software as possible. Although there are a few packages that are installed with the core operating system, sendmail being an example, it is possible to be extremely selective on what is installed on the server.

When you boot up with the installation CD provided with Red Hat Linux, you are presented with a series of screens that lead you through the installation process. The first four screens deal with language and hardware configuration. Select as appropriate for the region and basic system hardware (mouse, keyboard, etc…)

The next screen entitled "Install Options" allows you to select the basic sets of software you want to install (see Figure 7)

Figure 7

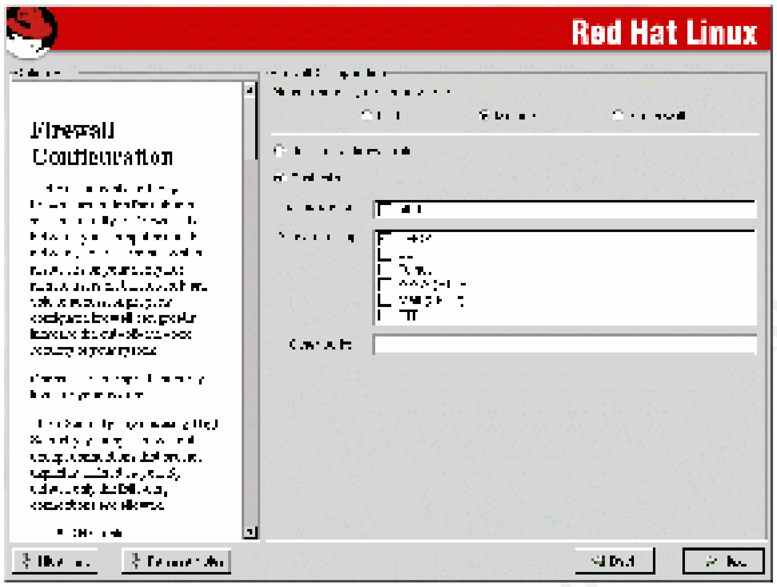


As shown in the figure, click on the custom installation type. This will enable selecting software packages later in the installation process.

The next tasks are to partition the hard drive and configure the boot loader, selecting the lilo boot loader. The network interface configurations require the IP addresses and subnet masks for each interface, as defined by the network design.

Figure 8 displays the Firewall Configuration screen.

Figure 8



Red Hat Linux

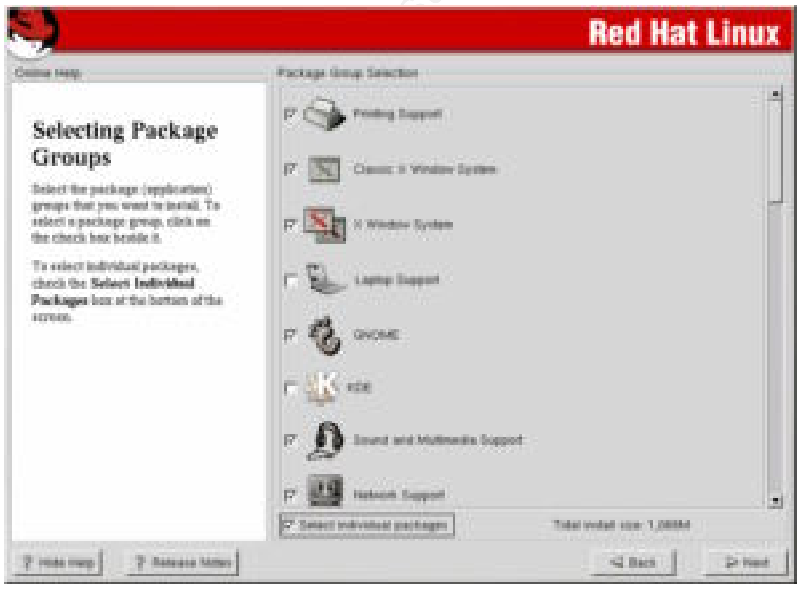
Firewall Configuration

[illegible]

This creates an initial firewall ruleset used when the system boots after the installation. Although the ruleset will be rewritten (the subject of this tutorial), it is a good practice to select "High," click on "Customize" and check "SSH." This creates a default firewall policy of "deny everything unless specifically allowed." Only DNS replies, and SSH will be allowed. This will keep the system safe while you finish hardening the server and create your site-specific firewall ruleset. This default ruleset will be located in /etc/sysconfig/ipchains and uses the ipchains firewall software.

Additional language, time zone, root and user accounts, and authentication configurations follow before reaching the "Selecting Package Groups" screen is displayed (see Figure 9)

Figure 9



Red Hat Linux

Online Help

Selecting Package Groups

Select the package (application) groups that you want to install. To select a package group, click on the check box beside it.

To select individual packages, check the **Select Individual Packages** box at the bottom of the screen.

Package Group Selection

- [✓] Printing Support
- [✓] Classic X Window System
- [✓] X Window System
- [ ] Laptop Support
- [✓] GNOME
- [ ] KDE
- [✓] Sound and Multimedia Support
- [✓] Network Support

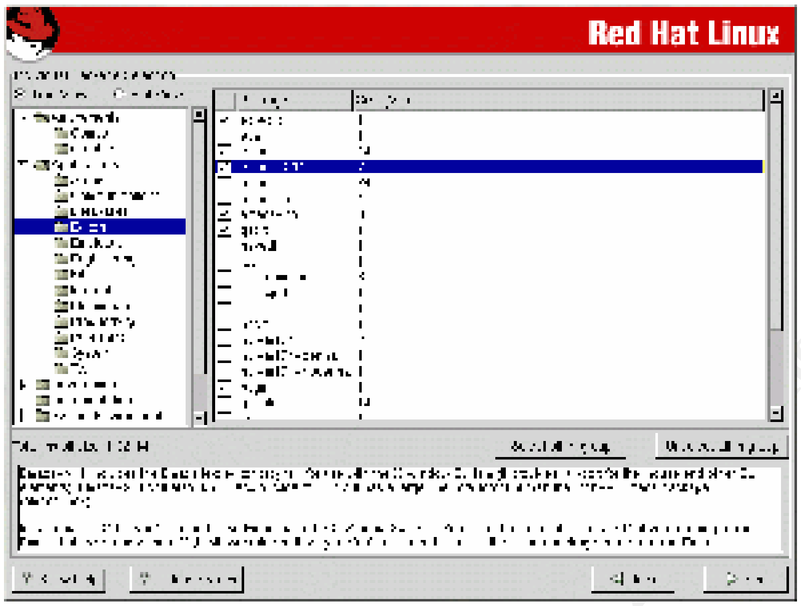
[✓] Select individual packages

Total install size: 1,080M

This screen allows for the selection of sets of software packages related to specific tasks, i.e. printing or the preferred GUI. Uncheck every box in the list so that they will not be installed by default. Check the box labeled "Select Individual Packages" to enable the ability to individually select the specific software that you do want to install.

The next screen (see Figure 10) displays the package groupings and lists individual software packages you can install. At this point, none of the packages should be checked.

Figure 10



The following is a list of packages that should be selected for installation:

Package Group	Software Package
Amusements	None
Applications -> Archive	unzip
Applications -> Internet	openssh, openssh-clients, tcpdump, traceroute
Applications -> System	procinfo, sudo, tripwire
Development -> Debug	lsof
Development -> Languages	cpp, gcc, perl
Development -> Libraries	glibc-devel, libpcap, ncurses-devel
Development -> System	glibc-kernheaders, kernel-source
Development -> Tools	binutils, make, patch
Documentation	None
System Environment -> Daemons	ntp, openssh-server, tcpwrappers
System Environment -> Lib	freetype, gd, libcap, libjpeg, libpng, ncurses4
System Environment -> Shells	pdksh
User Interface	None

The final tasks are to review the installation configuration and select to create an emergency boot disk. The software will install from three CDs and will reboot once the installation is complete.

Further hardening can be done to the server once it reboots. The SANS' Securing Linux Step-By-Step (www.sans.org) guide is an excellent resource to assist in securing a linux server. Since the smtp server software, sendmail, is loaded and started by default, you will have to stop it using the command:

```
/etc/rc.d/init.d/sendmail stop
```

To prevent it from restarting at bootup, disable the startup script by either renaming or removing the startup scripts from the /etc/rc.d/rc[0-5].d directories.

```
mv /etc/rc.d/rc2.d/S80sendmail /etc/rc.d/rc2.d/.NOS80sendmail
```

At the end of the hardening process, you should make sure you are not running any other services by checking the current listening ports using the lsof –i command. You should only see the sshd daemon listening on TCP port 22, as seen below. The second line is the remote established remote connection.

```
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd       649 root    3u  IPv4   1219       TCP *:ssh (LISTEN)
sshd      1688 root    4u  IPv4   2258       TCP 192.168.2.2:ssh->192.168.3.20:1028
(ESTABLISHED)
```

2.5.2 Modifying the Linux Kernel

As mentioned above, the Linux kernel can be recompiled to include the firewall software, iptables, as part of the kernel. In addition, it is possible to create a monolithic kernel, meaning all necessary kernel modules are compiled into the kernel and no other modules are allowed to be loaded after bootup. This helps prevent kernel level root kits from being installed, further protecting the system.

There are many references available that cover rebuilding the Linux kernel. The following was based on the FAQ at http://www.kernelnewbies.org and "Kernel Rebuild Procedure" by Kwan Lowe at http://www.digitalhermit.com/linux/kernel.html.

First, if you don't have the latest Linux Kernel source code, you need to obtain the newest stable Linux kernel version by downloading the bzip2 tar file(.tar.bz2) from ftp://ftp.xx.kernel.org/ (where xx is the country code and is not case sensitive) and place it in the /usr/src directory.

TIP: Use another system to download it via the web, and then use scp to upload the file to the firewall server. If you can't connect to a DNS server, you can still use the IP address of the system in the command line. For example, to copy from an internal system to the firewall server:

```
scp linux-2.4.18.tar.bz2  root@192.168.2.2:/usr/src/
```

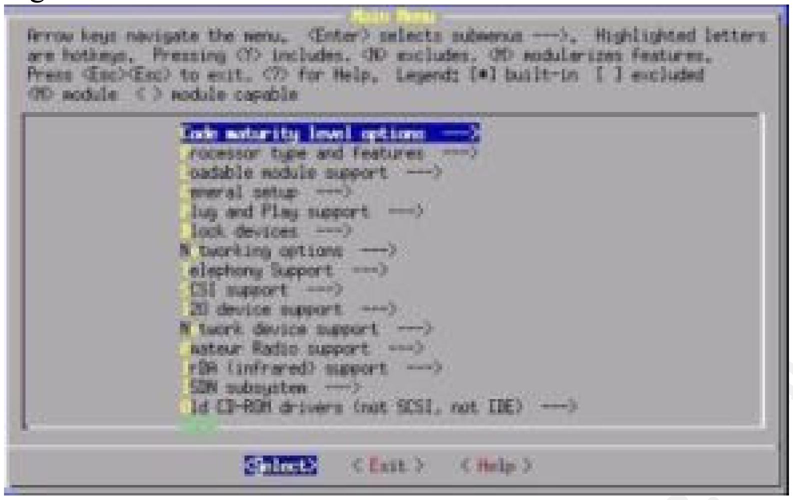
Now, unzip and untar the file using the command:

```
cd /usr/src;  /usr/bin/bzip2 –dc linux-2.4.18.tar.bz2 | tar xvf –
```

Change to the newly created linux directory and select the features you want to compile into the kernel.

This can be done in a few ways, but the easiest is to use the ncurses based interface via the command: make menuconfig. This will present you with a menu screen as seen in figure 11.

Figure 11



```
                                  Main Menu
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters
 are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.
 Press <Esc><Esc> to exit, <?> for Help.  Legend: [*] built-in  [ ] excluded
 <M> module  < > module capable

                  Code maturity level options  --->
                  Processor type and features  --->
                  Loadable module support  --->
                  General setup  --->
                  Plug and Play support  --->
                  Block devices  --->
                  Networking options  --->
                  Telephony Support  --->
                  SCSI support  --->
                  I2O device support  --->
                  Network device support  --->
                  Amateur Radio support  --->
                  IrDA (infrared) support  --->
                  ISDN subsystem  --->
                  Old CD-ROM drivers (not SCSI, not IDE)  --->

                      <Select>    < Exit >    < Help >
```

However, there are a few things you need to know beforehand, such as cpu type, network interface drivers, etc... This information can be gleaned from the output of the dmesg and lsmod commands, along with the boot log file in /var/log/boot.log. Lsmod lists the current loaded kernel modules, while dmesg collects system diagnostic messages from the /var/log/messages file. This output will help you identify the various hardware components of your system, and the necessary kernel modules that must be built into the kernel.

The menuconfig program is an ncurses based menu that allows you to select the various options to compile into the kernel, either as loadable modules, or included in the kernel. As mentioned above, you can create a monolithic kernel that contains all of the needed modules while disallowing any further modules from being loaded. When doing this, it is important to limit the size of the kernel to about one megabyte, so it is important to deselect any options that are not needed, otherwise the size of the kernel may be too large to load.

Table 1 lists various options that should be specifically enabled or disabled. The actions listed indicate how each of the options should be set, regardless of the defaults, unless otherwise specified. The "--->" denotes an additional submenu, while "->" indicates an item in the submenu.

HINT: In the configuration menu, enabled is termed "Built-in" since the selected module will be compiled as part of the core kernel. A disabled option is termed "Excluded" since the module will not be compiled. Finally, we are building a monolithic kernel, so none of the options will be selected as "Module," since loadable module support will be disabled.

While some aspects may depend on the hardware architecture, many are relevant regardless of the hardware platform. The "Networking options" submenu is where various firewall software components are added into the kernel. Table 1 addresses this section more fully. The Network packet filtering option installs the iptables firewall software. This replaces the older default firewall software, ipchains. The tunneling capability is disabled since the router itself will not perform tunneling, and only the IP protocol is supported.

Table 1

Menuconfig options	--->	Action
Code maturity level options	--->	Excluded
Loadable module support	--->	Excluded
Processor type and features --->	--->	
Select Processor Family	->	K6/K6-II/K6-III
Symmetric multi-processing support	->	Built-in, unless only one cpu
General setup	--->	Use defaults
Memory Technology Devices	--->	Excluded
Parallel port support	--->	Excluded
Plug and Play support	--->	Excluded
Block Devices --->		
normal PC floppy disk support	->	Built-in
all other submenu selections	->	Excluded
Multi-device support	--->	Excluded

Networking options --->	--->	
Packet socket	->	Built-in
Netlink device emulation	->	Built-in
Network packet filtering (replaces ipchains)	->	Built-in
Network packet filtering debugging	->	Built-in
Socket Filtering	->	Built-in
Unix domain sockets	->	Built-in
TCP/IP networking	->	Built-in
IP: multicast routing	->	Excluded
IP: advanced router	->	Built-in
IP: policy routing	->	Built-in
IP: use netfilter MARK value as routing	->	Built-in
IP: fast network address translation	->	Built-in
IP: equal cost multipath	->	Excluded
IP: use TOS value as routing key	->	Built-in
IP: verbose route monitoring	->	Built-in
IP: large routing tables	->	Excluded
IP: kernel level auto configuration	->	Excluded
IP: tunneling	->	Excluded
IP: GRE tunnels over IP	->	Excluded
IP: TCP Explicit Congestion Notification	->	Excluded
IP: TCP syncookie support	->	Built-in
IP: Netfilter Configuration --->	--->	
IRC protocol support	->	Excluded
All other submenu selections	->	Built-in
The IPX protocol	->	Excluded
Appletalk protocol support	->	Excluded
DECnet Support	->	Excluded
802.1d Ethernet Bridging	->	Excluded
QoS and/or fair queuing	--->	Excluded
Telephony Support	--->	Excluded
ATA/IDE/MFM/RLL support	--->	As needed
SCSI support	--->	Excluded
Fusion MPT device support	--->	Not implemented
I20 device support	--->	Excluded
Network device support --->	--->	
Network device support	->	Built-in
dummy net driver	->	Built-in
ethernet 10/100 Mbit --->	--->	
EISA, VLB, PCI on board controllers	->	Built-in
RealTek RTL-8139 card support	->	Built-in
All other submenu selections	->	Excluded
Amateur Radio support	--->	Excluded
IrDA support	--->	Excluded
ISDN subsystem	--->	Excluded

Old CD-ROM drivers	--->	Excluded
Input core support	--->	Excluded
Character devices --->	--->	
Virtual terminal	->	Built-in
Support for console on virtual terminal	->	Built-in
Standard/generic serial support	->	Built-in
Unix98 PTY support	->	Built-in
/dev/agpgart (AGP support)	->	Excluded
Direct Tendering manager	->	Excluded
Multimedia devices	--->	Excluded
File Systems --->	--->	
Kernel automounter version 4 support	->	Excluded
Ext3 journaling file system support	->	Built-in
Virtual memory file system support	->	Built-in
Network File Systems all submenu selections	--->	Excluded
Console drivers	--->	Use defaults
Sound	--->	Excluded
USB support	--->	Use defaults
Kernel hacking	--->	Excluded

Once all of the selections have been made, exit the main menu. It will ask if you want to save the new kernel configuration; answer yes. The new configuration is saved as the file ".config" in the /usr/src/linux directory.

To compile the kernel using the new configuration, use the following commands:

```
cd /usr/src/linux  (change to the linux source directory)
make clean         (remove any old compilation of the kernel)
make dep           (build the dependencies)
make bzImage       (compile the kernel )
```

The bzImage is a gzip compressed kernel image, whose layout and loading algorithm allow for a larger capacity than the standard zImage, e.g. a big zImage. (17)

The next step is to move the new kernel image into the boot directory and modify the boot loader (lilo) configuration to allow selecting the new kernel during bootup. First, copy the system map and kernel image to the /boot directory.

TIP: Rather than copy over the existing system map and kernel image, it is better to create new files in the /boot directory. This makes it possible to modify the boot loader configuration so you can select the kernel image that you want to load. If the newly created kernel fails for any reason, you can reboot and select the original kernel that you know is good.

Use the following commands to copy over the system map and kernel image:

```
cd /boot;
cp –p /usr/src/linux/System.map System.map-2.4.18
cp –p /usr/src/linux/arch/i386/boot/bzImage bzImage-2.4.18
rm System.map
ln –s System.map-2.4.18 System.map
```

TIP: If you do this several times, make sure you recheck that the symbolic link continues to point to the new System map. It may be reset if you boot with the original kernel.

Once the files are copied into the /boot directory, the boot loader needs to be configured by editing the /etc/lilo.conf file. Add the following lines to the end of the file:

```
image=/boot/bzImage-2.4.18
        label=2.4.18
        read-only
        root=/dev/hda#    (where # is the same number of the boot disk partition)
```

The final step is to install the new kernel image by running the command:

```
/sbin/lilo
```

TIP: This must be done each time a new kernel is copied into the /boot directory if the system is to boot correctly.

The system is ready to be rebooted. Enter the command reboot at the prompt and wait for the boot loader screen to display your options. You should see two choices, one above the other; “linux” and “2.4.18.” You have a number of seconds to make a selection using the up and down arrow keys. The default is 5 seconds, as set by the timeout parameter in the /etc/lilo.conf file, after which it will load the default image.

TIP: Leave the default image as the original kernel’s label. If anything goes wrong with the new kernel, you can hit the reset button and let the system come up with the original kernel. Once you are certain the new kernel is working as expected, change the “default” parameter in /etc/lilo.conf to “2.4.18”, the label of the new kernel image. Keep the original kernel as an alternate selection in case problems occur. Once testing is complete, and the firewall server is ready to go into service, then you can completely remove the original kernel image.

2.5.3 Implementing the Linux Firewall - iptables

Having hardened the server and rebuilt the Linux kernel to include all necessary system and firewall modules, you can now proceed with implementing the firewall ruleset as developed in section 2.3 above.

The initial installation of the firewall rule base must be done manually by executing the script that contains the firewall ruleset. The script developed in section 2.3 was named “rc.firewall” and was placed in the directory /etc/rc.d. By simply executing this script, iptables will be initialized and the rules added to the iptables chains.

TIP: As you begin this process, you may wish to initialize iptables manually each time the system boots until it is determined to have reached an operational state. At bootup, iptables assumes a permit everything policy by default. This is OK during development work, but not in production service.

Using the command,

```
/etc/rc.d/rc.firewall
```

is all that is needed to start up the firewall.

The following output is displayed when the script in Appendix B is run. It shows how the use of the echo command within the script can help in the debugging process. If any errors occur in the script, it will show up after the echo of the chain name that is currently being loaded with rules.

```
======= Start iptables initialization =======

Part 1 - Variable definitions and iptables initialization

Part 2 - User Defined Chain rules

tcp-state-flags
conn-track

Primary chains

ext-to-svc
svc-to-ext
svc-to-int
int-to-svc
ext-to-int
int-to-ext
EXT-input
EXT-output
SVC-input
SVC-output
INT-input
INT-output

secondary chains

svc-mail
www-query
squid-query
dns-query
ssh-connect
ntp-query
auth-query
syslog-message
db-query
icmp-out
```

```
icmp-in
ext-source-check
svc-source-check
int-source-check
source-address-check
dest-address-check

Part 3 - Start of filter processing

Initialize built-in chains

======= End of initialization ================
```

After the rules have been loaded successfully, it is helpful to get a listing of the loaded rules using the iptables --list command (see Appendix C).

When the ruleset is at the "operational" level, you can make the ruleset automatically startup during bootup by saving the firewall rules to the iptables startup script using the command:

```
/etc/init.d/iptables save
```

This will save the current initialization parameters and ruleset to a file named /etc/init.d/iptables. There are links to this file from the various run level scripts. For example the link /etc/rc.d/rc2.d/S08iptables points to this file /etc/init.d/iptables. Once the system reaches run level 2, it will execute the iptables script via the S08iptables link.

At this point, the firewall server is now fully operations. Each time it boots, it will initialize the network interfaces and start iptables. From this point on, all traffic will adhere to the ruleset that was defined.

If changes are needed due to errors, or improvements to the ruleset, all that is needed is to edit the /etc/rc.d/rc.firewall script, execute it, then save it.

TIP: If you are working on a specific chain of the firewall ruleset, you can use the command:

```
iptables -L chain-name
```

where chain-name is the name of the chain you modified This will display the current order of the rules of that chain. For more information use the command:

```
iptables –vn -L chain-name
```

which displays the number of packets and number of bytes that have been matched by each rule in the chain. This is helpful in optimizing the order of the rules in that chain. A word of caution though, order of rules should be dictated by the rule logic rather than simply by the number of packets matching the rule.

Other useful debugging tricks include using the log prefix facility to indicate when a certain type of packet has either reached or been rejected by a chain. For example, if you are having trouble with ssh, add a logging rule before the drop rule as seen below.

```
  iptables –A ssh-connect –j LOG –log-prefix "Illegal ssh connect: "
  iptables –A ssh-connect –j DROP
```

If a packet destined for port 22 is dropped, there will be a log entry in the /var/log/messages file that will indicate the source and destination addresses. If it is legitimate traffic, but is dropped, you will be alerted and can check the ruleset to see where the logic of that chain failed.

The following is an example from a log file showing an illegal attempted connection from an internal system to an Internet web server on port 80. Since web traffic is only allowed through the web proxy server, the traffic was logged and dropped. It shows a prefix of "Illegal WWW query" because the www-query chain that logged and dropped the packet checked the traffic. After looking up the IP address at the Arin Whois website (www.arin.net/whois/arinwhois.html), it was found to be a Microsoft web site, probably used by an automatic update service on the internal system. The update program needs to be configured to use the web proxy if it is really needed.

```
Aug 22 13:57:33  kernel: Illegal WWW query: IN=eth0 OUT=eth2 SRC=192.168.3.3
DST=65.54.249.126 LEN=44 TOS=0x00 PREC=0x00 TTL=31 ID=8960 DF PROTO=TCP SPT=1030
DPT=80 WINDOW=8192 RES=0x00 SYN URGP=0
```

3 Assignment 3 – Verify the Firewall Policy

The purpose of an audit of the firewall policy is to verify that the firewall ruleset is working as it expected to do. It is usually easy to know if a given service is working for the users. What is more difficult is knowing if there is unauthorized use of these services, or additional services being run that are not allowed by the security policy.

3.1 The Audit Plan

In order to verify the GIAC Enterprises firewall policy, various software tools are used to scan and probe the GIAC Enterprises networks to create a stimulus that may or may not evoke a response. Firewall log files and packet sniffer logs of the firewall interfaces will also be analyzed to determine the success or failure of the firewall policy.

Technical approach:

In order to test the networks fully, the stimulus scans and probes will be issued from a linux based audit system located externally, in the service network, and within the internal network. The audit system contains the necessary software tools to conduct the probes and may be moved from one network to another as needed rather than requiring multiple audit systems. The software utilized includes nmap, available from www.insecure.org/nmap, a network mapping and vulnerability scanner, sara, located at www.cisecurity.org, a specialized version of the Saint vulnerability scanner to detect the SANS top twenty vulnerabilities, as well as simple tools like ping, telnet and nslookup, and a web browser that are available as part of the Redhat Linux installation. The tool icmppush is a program that creates and sends icmp messages and is available from the URL: hispahack.ccc.de/programas/icmpush22.tgz, as well as the tool icmpquery available from www.angio.net/security.

Tcpdump, also part of Redhat Linux, will be used to watch traffic on each of the firewall interfaces, or from systems of interest on the service and internal networks.

Workplace considerations:

First and foremost, permission must be obtained from GIAC Enterprises management to conduct the audit. Management should be presented with the audit plan, and be informed of potential risks as well as when the auditing will take place. If approved, a signed copy of the audit plan should be given to managers who need to know in advance, and a copy to the audit team leader.

Some of the passive probing and service checks that are "system friendly" and have a low risk of causing service failure can be run during normal business hours. Tests that are considered risky should be run during off peak hours. At least one period of off peak hours should be designated as a "scheduled network down time" so that remote employees, customers, suppliers and partners are aware that the system will be unavailable for whatever reason. This gives the audit a chance to run procedures that may crash systems without having to broadcast the fact that an audit is being run.

Risks and concerns:

Since the most significant risk is possible corruption of the e-commerce database, it should be fully backed up prior to the audit, and critical tables taken offline during the scheduled down time before tests are run against the database listener service.

No actual attacks based on vulnerabilities will be run against any operational systems. However, it is possible that certain types of scans, like OS profiling, might have unexpected results, causing the system to crash. In the event a hard crash occurs, and the system will not reboot, the backups can be restore the affected system. NOTE: Since all internal GIAC Enterprises systems are routinely backed up, any significant damage done to any critical server, should be quickly recoverable.

Costs and effort:

It is anticipated that one man week of work will be required to run the actual tests, with an additional man week to produce the audit report and executive summary. In-house IT staff trained in security auditing will conduct this work using software that is in the public domain. As a result, there are no out of pocket expenses for GIAC Enterprises.

Audit Plan:

The audit testing can be broken down into four phases as follows:

- Firewall server host security
- Service network services

- Internal network services
- General checks against vulnerabilities

Firewall host security:

The firewall is audited to make sure that the server is properly defended. The following items should be checked:

- The system is checked to make sure the operating system contains the current level of patches
- A initial tripwire encrypted database has been created for file integrity checking and is stored on the internal syslog server
- Anti-virus software has been installed and contains up to date virus signatures
- Unneeded services and software have been removed
- System has been hardened using SANS Securing Linux Step-by-step guide
- Remote access to the firewall server is limited to only ssh access from the system administrator system

Service network services:

The service network offers web proxy, public web server, e-commerce web server, email, domain name service, network time synchronization and system logging. Each service should be checked for the following:

- Web proxy server allows queries only from internal systems and access only Internet or service network web servers.
- The only accessible http server is the public web server on the service net
- The only accessible https server is the public e-commerce web server on the service net
- The only accessible email server from the Internet is the service net email server
- The DNS server does not respond to queries from Internet clients, and only responds to the internal DNS server, service net clients, or the firewall server.
- The NTP timeserver only acts as a server to the internal NTP server, other service net clients and the border router.
- System logging only occurs for the border router to the service net syslog server
- Auth requests are rejected with a TCP reset
- ICMP traffic is limited as specified by the firewall policy
- All other traffic or services are denied without response

Internal network services:

The internal network services offered include ssh, DNS, NTP, email and a database listener. These services should be checked for the following:

- The ssh server is the only internal system that can be connected to directly from the Internet using the ssh protocol for VPN activities.

- Systems cannot directly connect to Internet web servers, but use the service net web proxy server
- The DNS server can only make requests to the service net DNS server
- The NTP server can only make requests to the service net NTP server
- The email server will relay outgoing email, or receive incoming email only from the service net email server, and that internal systems can only interact with the internal email server for sending or receiving email
- The database listener is only accessible from the service net e-commerce web server
- Auth requests are rejected with a TCP reset to the service network email systems only
- ICMP traffic is limited as specified by the firewall policy
- All other traffic or services are denied without response

General checks against vulnerabilities:

Some additional general tests provide checks against some frequently used attacks, and help prove the GIAC Enterprises networks are acting as a "good Internet Neighbor" by blocking spoofed addresses.

- Source spoofing or use of private or illegal IP addresses is detected and prevented
- Specific problem ports or source addresses are logged and dropped
- Scanning elicits no response from any internal or service net system except for the public web/e-commerce and mail servers.
- The only public DNS information available for the GIAC Enterprises are the fully qualified domain names for the web/e-commerce and mail servers.
- Check against the top twenty common vulnerabilities as reported by the SANS institute

3.2 Conducting the Audit:

After obtaining the permission of GIAC Enterprises management and notifying the appropriate managers as well as the ISP for GIAC Enterprises, the audit was performed according to the audit plan. The results are summarized below in the "Audit Evaluation" section. This section contains commands and sample output used to conduct the audit for each of the four phases described above.

3.2.1 Firewall host security:

To determine what services and software are installed, two commands are useful, lsof to list running services and the rpm command to list installed packages.

The command, lsof –i ,was used to check to see which services were active on the firewall server. The output below shows that only the sshd server was listening on port 22 (ssh) and that there was one connection established from the address 192.168.3.20, which is the system administrator console.

```
COMMAND    PID USER    FD   TYPE DEVICE SIZE NODE NAME
sshd       649 root    3u   IPv4   1219       TCP *:ssh (LISTEN)
sshd      1688 root    4u   IPv4   2258       TCP 192.168.3.1:ssh->192.168.3.20:1028
(ESTABLISHED)
```

The command, "rpm --query --all", displays a listing of all installed packages and can be reviewed to make sure only needed packages have been installed. The rpm command can be used to remove any overlooked packages that are unnecessary. The Firewall tutorial (section 2.5) lists a minimal set of packages that should be installed on the firewall server.

Attempts to access the firewalls' three interface IP addresses from the Internet, the service network or internal systems (except for the system administrator console) produced the following entries in the firewall log file using commands similar to the following:

```
ssh 192.168.2.2 –l root
```

Output from Internet audit system:

```
Aug 26 12:34:31 FW kernel: Illegal ssh connect: IN=eth2 OUT= SRC=xxx.xxx.xxx.xxx
DST=192.168.2.2 LEN=48 TOS=0x00 PREC=0x00 TTL=63 ID=5194 DF PROTO=TCP SPT=47604
DPT=22 WINDOW=24820 RES=0x00 SYN URGP=0
Aug 26 12:35:35 FW kernel: Default drop: IN=eth2 OUT= SRC=xxx.xxx.xxx.xxx
DST=192.168.1.1 LEN=48 TOS=0x00 PREC=0x00 TTL=63 ID=5196 DF PROTO=TCP SPT=47604
DPT=22 WINDOW=24820 RES=0x00 SYN URGP=0
Aug 26 12:36:11 FW kernel: Default drop: IN=eth2 OUT= SRC=xxx.xxx.xxx.xxx
DST=192.168.3.1 LEN=48 TOS=0x00 PREC=0x00 TTL=63 ID=5198 DF PROTO=TCP SPT=47604
DPT=22 WINDOW=24820 RES=0x00 SYN URGP=0
```

Note: The external source address is kept confidential, and is represented by "xxx.xxx.xxx.xxx."

Also Note: The first line uses the "Illegal ssh connect" prefix while the other two indicate the "Default drop." This indicates the first packet was matched and sent to a special chain dealing with the ssh protocol. The other two packets were caught by the default policy of deny everything unless accepted. The latter traffic comes in on interface eth2, the external interface, but is attempting to connect to the IP address of one of the internal facing interfaces. This does not match any of the traffic flow patterns, and is dropped by default.

The service network and internal systems repeated the above pattern, and their output is not shown.

Firewall OS fingerprint:

One item of special interest is to see if nmap can determine the operating system of the firewall. This can be used to target vulnerabilities of the OS. The following nmap command was used:

```
nmap –sS –p 22 –O –P0 –n 192.168.2.2
```

Results:

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find at least
1 open and 1 closed TCP port
All 1100 scanned ports on (192.168.2.2) are: filtered
Too many fingerprints match this host for me to give an accurate OS guess.
```

```
Nmap run completed – 1 IP address (1 host up) scanned in 1551 seconds
```

3.2.2 Service network services:

Attempted access of the various services was tried from the Internet, service network and the internal network locations of the audit system. The results are listed per service.

Queries to the web proxy service:

Attempts to connect to the web proxy port 8008 using the command "telnet 192.168.1.2 8008" failed from external systems, as shown by the syslog entry below.

```
Aug 26 17:42:07 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.2 LEN=40 TOS=0x00 PREC=0x00 TTL=62 ID=31041 DF PROTO=TCP SPT=48156
DPT=8008 WINDOW=24820 RES=0x00 RST URGP=0
```

When the web browser on the external audit system was configured to point to the service net web proxy browser, an error messages stating "The operation timed out when attempting to contact `www.sans.org`". The firewall log also showed a blocked attempt to access port 8008 similar to that above.

Finally, a port scan was run using nmap to scan the service net for port 8008. The nmap command is: nmap –sT –p 8008 –PO –n 192.168.1.0/24. The output is summarized below.

Briefly, the nmap syntax is (refer to the nmap man page from a full description):

-sT TCP	- attempts to perform a tcp connect to every port
-sU UDP	- sends zero length udp packets to each port, if it receives an icmp port unreachable message, then the port is closed, otherwise it assumes it is open
-p #,#-#	- specific port numbers to scan
-F	- fast scan using ports listed in the services file of nmap
-n	- do not perform reverse DNS lookups

The ip address using the netmask or range of addresses can be listed

The output from nmap lists for each IP address scanned in the 192.168.1.0/24 network the port number, state and related service. The state is open, filtered or unfiltered. According to the nmap man page, an open port will accept connections. A filtered state indicates some kind of filter is obstructing nmap from determining the status of the port, so it is unknown if the port is open or not. An unfiltered state indicates the port is known to be closed, is not filtered and will not accept connections.

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.1.0):
Port        State       Service
8008/tcp    filtered    unknown
```

These results were repeated for the remaining addresses 1 through 254.

Queries from the web proxy service:

The internal audit system was configured so that its web browser would use the web proxy server. The attempt to load a web page from the internal web server resulted in an error message from the browser that the connection failed with a system messages of "`(110) Connection timed out`." The following syslog entry shows that the packet from the web proxy server (192.168.1.2) destined for the internal web server (192.168.3.4) was blocked by the default rule.

```
Aug 26 17:55:31 FW kernel: Default drop: IN=eth1 OUT=eth0 SRC=192.168.1.2
DST=192.168.3.4 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=11539 DF PROTO=TCP SPT=1432
DPT=80 WINDOW=5840 RES=0x00 SYN URGP=0
```

Public web / e-commerce server:

The following output is from the nmap program specifically scanning for web servers listening on port 80. The nmap command is: nmap –sT –p 80,443 –PO –n 192.168.1.0/24. The output is summarized below.

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )

Interesting ports on  (192.168.1.0):
Port       State       Service
80/tcp     filtered    http
443/tcp    filtered    https

Interesting ports on  (192.168.1.1):
Port       State       Service
80/tcp     filtered    http
443/tcp    filtered    https

Interesting ports on  (192.168.1.2):
Port       State       Service
80/tcp     filtered    http
443/tcp    filtered    https

Interesting ports on  (192.168.1.3):
Port       State       Service
80/tcp     open        http
443/tcp    open        https
```

This last entry is repeated for the remaining IP address 4 – 254.

Nmap finds that only ip address 192.168.1.3 has ports 80 (http) and 443 (https) open. This was the expected result of the scan. The output from the firewall log also confirms the scans were being blocked.

```
Aug 26 22:34:07 FW kernel: Illegal WWW query: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.4 LEN=60 TOS=0x00 PREC=0x00 TTL=62 ID=56470 DF PROTO=TCP SPT=1859
DPT=80 WINDOW=5840 RES=0x00 SYN URGP=0
Aug 26 22:34:10 FW kernel: Illegal WWW query: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.4 LEN=60 TOS=0x00 PREC=0x00 TTL=62 ID=57011 DF PROTO=TCP SPT=1856
DPT=443 WINDOW=5840 RES=0x00 SYN URGP=0
```

Email server:

The following nmap scan was run on the service network**:**

nmap –sT –p 80,443 –PO –n 192.168.1.0/24

The results were as follows:

The address 0-2 and 4-254 were as follows:

```
Interesting ports on  (192.168.1.3):
Port        State       Service
25/tcp      filtered    smtp
```

The email server at address 192.168.1.3 indicated the following:

```
Interesting ports on  (192.168.1.3):
Port        State       Service
25/tcp      open        smtp
```

DNS server:

The external audit server attempted to connect to the service network DNS server using nslookup produced an error message.

```
nslookup
> server 192.168.1.2
Default server: 192.168.1.2
Address: 192.168.1.2#53
> www.sans.org
;; connection timeout, no servers could be reached
```

The firewall log file also showed a block on this attempt as follows:

```
Aug 26 23:51:59 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.2 LEN=57 TOS=0x00 PREC=0x00 TTL=62 ID=0 DF PROTO=UDP SPT=1025 DPT=53
LEN=37
```

Interestingly, the nmap scan showed that udp port 53 was open on the entire address range, while the tcp scan of port 53 showed all were filtered. The firewall log showed that the packets were being blocked as seen in a excerpt below:

Nmap udp and tcp port scan commands:
nmap –sU –p 53 –PO –n 192.168.1.0/24
nmap –sT –p 53 –PO –n 192.168.1.0/24

```
Aug 26 23:47:03 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.2 LEN=28 TOS=0x00 PREC=0x00 TTL=35 ID=32094 PROTO=UDP SPT=63988
DPT=53 LEN=8
```

```
Aug 26 23:47:15 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.3 LEN=28 TOS=0x00 PREC=0x00 TTL=35 ID=39451 PROTO=UDP SPT=63988
DPT=53 LEN=8
Aug 26 23:47:27 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.4 LEN=28 TOS=0x00 PREC=0x00 TTL=35 ID=52095 PROTO=UDP SPT=63988
DPT=53 LEN=8
```

Syslog service:

An nmap scan was run against the service network to check for the syslog port 514 using the command: nmap –sU –p 514 –PO –n 192.168.1.0/24

Again the results indicated port 514 was open on all addresses, but the firewall log showed the packets were blocked. Note the log prefix of "Illegal syslog message."

```
Aug 27 00:15:18 FW kernel: Illegal syslog message: IN=eth2 OUT=eth1
SRC=xxx.xxx.xxx.xxx DST=192.168.1.2 LEN=28 TOS=0x00 PREC=0x00 TTL=42 ID=29351
PROTO=UDP SPT=58278 DPT=514 LEN=8
Aug 27 00:15:30 FW kernel: Illegal syslog message: IN=eth2 OUT=eth1
SRC=xxx.xxx.xxx.xxx DST=192.168.1.3 LEN=28 TOS=0x00 PREC=0x00 TTL=42 ID=30047
PROTO=UDP SPT=58278 DPT=514 LEN=8
Aug 27 00:15:42 FW kernel: Illegal syslog message: IN=eth2 OUT=eth1
SRC=xxx.xxx.xxx.xxx DST=192.168.1.4 LEN=28 TOS=0x00 PREC=0x00 TTL=42 ID=12351
PROTO=UDP SPT=58278 DPT=514 LEN=8
```

Auth service:

Using the telnet command: telnet 192.168.1.2 113 from the external audit system produced the immediate message "Connection refused." Using tcpdump to sniff the traffic on the service network indicated that the attempted connection was allowed through the firewall, but was responded to with a tcp reset from 192.168.1.2.

```
00:27:24.713302 xxx.xxx.xxx.xxx.2078 > 192.168.1.2.auth: S
2775261970:2775261970(0) win 5840 <mss 1460,sackOK,timestamp 1055271 0,nop,wscale
0> (DF) [tos 0x10]
00:27:24.713845 192.168.1.2.auth > xxx.xxx.xxx.xxx.2078: R 0:0(0) ack 2775261971
win 0 (DF) [tos 0x10]
```

The nmap scan using the command nmap –sT –p 113 –PO –n 192.168.1.0/24 indicated the first IP address was filtered (the firewall service interface), while for all remaining IP addresses, the ports were closed.

ICMP traffic:

The icmppush tool can be used to send icmp messages to system and is a useful tool to test icmp blocking. Running the tool and comparing output from tcpdump and the firewall log will determine if the icmp blocking rules are working correctly.

NOTE: For this assignment, the software would not compile without some effort, but the concept was left in as a useful tool.

The icmpquery tool does allow you to test some basic icmp messages. Using the command icmpquery –m 192.168.5.254 sends an icmp message to the border router to query the address mask or time stamp. The border router detects this and drops the packets with the log entry:

```
Aug 27 16:13:00 192.168.2.1 12009: %SEC-6-IPACCESSLOGDP: list 101 denied icmp
xxx.xxx.xxx.xxx -> 192.168.5.254 (17/0), 1 packet
Aug 27 16:26:12 192.168.2.1 12061: %SEC-6-IPACCESSLOGDP: list 101 denied icmp
xxx.xxx.xxx.xxx -> 192.168.5.254 (13/0), 1 packet
```

Similar testing from the internal audit system to the service network produced similar results as seen by the firewall log entry.

```
Aug 27 16:39:00 FW kernel: Illegal outgoing ICMP IN=eth0 OUT=eth1
SRC=192.168.3.123 DST=192.168.1.2 LEN=32 TOS=0x00 PREC=0x00 TTL=254 ID=4321
PROTO=ICMP TYPE=17 CODE=0
```

All remaining traffic:

An nmap scan of the service network IP address range was performed using the command:

```
nmap –sT –F –PO –n 192.168.1.0/24
nmap –sU –F –PO –n 192.168.1.0/24
```

Other than the traffic mentioned above, no other ports were listed as open. All UDP ports were listed as filtered. The syslog file was filled with messages of dropped traffic. A sample is given below:

```
Aug 27 01:07:23 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.4 LEN=60 TOS=0x00 PREC=0x00 TTL=62 ID=3766 DF PROTO=TCP SPT=3509
DPT=458 WINDOW=5840 RES=0x00 SYN URGP=0
Aug 27 01:07:23 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.4 LEN=60 TOS=0x00 PREC=0x00 TTL=62 ID=25864 DF PROTO=TCP SPT=3510
DPT=2601 WINDOW=5840 RES=0x00 SYN URGP=0

Aug 27 11:36:07 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.4 LEN=28 TOS=0x00 PREC=0x00 TTL=38 ID=30917 PROTO=UDP SPT=40657
DPT=56 LEN=8
Aug 27 11:36:07 FW kernel: Default drop: IN=eth2 OUT=eth1 SRC=xxx.xxx.xxx.xxx
DST=192.168.1.4 LEN=28 TOS=0x00 PREC=0x00 TTL=38 ID=45846 PROTO=UDP SPT=40657
DPT=7009 LEN=8
```

3.2.3 Internal network services:

Ssh service:

Attempts to connect to internal network systems, other than the ssh server, from the external audit system were blocked and logged. Attempts from the service net audit system were also blocked and logged.

```
Aug 27 12:06:01 FW kernel: Illegal ssh connect: IN=eth2 OUT=eth0
SRC=xxx.xxx.xxx.xxx DST=192.168.3.3 LEN=60 TOS=0x00 PREC=0x00 TTL=62 ID=32130 DF
PROTO=TCP SPT=3521 DPT=22 WINDOW=5840 RES=0x00 SYN URGP=0
Aug 27 12:15:59 FW kernel: Illegal ssh connect: IN=eth1 OUT=eth0 SRC=192.168.1.123
DST=192.168.3.3 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=11011 DF PROTO=TCP SPT=1436
DPT=22 WINDOW=5840 RES=0x00 SYN URGP=0
```

Nmap scans using the command, nmap –sS –p 22 –PO –n 192.168.3.0/24, indicated the service was filtered.

Direct Internet web access:

When the web browser of the internal audit system was configured for direct Internet connections, the traffic was blocked and logged as an "Illegal WWW query" showing both the source IP and the intended destination web server.

```
Aug 27 12:22:37 FW kernel: Illegal WWW query: IN=eth0 OUT=eth2 SRC=192.168.3.123
DST=63.100.47.46 LEN=44 TOS=0x00 PREC=0x00 TTL=31 ID=9472 DF PROTO=TCP SPT=1030
DPT=80 WINDOW=8192 RES=0x00 SYN URGP=0
```

DNS service:

Attempts to connect to the service net DNS server or other Internet DNS servers by the internal network auditing system were blocked, as shown in the example log entry.

```
Aug 27 11:54:24 FW kernel: Illegal DNS query: IN=eth0 OUT=eth1 SRC=192.168.3.123
DST=192.168.1.2 LEN=56 TOS=0x00 PREC=0x00 TTL=31 ID=5632 PROTO=UDP SPT=1027 DPT=53
LEN=36
```

NTP service:

NTP configuration changes on the internal audit system to a service net or external NTP server created log entries for dropped traffic.

```
Aug 27 12:31:11 FW kernel: Default drop: IN=eth0 OUT=eth2 SRC=192.168.3.123
DST=xxx.xxx.xxx.xxx LEN=76 TOS=0x00 PREC=0x00 TTL=63 ID=0 DF PROTO=UDP SPT=1035
DPT=123 LEN=56
```

Email service:

The internal audit system was configured to use the service network email server. Attempts to send email were blocked and logged with the message "Illegal Mail connect."

```
Aug 27 12:44:36 FW kernel: Illegal Mail connect: IN=eth0 OUT=eth1
SRC=192.168.3.123 DST=192.168.1.2 LEN=44 TOS=0x00 PREC=0x00 TTL=31 ID=15872 DF
PROTO=TCP SPT=1034 DPT=25 WINDOW=8192 RES=0x00 SYN URGP=0
```

Auth queries:

Auth queries from the service net audit system to the internal audit system, or vice versa, resulted in TCP reset packets that were logged.

```
Aug 27 12:50:57 FW kernel: Reject auth query: IN=eth0 OUT=eth1 SRC=192.168.3.123
DST=192.168.1.123 LEN=44 TOS=0x00 PREC=0x00 TTL=31 ID=19456 DF PROTO=TCP SPT=1036
DPT=113 WINDOW=8192 RES=0x00 SYN URGP=0
Aug 27 12:52:01 FW kernel: Reject auth query: IN=eth1 OUT=eth0 SRC=192.168.1.123
DST=192.168.3.123 LEN=60 TOS=0x10 PREC=0x00 TTL=63 ID=13770 DF PROTO=TCP SPT=1437
DPT=113 WINDOW=5840 RES=0x00 SYN URGP=0
```

ICMP traffic:

Similar tests and results as those on the service net, were encountered on the internal network with icmpush and icmpquery commands.

All remaining traffic:

An nmap scan of the internal network IP address range was performed using the command:

```
nmap –sT –F –PO –n 192.168.3.0/24
nmap –sU –F –PO –n 192.168.3.0/24
```

Other than the traffic mentioned above, no other ports were listed as open. All UDP ports were listed as filtered

3.2.4 General checks against vulnerabilities:

Nmap was used to generate tcp SYN packets with spoofed source addresses using the command,:

```
nmap –sS –p 22 –n –P0 –S912.168.3.2 192.168.3.30
```

The external audit system created log entries on the border router when nmap was run.

```
Aug 27 13:18:56 192.168.2.1 11343: %SEC-6-IPACCESSLOGP: list 101 denied tcp
192.168.3.1(34396) -> 192.168.3.30(22), 1 packet
```

Both the service and internal audit system using a similar command had packets blocked and logged by the "martian source" module loaded as part of the iptables software before they were even tested by the user-defined chains designed for that purpose.

```
Aug 27 13:37:53 FW kernel: martian source 192.168.3.2 from 172.16.1.1, on dev eth1
Aug 27 13:38:40 FW kernel: martian source 192.168.1.2 from 172.16.1.1, on dev eth0
```

Similar results were obtained for the various private address spaces or illegal address, as well as GIAC Enterprises network addresses used as source addresses on packets incoming to that network.

Problem addresses and ports dropped:

Since these are specific addresses or ports that are dropped silently, there are no log entries to examine. However, using the iptables –vn –L chain_name command, the number of packets dropped by the rule can be verified as seen by a sampling of the output listed below:

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target prot opt in     out source     destination
    4   184 DROP   tcp  --  *      *   0.0.0.0/0  0.0.0.0/0   tcp dpt:23
```

It can be seen that 4 packets sent by the external audit system using telnet port 23 to the internal audit system where dropped. Similar results were seen for the firewall interfaces (INPUT chain), but no packets have been seen by the rule dropping tcp port 139 traffic.

No response from non-public systems:

Using tcpdump to sniff packets while nmap was running indicated no ICMP response messages or TCP resets were returned when unused ports were scanned, regardless of whether the system in fact existed or not. The exception being the auth protocol on TCP port 113 which sent TCP reset packets for all addresses, even if no system utilized the address.
A partial listing of the tcpdump output is shown below for a scan of TCP port 111.

```
13:57:22.052782 xxx.xxx.xxx.xxx.50457 > 192.168.3.3.sunrpc: S
3293352326:3293352326(0) win 4096
13:57:28.073087 xxx.xxx.xxx.xxx.50458 > 192.168.3.3.sunrpc: S
2473971776:2473971776(0) win 4096
13:57:34.092904 xxx.xxx.xxx.xxx.50459 > 192.168.3.3.sunrpc: S
3092384210:3092384210(0) win 4096
13:57:40.112776 xxx.xxx.xxx.xxx.50460 > 192.168.3.3.sunrpc: S
3293352326:3293352326(0) win 4096
13:57:46.133011 xxx.xxx.xxx.xxx.50461 > 192.168.3.3.sunrpc: S
2473971776:2473971776(0) win 4096
13:57:52.145443 xxx.xxx.xxx.xxx.50456 > 192.168.3.4.sunrpc: S
3743254509:3743254509(0) win 4096
13:57:58.153098 xxx.xxx.xxx.xxx.50457 > 192.168.3.4.sunrpc: S
153249326:153249326(0) win 4096
13:58:04.172893 xxx.xxx.xxx.xxx.50458 > 192.168.3.4.sunrpc: S
2713385130:2713385130(0) win 4096
13:58:10.192788 xxx.xxx.xxx.xxx.50459 > 192.168.3.4.sunrpc: S
3743254509:3743254509(0) win 4096
13:58:16.212803 xxx.xxx.xxx.xxx.50460 > 192.168.3.4.sunrpc: S
153249326:153249326(0) win 4096
13:58:22.232780 xxx.xxx.xxx.xxx.50461 > 192.168.3.4.sunrpc: S
2713385130:2713385130(0) win 4096
```

Domain Name information:

Using nslookup to dump the domain information for giacenterprises.com produced the following results: (NOTE: These results are fictitious and used only as an example of what might be expected as output for the purposes of this assignment.)

```
nslookup
> set querytype=all
> ls -d giacenterprises.com
mail                    1H IN MX        10 mail
                        1H IN MX        20 my.isp.com.
                        1H IN A           192.168.1.2

www                     1H IN A           192.168.1.3
```

SANS Top Twenty Vulnerabilities:

The SANS web site lists the most common vulnerabilities at the URL: www.sans.org/top20.htm. It is a best practice to make sure the systems and networks are secure against these specific vulnerabilities. Referenced on the web page is a modified vulnerability scanner named "sara" that will specifically scan for the listed vulnerabilities. The URL is www.cisecurity.org and the source code (tar ball) can be downloaded and installed on the auditing system.

NOTE: Due to the test nature of this assignment and available resources, this scanner was not in fact run. In the course of an actual audit, this step would be fully taken. Although some aspects would be redundant to some of the tests performed above, independent verification using alternate tools is helpful, especially when such a tool is free and concise in its vulnerability focus.

3.2.5 Audit Results Summary

Checklist: Firewall host security	Status:	Comments:
operating system with current level of patches	OK	Linux kernel 2.4.18 & RedHat Linux 7.3
Initial tripwire encrypted database	OK	Database created 8/5/02 stored on sysadm console
Anti-virus software and signatures up-to-date	OK	McAfee SuperDat file 4160 as of 8/21/02
Unneeded services /software removed	OK	Ssh only, Minimal installation
Host hardening	OK	See worksheet from Securing Linux Stp-by-step
Remote access to the firewall server is limited	OK	Verified (see output section 3.2.1 above)
Firewall OS fingerprint	OK	nmap

Checklist: Service network services	Status:	Tools:
Web proxy server allows queries only from internal systems and access only Internet or service network web servers	OK	telnet, web browser nmap scan on port 8008
The only accessible http server is the public web server on the service net	OK	Web browser and nmap on port 80
The only accessible https server is the public e-commerce web server on the service net	OK	nmap scan on port 443
The only accessible email server from the Internet is the service net email server	OK	Nmap scan on port 25
The DNS server does not respond to queries from Internet clients, and only responds to the internal DNS server, service net clients, or the firewall server	OK	Nslookup and nmap scan of port 53
The NTP time server only acts as a server to the internal NTP server,	OK	nmap scan tcp port 123

other service net clients and the border router		
System logging only occurs for the border router to the service net syslog server	OK	nmap scan udp port 514
Auth requests are rejected with a TCP reset	OK	Nmap scan of tcp port 514 and telnet
ICMP traffic is limited as specified by the firewall policy	OK	tcpdump icmppush, icmpquery
All other traffic or services are denied without response	OK	nmap fast scan all ports

Checklist: Internal network services	Status:	Tools:
The ssh server is the only internal system that can be connected to directly from the Internet using the ssh protocol for VPN activities	OK	Ssh, nmap scan port 22
Systems cannot directly connect to Internet web servers	OK	Web browser
The DNS server can only make requests to the service net DNS server	OK	Name service config
The NTP server can only make requests to the service net NTP server	OK	NTP config
Internal systems can only interact with the internal email server for sending or receiving email	OK	Email config
The database listener is only accessible from the service net e-commerce web server	OK	Nmap scan port 1523
Auth requests are rejected with a TCP reset to the service network email systems only	OK	telnet on port 113
ICMP traffic is limited as specified by the firewall policy	OK	tcpdump icmppush icmpquery
All other traffic or services are denied without response	OK	nmap scans

Checklist: General checks against vulnerabilies	Status:	Tools:
Source spoofing or use of private or illegal IP addresses is detected and prevented	OK	nmap with spoofed addressing to port 22
Specific problem ports or source addresses are dropped	OK	Telnet iptables --list
Scanning elicits no response from any internal or service net system except for the public web/e-commerce and mail servers.	OK	nmap and tcpdump
Only public DNS information available for GIAC Enterprises are fully qualified domain names for web/e-commerce and mail servers	OK	nslookup
Check against the top twenty common vulnerabilities as reported by the SANS institute	TO-DO	sara

3.3 Audit Evaluation

The audit results summarized in section 3.2.5 above indicate that all services and checks are working as designed.

The firewall server is well guarded with a minimally installed set of software. The latest stable linux kernel is installed and was recompiled to include iptables and disable loadable kernel modules. Up to date file integrity and virus checking helps insure the system remains clean. Excess services have been disabled with the only remote access to the system via ssh on the internal interface. Furthermore, the firewall wall's address is not listed publicly, nor is operating system fingerprinting successful. If the route through the firewall is detected, it will be more difficult for an attacker to target specific vulnerabilities against the system.

The service network does have public ports that can be identified by scanning. However, since these three public ports, http, https and smtp, are available in the IPS's DNS table, nothing is to be gained by the scan. Since the DNS and NTP servers are acting as clients to Internet DNS and NTP servers, they do not have visible public ports for those services. Remote access to these systems is limited to ssh from the system administrators console. One service, Auth, does present a question. Since all other ports do not respond, while Auth returns a TCP reset for all ip addresses, it becomes obvious that there is some general filtering in place. Granted, nmap indicates this in its output with the "filtered" state. Since this usually is a result of nmap not receiving icmp messages that it expects, the filter could just be a simple router, rather than a more complex firewall. Most routers today employ some basic level of filtering capability, even if they are designed for a small office or home. Thus the presence of an unexpected tcp reset for auth makes it more apparent a higher level of filtering sophistication is being used. Though auth can be used to speed up delivery of email, it may be questionable as to whether or not to support it.

The internal network is likewise behaving as expected. Traffic is primarily limited to using service net system as a proxy for all Internet bound services. The only exception is for external ssh connections used for VPN purposes. Its visibility from either the Internet or the service net is limited to only the open ssh port on the ssh server. All other systems act in client mode and do not service requests from outside its network.

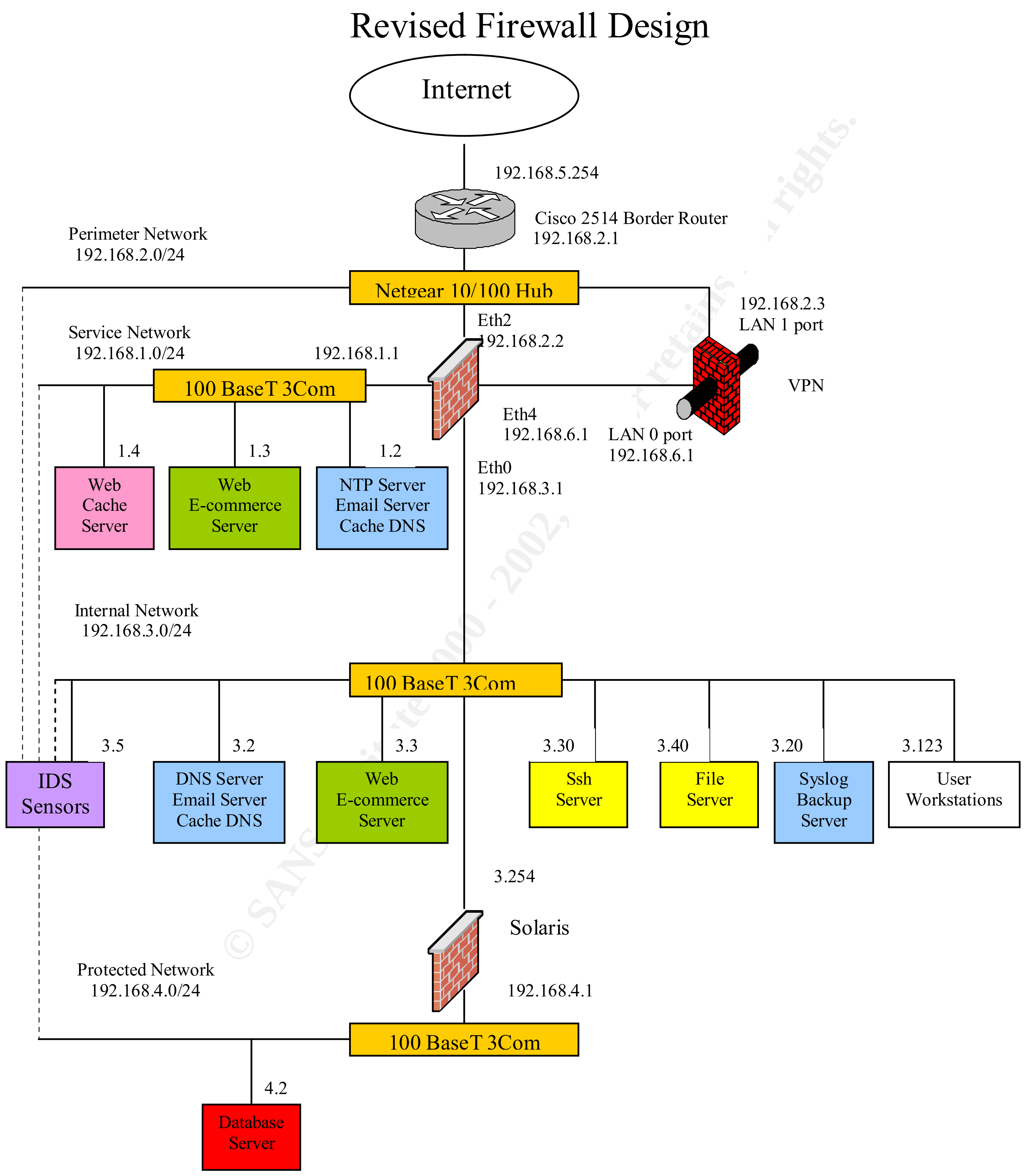
The general vulnerabilities checks indicate that overall, the network footprint is minimal. Little information can be obtained about the various systems. However, it is notable that it can be easily determined that at least two networks do exist by the difference in the network addresses for the public servers (email and web) and the ssh server on the internal network. If the router's internal network address can be determined, than the obvious tri-homed firewall architecture becomes apparent to a would-be hacker, giving them insight into how to further probe the system.

Recommendations:

There are four main recommendations to the overall perimeter architecture for GIAC Enterprises. The first addresses the issued brought up by the address scheme for the public and ssh server. It would be better if all addresses indicated a single network. To provide this footprint, network address translation could be utilized by the firewall server to map the public address to the real server address. It might require further subnetting off a small portion of the perimeter net using perhaps a 29 bit or even 30 bit network mask, and routing the remaining addresses to the firewall gateway for "NATting." The end result would be a single network signature. Security through obscurity is not much help, but it does make the attackers job harder if they think it is a simple network behind a simple router.

The second recommendation is to change from an ssh based VPN to a VPN device such as the Cisco 3005 VPN concentrator, placing its public interface in the perimeter net, and its internal interface into a forth network interface on the firewall (see figure 12). This would allow the VPN to do the decryption of traffic and subject the traffic to the firewall rules. This provides additional protection

Figure 12



Revised Firewall Design
Internet
192.168.5.254
Cisco 2514 Border Router
192.168.2.1
Perimeter Network
192.168.2.0/24
Netgear 10/100 Hub
192.168.2.3
LAN 1 port
Eth2
192.168.2.2
Service Network
192.168.1.0/24
192.168.1.1
100 BaseT 3Com
VPN
Eth4
192.168.6.1
LAN 0 port
192.168.6.1
1.4
1.3
1.2
Web Cache Server
Web E-commerce Server
NTP Server Email Server Cache DNS
Eth0
192.168.3.1
Internal Network
192.168.3.0/24
100 BaseT 3Com
3.5
3.2
3.3
3.30
3.40
3.20
3.123
IDS Sensors
DNS Server Email Server Cache DNS
Web E-commerce Server
Ssh Server
File Server
Syslog Backup Server
User Workstations
3.254
Solaris
Protected Network
192.168.4.0/24
192.168.4.1
100 BaseT 3Com
4.2
Database Server

when the supplier or partner networks may not be as secure as you would like them. It would also provide for quicker access, while still maintaining the single network look.

The third recommendation is that some level of failover be built into the network. Due to budget constraints, the systems used are older, secondhand or minimally configured hardware. These are possible "single point of failure" items within the overall design. As soon as it is tenable, either pre-configured spares should be made available to decrease downtime, or better yet a fail-over and load balancing capability would be preferred. Certainly GIAC Enterprises must learn to walk before running, but they don't want to crawl for long the in their current endeavor.

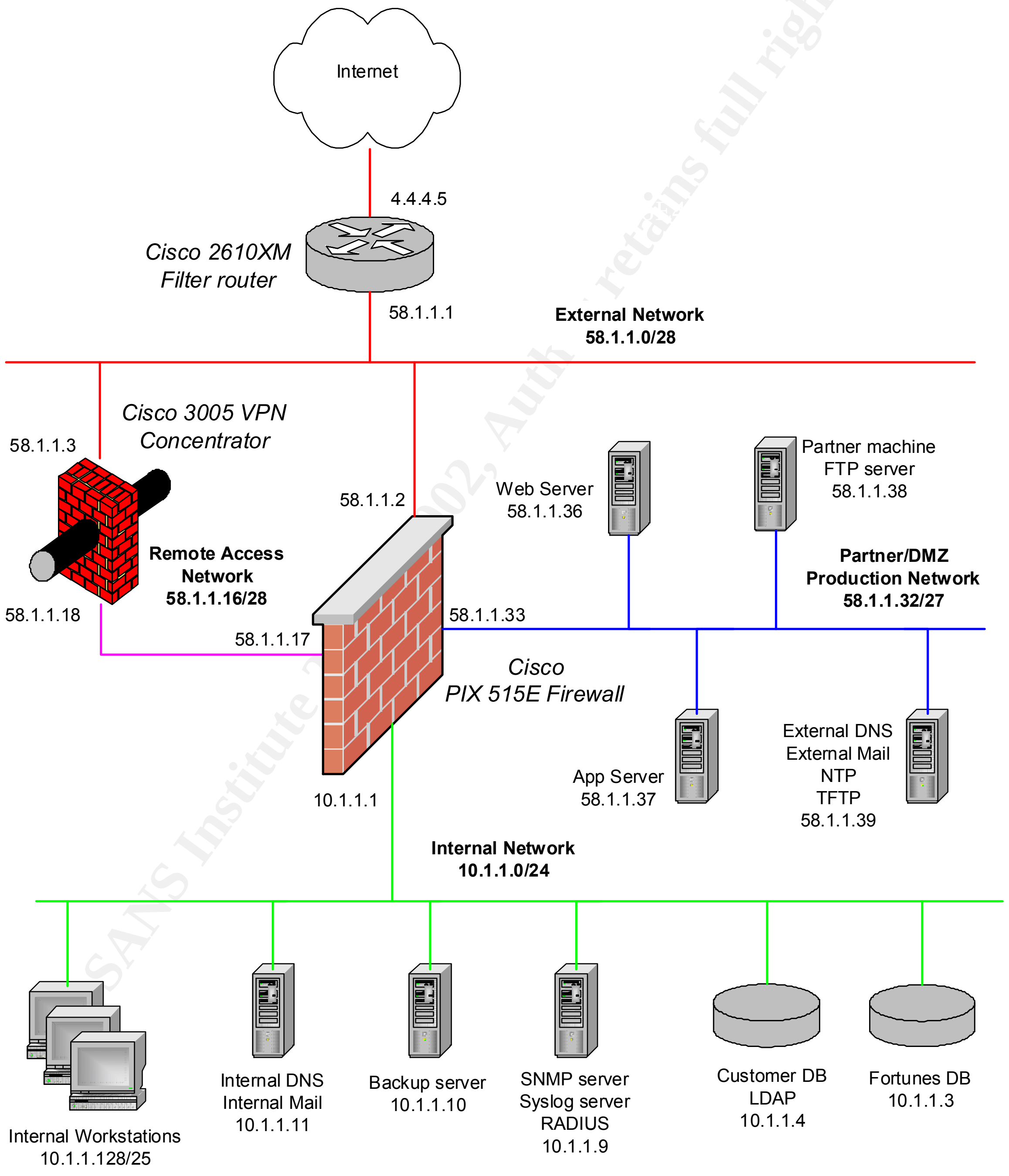
The fourth recommendation has to do with iptables' "martian source" capability. This module, built into the kernel, is able to detect all of the address spoofing and illegal address usage that the user-defined chains in the firewall ruleset were designed to do. By removing these chains, it decreases the number of rules each packet must traverse, since these rules were applied high in the rule set, but after connection checking was completed. Although this removes a layer of security, there already exist two layers at the external side to check this specific concern, namely the border router and the Martian source module. The internal exchanges between the service and internal networks would solely rely on the Martian source module, but since these networks are under GIAC Enterprises control, the risk should be low enough to rely on a single layer of testing, especially if audits such a s this continue to affirm that the traffic is being handled correctly.

The final recommendation would be to hire an outside security audit service to perform and independent audit on a periodic basis. This would provide a different approach to the network and might provide additional insights or catch problems that were overlooked by GIAC Enterprises IT staff.

4. Assignment 4 – Design Under Fire

For the purposes of this assignment, I have selected the firewall design of Steve Keifling. The firewall design is laid out in figure 13.

Figure 13


Internet
4.4.4.5
Cisco 2610XM
Filter router
58.1.1.1
External Network
58.1.1.0/28
Cisco 3005 VPN
Concentrator
58.1.1.3
58.1.1.2
Web Server
58.1.1.36
Partner machine
FTP server
58.1.1.38
Partner/DMZ
Production Network
58.1.1.32/27
Remote Access
Network
58.1.1.16/28
58.1.1.18
58.1.1.17
58.1.1.33
Cisco
PIX 515E Firewall
App Server
58.1.1.37
External DNS
External Mail
NTP
TFTP
58.1.1.39
10.1.1.1
Internal Network
10.1.1.0/24
Internal Workstations
10.1.1.128/25
Internal DNS
Internal Mail
10.1.1.11
Backup server
10.1.1.10
SNMP server
Syslog server
RADIUS
10.1.1.9
Customer DB
LDAP
10.1.1.4
Fortunes DB
10.1.1.3

4.1 Attack against the firewall

With the focus of perimeter security often looking outward to protect the internal system against attack, it is important to keep in mind the possibilities of attack against the firewall server coming from the inside, either by a compromised internal system, or the malicious intent of an insider, be it an employee or a partner who has been granted access to the internal network via a VPN.

A security advisory was posted June 21, 2002 entitled "Weak Cisco PIX Enable Password Encryption Algorithm." The advisory is available at the URL: archives.neohapsis.com/archives/vulnwatch/2002-q2/0121.html. The authors contend that this vulnerability allows the attacker to perform a brute force attack against the PIX password hashes. Once the password is obtained, than the attacker can gain access to the PIX firewall and change its configuration at will.

The authors summarize the vulnerabilities as follows:

Cisco PIX passwords are limited to a length of 16 Bytes, so in theory there are 255^16 possible passwords, but in real life there are about 80^16 useful password combinations, take a look at your keyboard to verify, even if strong passwords are used.

Cisco's password encryption is based on base64 encoded MD5 hashes. Routers IOS uses 1000 MD5 Update rounds to make password brute forcing attacks harder, but the PIX firewall uses only one MD5 update and then the digest is base64 encoded.

For base64 encoding Cisco uses the _crypt_to64() Function of the FreeBSD libcrypt library.

Here's the code to compute PIX password hashes:

```
 ==========================================================

          MD5Context ctx1;
          unsigned char final[MD5_SIZE+1];
          unsigned char cleartext [16+1];
          unsigned char cisco_encoded [16+1];

          memset(cisco_encoded,0,sizeof(cisco_encoded));
          memset(cleartext,0,sizeof(cleartext));
          strcpy((char*) cleartext,"test");

          MD5Init2(&ctx1);
          MD5Update2(&ctx1,(unsigned char*) cleartext,16);
          MD5Final2(final,&ctx1);

          char* p = (char*) cisco_encoded;
          _crypt_to64(p,*(unsigned long*) (final+0),4); p += 4;
          _crypt_to64(p,*(unsigned long*) (final+4),4); p += 4;
          _crypt_to64(p,*(unsigned long*) (final+8),4); p += 4;
          _crypt_to64(p,*(unsigned long*) (final+12),4); p += 4;

 ==========================================================
Due to some weaknesses in the MD5 hash algorithm (den Boer and Bosselaers  found a
so called pseudo-collision) there may be more effective attacks in the future.
```

For an attack of this kind to succeed, there are several hurdles an inside attacker must clear. First off, the attacker must be able to capture the PIX configuration file. The configuration file contains the password hash against which the above brute password attack will be launched offline. The line of interest is listed in the firewall-under-fire's configuration file as:

```
  enable password 8Ry2YjIyt7RRXU24 encrypted
```

Likely methods for capture include accessing the stored configuration file from a TFTP server or backup media. It also might be possible to capture the configuration by sniffing traffic during an upload via TFTP, or telnet, or even possibly as an email message. This may be more likely during an initial installation period, when the PIX firewall is being deployed, and modifications to the firewall configuration are more frequent. For an established firewall, the backup media would be the most likely target. Finally, it is possible the configuration file is "posted" in the room with firewall server, or on a system administrators web notes pages.

If the attacker is able to obtain the enable password, they must connect to the PIX firewall in an interactive session. Since the default is to limit interactive sessions to the console port, the attacker would either have to have physical access to the system, or wait until remote sessions were enabled. Remote access is usually the norm for busy system administrators.

The configuration file does enable ssh and https, but not telnet. However, tftp is used to copy the configuration to a remote tftp server. This does present an opportunity, albeit slim, to grab the configuration file. Furthermore administrative access is granted to all internal workstations, overriding the default of a single administrative address. If the user was able to crack the enable password, they would be able to log in from any internal workstation using ssh or the https interface.

Finally, the password itself is key, especially against a brute force attack. A strong password and a database that uses "salted" passwords will make the task of password cracking that much more difficult. (lists.insecure.org/bugtraq/2002/Jul/0142.html)

The PIX firewall in this design does use alternate authentication, as seen by the configuration lines:

```
   !Enable TACACS
     aaa-server mytacacs protocol tacacs+
     aaa-server RADIAS protocol radius
     aaa-server LOCOL protocl local
```

If this kind of attack succeeded, the attacker would be able to modify the firewall policy to their own designs. If review of the firewall policy was not routine and thorough, the attacker could conduct activities for other internal systems without being detected. They might compromise confidential company information, or simply using the system to conduct attacks against other, more lucrative targets, covering their tracks in the process.

To defeat this kind of attack, it might be recommended to limit ssh and https access to a single administrative address, making access to the PIX firewall more strictly controlled. A review of the configuration file security, looking at the backup procedure and storage, as well as independently encrypting the stored configuration file, would complete a defense against this kind of attack.

4.2 Denial of Service Attack

Sometimes, in the rush to fix serious vulnerabilities that have been recently discovered, a patch may create another unforeseen vulnerability in the process. This was the case with a vulnerability posted June 27th, 2002, that occurred in the SSH protocol in Cisco PIX Firewall version 5.2 and 5.3. (Cisco VU#945216 or CVE-2001-0572). This was a significant vulnerability that permitted "the insertion of arbitrary commands into an established SSH session to collect information that may help in brute force key recovery, or brute force a session key." (www.cisco.com/warp/public/707/SSH-multiple-pub.html)

About eight hours later a second advisory was issued indicating that a possible Denial of Service vulnerability had been introduced into the software while trying to resolve the previous vulnerability. If the attacker tried to exploit the previous vulnerability, the SSH module would consume too much of the processor's time, causing the system to crash and possibly reboot. (www.cisco.com/warp/public/707/SSH-scanning.shtml) If the attacker repeatedly tried to exploit VU#945216, this would cause a Denial of Service against the PIX firewall. An added consequence might be confusion in associating the attack signature with the system crash.

It is often the case that when a significant vulnerability occurs, a lot of messages go out alerting system administrators to the risk. In the barrage of information, such closely spaced alerts might be misinterpreted or the second alert overlooked. The first patch might be applied thinking nothing more needs to be done concerning these kinds of vulnerabilities.

An effective attack could be mounted using a pool of compromised systems that utilized the SSH vulnerability from various systems over a period of time. The attack could bring the network to a halt by continually bombarding the firewall with ssh queries which caused it to reboot or crash. The system would be useless until it was patched. Another method would be to execute the attack more slowly from the various compromised systems taking advantage of the confusion to extend the attack.

There are several tools available to available for a distributed Denial of Service attack. Nessus (online.securityfocus.com/tools/201) is a client/server vulnerability scanning tool. Nessus is available for both Unix and Windows platforms. The server portion of Nessus would be installed on the compromised hosts. A client makes request to the server to run a specific "plugin", which will scan for a specific vulnerability. The VU#945216 vulnerability is available as a Nessus plugin (id 10972) from cgi.nessus.org/plugins/dump.php3?id=10972.

Once the Nessus server is installed on compromised hosts, then all than needs to be done is for the client to make requests at the desired interval to start the scan, and the resultant Denial of Service.

The most obvious defense against this is first off is to install the more recent fix of the software. The other possible defense is to block all SSH connections. In the case of the design under attack, the only ssh connections allowed are from internal systems, so either the attacker must have already compromised an internal host and installed Nessus on it, or they would have to have access as an insider.

4.3 Compromise of an internal system

In the constant struggle to keep software up-to-date with fixes and patches, or newest releases, it is possible that malicious software can be installed within you system. When security alerts prompt the mass downloading of critical software, it provides an interesting opportunity to inject malicious software.

On June 26th, CERT advisory CA-2002-18 was released indication a significant vulnerability in OpenSSH versions 2.3.1p1 through 3.3. Since ssh is a primary tool in secure administration of a network, key systems are usually upgraded when such alerts occur.

On August 1st, CERT sent out another advisory, CA-2002-24, that the openSSH software on the ftp.openssh.com and ftp.openbsd.org sites had contaminated with a Trojan sometime on the 30th or 31st of July. When the software is compiled, it executes code that attempts to connect to a fixed address on port 6667. Anyone who was able to impersonate that address would then receive connections from the "user" who installed the software. Often, this user would be root.

If the Trojaned copy of the software had been installed on this firewall system, it might have been possible to get root access. If so, it would be a simple matter of installing a root kit. Even more difficult to detect, would be a kernel root kit. Kernel rootkits are extremely difficult to detect, since it is the operating system itself that has been compromised. Normal integrity checkers do not usually work, because the system calls they make to determine file names and sizes have been altered, so as not to reveal files installed by the hacker. The system looks normal.

The above is more of an example of why it is important to verify the signatures of all downloaded software and patches, than it is a likely compromise against this firewall. I thought it was an interesting, opportunistic attack that was worth mentioning. Along those lines …

Selecting the target – Internal User:

It is difficult to gain access into a well-secured system. One of the few ways to gain a foothold without detection is to prompt the user into downloading software that would open a legitimate channel through the firewall.

Attack method – covert channel:

The goal would be to get a user to download, either via http, https or email, a binary file that would install a "server" which could be controlled from an external client in a method similar to nessus above. The tool of choice would be "Reverse ww Shell" written by van Hauser, a legendary member of THC (The Hackers Club). Reverse www shell, once installed on an internal system uses http requests and responses to send commands to the server from the client. This is referred to as a covert channel attack (19). Since web traffic is commonly allowed on most networks, chances are good that reverse www shell would be successful.

Obviously, the delivery of the malicious software is the key. Most systems have some kind of email virus and Trojan detection software. Since reverse www shell can be rewritten into the C language and compiled for the platform needed, it may be possible to slip the program by the usual anti-virus checking software. (18) It would be easy to test this before hand.

A second factor is whether or not the email server is configured to allow through attachments, or if some other add-on filter is present which prevents the delivery of attached binary code. By checking the DNS entries for the company, the email server could be located. Attempts to connect to the email server on port 25 could determine what software is running. Telnetting to port 25 of an email server can return the messages:

```
  mail.giacfoo.com ESMTP Sendmail 8.11.6/8.11.6; Wed 28 Aug 2002 17:00:46
```

If sendmail is apparent, it may be possible that binary files are not checked for on a Unix system and are forwarded on through. Maybe, maybe not.

In order to trick a user into downloading the software, a possible message aimed at social engineering, would be to request the user to click on attachment to watch the dancing dogs, or download the latest and greatest software plugin for their browser. A similar approach could be run from a malicious web site that was referenced with a link in the email message. A window might pop up, looking very official, prompting the user to download the necessary software.

If the installation was successful, the reverse www shell periodically attempts to contact the attacker at a preconfigured address. The system looks as if it is really just surfing the Internet. Once a connection is made to the attacker's system, the reverse www shell provides the command line prompt to the attacker, permitting him to execute arbitrary commands. The commands are passed out as if they are actual http protocol GET commands.

There is a good chance that this method of attack could succeed against this firewall design for two reasons. First, the firewall allows direct connections to the Internet. As a result there is no control of the traffic that passes between the Internet and internal users through using port 80. Secondly, since a web proxy server is not used, there is no easy way to determine what URL's are being accessed and if the requests were adhering to http protocol rules (stateful inspection). Therefore, this traffic will go unnoticed.

Defenses against such an attack includes, user education and good internal host defenses. The use of a web proxy server, while it might still pass the traffic, would provide some logging capability. The use of intrusion detection which did some inspection of the protocol, might pickup on the unusual nature of the GET command contents. Finally, well managed user rights and privileges would mitigate the extent of the compromise, limiting what the attacker might be able to do, and increasing the time it would take to gain additional information for further attacks from the inside.

(NOTE: the URL for reverse www shell was reported as being r3wt.base.org in the course notes for the SANS GIAC Incident Handling and Hacker Exploits, but is no longer valid. A more thorough web search would most likely turn up a new location for this code, though, other covert channel attacks, also referred to as back channel attacks, are available. Netcat is a good example of a versatile tool and can be found at www.atstake.com/research/tools/ which has both Unix and Windows versions.)

Appendix A – Border Router Configuration

```
Current configuration:
!
! Last configuration change at 12:47:29 MST Sun Aug 25 2002
!
version 11.0
no service finger
no service pad
service password-encryption
no service udp-small-servers
no service tcp-small-servers
!
hostname GIAC-gw
!
enable secret 5 $1$X7ON$9FpBH8BudJ.bJwAxEblm61
enable password 7 19347957394C9458B837A88DD3
!
no ip bootp server
no ip source-route
no ip domain-lookup
ip tcp path-mtu-discovery
!
interface Ethernet0
 no ip address
 shutdown
!
interface Ethernet1
 ip address 192.168.2.1 255.255.255.0
 no ip redirects
 no ip unreachables
 no ip directed-broadcast
 no ip proxy-arp
 no ip mroute-cache
!
interface Serial0
 ip address 192.168.5.254 255.255.255.0
 ip access-group 101 in
 ip access-group 102 out
 no ip redirects
 no ip unreachables
 no ip directed-broadcast
 no ip proxy-arp
 no ip mroute-cache
 ntp disable
!
interface Serial1
 no ip address
 shutdown
!
interface Async1
 no ip address
 shutdown
!
no ip classless
!
```

```
ip default-network 140.222.0.0
ip route 0.0.0.0 0.0.0.0 192.168.5.254
ip route 192.168.1.0 255.255.255.0 192.168.2.2
ip route 192.168.3.0 255.255.255.0 192.168.2.2
ip route 192.168.4.0 255.255.255.0 192.168.2.2

logging 192.168.1.2
!
access-list 10 permit 192.168.3.20
!
access-list 101 deny   ip 10.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 172.16.0.0 0.15.255.255 any log
access-list 101 deny   ip 169.254.0.0 0.0.255.255 any log
access-list 101 deny   ip 127.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 0.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 240.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 224.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 192.168.1.0 0.0.0.255 any log
access-list 101 deny   ip 192.168.2.0 0.0.0.255 any log
access-list 101 deny   ip 192.168.3.0 0.0.0.255 any log
access-list 101 deny   ip 192.168.4.0 0.0.0.255 any log
access-list 101 deny   ip host 192.168.5.254 any log
access-list 101 deny   ip any any eq 1523 log
access-list 101 deny   ip any any eq 69 log
access-list 101 deny   udp any any eq 87 log
access-list 101 deny   ip any any eq 111 log
access-list 101 deny   ip any any eq 135
access-list 101 deny   ip any any eq 136
access-list 101 deny   ip any any eq 137
access-list 101 deny   ip any any eq 138
access-list 101 deny   ip any any eq 139
access-list 101 deny   ip any any eq 445
access-list 101 deny   tcp any any eq 512 log
access-list 101 deny   tcp any any eq 513 log
access-list 101 deny   tcp any any eq 514 log
access-list 101 deny   tcp any any eq 515 log
access-list 101 deny   tcp any any eq 540 log
access-list 101 deny   ip any any eq 2000 log
access-list 101 deny   ip any any eq 2049 log
access-list 101 deny   ip any any eq 6000 log
access-list 101 deny   ip any any eq 6001 log
access-list 101 deny   udp any any eq snmp log
access-list 101 deny   udp any any eq snmptrap log
access-list 101 permit ip any 192.168.1.0 0.0.0.255
access-list 101 permit ip any 192.168.2.0 0.0.0.255
access-list 101 permit ip any 192.168.3.0 0.0.0.255
access-list 101 permit ip any 192.168.4.0 0.0.0.255
access-list 101 deny   ip any any log
!
access-list 102 permit ip 192.168.1.0 0.0.0.255 any
access-list 102 permit ip 192.168.2.0 0.0.0.255 any
access-list 102 permit ip 192.168.3.0 0.0.0.255 any
access-list 102 permit ip 192.168.4.0 0.0.0.255 any
access-list 102 deny   ip any any log
!
banner motd ^C WARNING: Authorized Access Only ^C
!
```

```
line con 0
 password 7 018A44D2195430B1284
 login
line aux 0
 no exec
line vty 0 4
 access-class 10 in
 password 7 09374B62534A32472C
 login
!
ntp clock-period 17179873
ntp server 192.168.1.2 source Ethernet1
end
```

Appendix B – Iptables rc.firewall script

```
#!/bin/bash
#
# External Firewall configuration in tri-homed screened subnet architecture
#    eth2 interface leads to subnet with border router
#    eth1 interface leads to screened subnet for service network
#    eth0 interface leads to screened subnet for internal network

echo ======= Start iptables initialization =======
echo

##############################################################
##############################################################
#
# Part 1 - Variable definitions and iptables initialization
#
##############################################################
##############################################################

##############################################################
echo Part 1 - Variable definitions and iptables initialization

PRIV="0:1023"                      # well-known, privileged port range
UNPRIV="1024:65535"                # unprivileged port range
BAD_TCP_PORTS="1080,2000,2049,3128" # Problem tcp ports to automatically deny
BAD_UDP_PORTS="2049,4045"          # Problem udp ports to automatically deny

LOOPBACK="127.0.0/8"               # loopback address
CLASS_A="10.0.0.0/8"               # Class A private networks
CLASS_B="172.16.0.0/12"            # Class B private networks
CLASS_C="192.168.0.0/16"           # Class C private networks
MULTICAST="224.0.0.0/4"            # Class D multicast addresses
RESERVED_NET="240.0.0.0/5"         # Class E reserved addresses
BROADCAST="255.255.255.255"        # Broadcast address

BDRADDR="192.168.2.1"              # Border Router Gateway Address
EXTADDR="192.168.2.2"              # External Interface IP address on eth2
SVCADDR="192.168.1.1"              # Internal Interface IP address on eth1
INTADDR="192.168.3.1"              # Internal Interface IP address on eth0

EXTNIC="eth2"                      # External Interface
SVCNIC="eth1"                      # Service network Interface
INTNIC="eth0"                      # Internal network Interface

SVCNET="192.168.1.0/24"            # Service Network
SVCNET_BASE="192.168.1.0"          # Service Network
SVCNET_BROADCAST="192.168.1.255"   # Service Network
EXTNET="192.168.2.0/24"            # Internal Network
EXTNET_BASE="192.168.2.0"          # Internal Network
EXTNET_BROADCAST="192.168.2.255"   # Internal Network
INTNET="192.168.3.0/24"            # Internal Network
INTNET_BASE="192.168.3.0"          # Internal Network
INTNET_BROADCAST="192.168.3.255"   # Internal Network

SSHSERVER="192.168.3.2/32"         # Internal SSH server
```

```
SVCMAIL="192.168.1.2/32"            # Service Net Mail Server
INTMAIL="192.168.3.2/32"            # Internal Net Mail Server

DNSSERVER="192.168.5.200/32"        # DNS server
SVCDNSSERVER="192.168.1.2/32"       # DNS service net forwarding server
INTDNSSERVER="192.168.3.2/32"       # DNS internal forwarding server

SQUIDSERVER="192.168.1.4/32"        # Squid web cache proxy server
WWWSERVER="192.168.1.3/32"          # Web server

INTNTPSERVER="192.168.3.2/32"       # Internal net NTP server
SVCNTPSERVER="192.168.1.2/32"       # Service net NTP server
EXTNTPSERVER="192.168.5.5/32"       # External trusted NTP server


SVCSYSLOG="192.168.1.2/32"          # Service net syslog server

DBSERVER="192.168.3.6/32"           # Internal protected network Database server
DBPORT="1523"                       # Database listener port

SYSADM="192.168.3.123/32"           # Remote sysadm management

USER_CHAINS="INT-input                    INT-output \
                  EXT-input               EXT-output \
                  SVC-input               SVC-output \
                  ext-to-int              ext-to-svc \
                  int-to-ext              svc-to-ext \
                  int-to-svc              svc-to-int \
                  tcp-state-flags         log-tcp-state \
                  conn-track              ssh-connect \
                  ntp-query               auth-query \
                  dns-query               squid-query \
                  source-address-check    ext-source-check \
                  int-source-check        svc-source-check \
                  dest-address-check      db-query \
                  www-query               svc-mail \
                  syslog-message          \
                  icmp-in                 icmp-out"


###############################################################

# Enable ip forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# Enable broadcast echo Protection
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Disable Source Routed Packets
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

# Disabled -- must enable in kernel first --  Enable TCP SYN Cookie Protection
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

```
# Disable ICMP Redirect Acceptance
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done

# Don¹t send Redirect Messages
for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
    echo 0 > $f
done

# Drop Spoofed Packets coming in on an interface, which if replied to,
# would result in the reply going out a different interface.
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# Log packets with impossible addresses.
for f in /proc/sys/net/ipv4/conf/*/log_martians; do
    echo 1 > $f
done

###############################################################

# Remove any existing rules from all chains
iptables --flush
iptables -t nat --flush
iptables -t mangle --flush

# Activate traffic on loopback interface
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Set the default policy to drop
iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP

# Remove any pre-existing user-defined chains
iptables --delete-chain
iptables -t nat --delete-chain
iptables -t mangle --delete-chain

#Create the user-defined chains
for i in $USER_CHAINS; do
    iptables -N $i
done

###############################################################
###############################################################
#
# Part 2 - User Defined Chain rules
#
###############################################################
###############################################################
echo
echo Part 2 - User Defined Chain rules
echo
```

```
###############################################################
#
# tcp-state-flags
#
# Check for Stealth Scans and illegal TCP State Flags
echo tcp-state-flags

# All of the bits are cleared
iptables -A tcp-state-flags   -p tcp --tcp-flags ALL NONE -j log-tcp-state

# SYN and FIN are both set
iptables -A tcp-state-flags   -p tcp --tcp-flags SYN,FIN SYN,FIN -j log-tcp-state

# SYN and RST are both set
iptables -A tcp-state-flags   -p tcp --tcp-flags SYN,RST SYN,RST -j log-tcp-state

# FIN and RST are both set
iptables -A tcp-state-flags   -p tcp --tcp-flags FIN,RST FIN,RST -j log-tcp-state

# FIN is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags   -p tcp --tcp-flags ACK,FIN FIN -j log-tcp-state

# PSH is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags   -p tcp --tcp-flags ACK,PSH PSH -j log-tcp-state

# URG is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags   -p tcp --tcp-flags ACK,URG URG -j log-tcp-state


#--------------------------------------------------------------
#
# log-tcp-state
#
# Log TCP packets with bad state combinations
#
iptables -A log-tcp-state -p tcp -j LOG \
         --log-prefix "Illegal TCP state: " \
         --log-ip-options --log-tcp-options

iptables -A log-tcp-state -j DROP

###############################################################
#
# conn-track
#
# Use Connection State to By-pass Rule Checking
#
echo conn-track

iptables -A conn-track -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A conn-track -m state --state INVALID -j LOG \
         --log-prefix "INVALID packet: "
iptables -A conn-track -m state --state INVALID -j DROP

###############################################################
```

```
#
# Primary User-Defined Chains based on traffic flow
echo
echo Primary chains
echo
#
#---------------------------------------------------------------
#
# ext-to-svc
#
# Packets inbound from Internet to the service network
echo ext-to-svc

# Allow inbound queries to the web server
iptables -A ext-to-svc -p tcp -m multiport \
  --destination-port 80,443 --syn -j www-query
# Allow inbound email delivery to the email server
iptables -A ext-to-svc -p tcp --dport 25 -j svc-mail
# Allow inbound time synchronization from border router to service ntp server
iptables -A ext-to-svc -p udp --dport 123 -j ntp-query
# Allow inbound syslog message to service net syslog server
iptables -A ext-to-svc -p udp --dport 514 -j syslog-message
# Allow certain types of icmp messages inbound
iptables -A ext-to-svc -p icmp -j icmp-in

# Reject auth queries
iptables -A ext-to-svc -p tcp --dport 113 -j auth-query
# Do not allow ssh connections, log and drop them via chain ssh-connect
iptables -A ext-to-svc -p tcp --sport $UNPRIV --dport 22 -j ssh-connect

#---------------------------------------------------------------
#
# svc-to-ext
#
# Packets outbound from service network to the Internet
echo svc-to-ext

# Allow web proxy queries of "safe" well known web services
iptables -A svc-to-ext -p tcp -m multiport \
  --destination-port 21,80,443,563,591,70,210 \
  --syn --sport $UNPRIV -j squid-query
# Allow email delivery to internet email servers
iptables -A svc-to-ext -p tcp --dport 25 -j svc-mail
# Allow dns server queries to the Internet
iptables -A svc-to-ext -p udp --dport 53 -j dns-query
iptables -A svc-to-ext -p tcp --dport 53 -j dns-query
# Allow ntp client queries
iptables -A svc-to-ext -p udp --dport 123 -j ntp-query
# Allow certain icmp messages out, as needed for normal service operations
iptables -A svc-to-ext -p icmp -j icmp-out
# Allow auth queries in special cases
iptables -A svc-to-ext -p tcp --dport 113 -j auth-query

# Do not allow ssh connection out, drop and log
iptables -A svc-to-ext -p tcp --sport $UNPRIV --dport 22 -j ssh-connect

#---------------------------------------------------------------
```

```
#
# svc-to-int
#
# Packets inbound from service network to internal network
echo svc-to-int

# Allow email delivery to internal email server
iptables -A svc-to-int -p tcp --dport 25 -j svc-mail
# Allow database queries to internal database server
iptables -A svc-to-int -p tcp --dport $DBPORT -j db-query
# Allow certain types of icmp
iptables -A svc-to-int -p icmp -j icmp-in

# Reject auth queries
iptables -A svc-to-int -p tcp --dport 113 -j auth-query
# Do not allow ssh connections, log and drop them via chain ssh-connect
iptables -A svc-to-int -p tcp --sport $UNPRIV --dport 22 -j ssh-connect

#---------------------------------------------------------------
#
# int-to-svc
#
# Packets outbound from internal network to service network
echo int-to-svc

# Allow client requests to the web proxy server
iptables -A int-to-svc -p tcp --sport $UNPRIV --dport 8008 -j squid-query
# Allow ssh connections
iptables -A int-to-svc -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow email forwarding from internal email server to service net email server
iptables -A int-to-svc -p tcp --dport 25 -j svc-mail
# Allow DNS recursive queries from internal dns server to service net dns server
iptables -A int-to-svc -p udp --dport 53 -j dns-query
iptables -A int-to-svc -p tcp --dport 53 -j dns-query
# Allow NTP client queries to the service net NTP server
iptables -A int-to-svc -p udp --dport 123 -j ntp-query
# Allow certain icmp messages out, as needed for normal service operations
iptables -A int-to-svc -p icmp -j icmp-out

# Reject auth queries
iptables -A int-to-svc -p tcp --dport 113 -j auth-query

#---------------------------------------------------------------
#
# ext-to-int
#
# Packets inbound from Internet to the internal network
echo ext-to-int

# Allow incoming ssh connections
iptables -A ext-to-int -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow certain types of icmp
iptables -A ext-to-int -p icmp -j icmp-in

#---------------------------------------------------------------
#
# int-to-ext
```

```
#
# Packets only outbound from internal network to the Internet
echo int-to-ext

# Allow outbound ssh
iptables -A int-to-ext -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow certain icmp messages out, as needed for normal service operations
iptables -A int-to-ext -p icmp -j icmp-out

# Do not allow outbound queries to the Internet web servers
iptables -A int-to-ext -p tcp --dport 80 -j www-query

#---------------------------------------------------------------
#
# EXT-input
#
# Packets inbound on the external interface intended for this server
echo EXT-input

# Do not allow inbound ssh, log and drop
iptables -A EXT-input -p tcp --sport $UNPRIV --dport 22 -j ssh-connect

#---------------------------------------------------------------
#
# EXT-output
#
# Packets outbound on the external interface intended for the Internet
echo EXT-output

# Do not allow outbound ssh, log and drop
iptables -A EXT-output -p tcp --sport $UNPRIV --dport 22 -j ssh-connect

#---------------------------------------------------------------
#
# SVC-input
#
# Packets inbound on the service network interface intended for this server
echo SVC-input

# Do not allow inbound ssh, log and drop
iptables -A SVC-input -p tcp --sport $UNPRIV --dport 22 -j ssh-connect

#---------------------------------------------------------------
#
# SVC-output
#
# Packets outbound on the service network interface intended for the service net
echo SVC-output

# Allow DNS queries to service net DNS server
iptables -A SVC-output -p tcp --dport 53 -j dns-query
# Allow ntp client queries
iptables -A SVC-output -p udp --dport 123 -j ntp-query

# Do not allow outbound ssh, log and drop
iptables -A SVC-output -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
```

```
#--------------------------------------------------------------
#
# INT-input
#
# Packets inbound on the internal interface intended for this server
echo INT-input

# Allow ssh connections for remote admin
iptables -A INT-input -p tcp --sport $UNPRIV --dport 22 -j ssh-connect
# Allow certain types of icmp
iptables -A INT-input -p icmp -j icmp-in

#--------------------------------------------------------------
#
# INT-output
#
# Packets outbound on the internal interface intended for the internal net
echo INT-output

# Allow certain icmp messages out, as needed for normal service operations
iptables -A INT-output -p icmp -j icmp-out

# Do not allow outbound ssh to internal net, log and drop
iptables -A INT-output -p tcp --sport $UNPRIV --dport 22 -j ssh-connect

##############################################################
#
# Secondary user-defined chain rules
echo
echo secondary chains
echo

#--------------------------------------------------------------
# svc-mail
#
# Mail server (destination tcp port 25)
echo svc-mail

# Allow outgoing mail to the Internet from service net mail server
iptables -A svc-mail -p tcp  -s $SVCMAIL --sport $UNPRIV \
      -d ! $INTNET -m state --state NEW -j ACCEPT

# Allow incoming mail from the Internet to service net mail server
iptables -A svc-mail -p tcp -s ! $INTNET --sport $UNPRIV -d $SVCMAIL \
      -m state --state NEW -j ACCEPT

# Allow outgoing mail to the internal mail server from service net mail server
iptables -A svc-mail -p tcp  -s $SVCMAIL --sport $UNPRIV \
      -d $INTMAIL -m state --state NEW -j ACCEPT

# Allow incoming mail from the internal mail server to service net mail server
iptables -A svc-mail -p tcp -s $INTMAIL --sport $UNPRIV -d $SVCMAIL \
      -m state --state NEW -j ACCEPT

# Deny any other email connections
iptables -A svc-mail -j LOG --log-prefix "Illegal Mail connect: "
iptables -A svc-mail -j DROP
```

```
#-------------------------------------------------------------
#
# www-query
#
# Web server (tcp port 80)
echo www-query

iptables -A www-query -d $WWWSERVER \
         -m state --state NEW -j ACCEPT

# Deny any other squid connections
iptables -A www-query -j LOG --log-prefix "Illegal WWW query: "
iptables -A www-query -j DROP

#-------------------------------------------------------------
#
# squid-query
#
# Web cache proxy server (tcp port 8008)
echo squid-query

iptables -A squid-query -s $INTNET -d $SQUIDSERVER \
         -m state --state NEW -j ACCEPT

iptables -A squid-query -s $SQUIDSERVER \
         -m state --state NEW -j ACCEPT

# Deny any other squid connections
iptables -A squid-query -j LOG --log-prefix "Illegal Squid query: "
iptables -A squid-query -j DROP

#-------------------------------------------------------------
#
# dns-query
#
# dns (UDP/TCP Port 53)
echo dns-query

# Allow service net DNS server queries to an external DNS server via UDP 53
iptables -A dns-query -p udp -s $SVCDNSSERVER --sport 53 -d ! $INTNET \
       -m state --state NEW -j ACCEPT

# Allow service DNS client queries to external DNS server via UDP/TCP UNPRIV
iptables -A dns-query -p udp -s $SVCDNSSERVER --sport $UNPRIV -d ! $INTNET \
       -m state --state NEW -j ACCEPT
iptables -A dns-query -p tcp -s $SVCDNSSERVER --sport $UNPRIV -d ! $INTNET \
       -m state --state NEW -j ACCEPT

# Allow internal DNS server server queries to service net DNS server via UDP 53
iptables -A dns-query -p udp -s $INTDNSSERVER --sport 53 -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT

# Allow internal DNS client queries to service net DNS server via UDP/TCP UNPRIV
iptables -A dns-query -p udp -s $INTDNSSERVER --sport $UNPRIV -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT
iptables -A dns-query -p tcp -s $INTDNSSERVER --sport $UNPRIV -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT

# Allow firewall service interface queries to service net server via tcp or udp
iptables -A dns-query -s $SVCADDR -d $SVCDNSSERVER \
       -m state --state NEW -j ACCEPT

# Deny any other DNS connections
iptables -A dns-query -j LOG --log-prefix "Illegal DNS query: "
iptables -A dns-query -j DROP

#-------------------------------------------------------------
#
# ssh_connect
#
# ssh (TCP Port 22)
echo ssh-connect

# Allow ssh connections to service network only from internal network
iptables -A ssh-connect -s $INTNET -d $SVCNET -m state --state NEW -j ACCEPT

# Allow ssh connections initialed by internal network to any on the Internet
iptables -A ssh-connect -s $INTNET -d ! $SVCNET -m state --state NEW -j ACCEPT

# Allow ssh connections to specific internal ssh server from Internet
iptables -A ssh-connect -s ! $SVCNET -d $SSHSERVER -m state --state NEW -j ACCEPT

# Allow ssh connections to external interface of firewall for remote management
iptables -A ssh-connect -s $SYSADM -d $INTADDR -m state --state NEW -j ACCEPT

# Deny any other ssh connections
iptables -A ssh-connect -j LOG --log-prefix "Illegal ssh connect: "
iptables -A ssh-connect -j DROP

#-------------------------------------------------------------
#
# ntp_query
#
# ntp (UDP Port 123)
echo ntp-query

# Allow ntp synchronization from internal ntp server to service net ntp server
iptables -A ntp-query -s $INTNTPSERVER -d $SVCNTPSERVER -m state --state NEW -j ACCEPT

# Allow ntp synchronization from border router to service net ntp server
iptables -A ntp-query -s $BDRADDR -d $SVCNTPSERVER -m state --state NEW -j ACCEPT

# Allow ntp synchronization from service net ntp server to external ntp server
iptables -A ntp-query -s $SVCNTPSERVER -d $EXTNTPSERVER -m state --state NEW -j ACCEPT

# Deny any other ntp connections
iptables -A ntp-query -j LOG --log-prefix "Illegal ntp query: "
iptables -A ntp-query -j DROP

#-------------------------------------------------------------
#
# auth-query
#
```

```
# auth/identd (TCP Port 113)
echo auth-query

# Allow auth queries from service network email server to the Internet
iptables -A auth-query -p tcp --syn -s $SVCMAIL -d ! $INTNET \
       -m state --state NEW -j ACCEPT

# Reject, with response, any auth queries
iptables -A auth-query -p tcp --syn \
       -j REJECT --reject-with tcp-reset

# Deny any other squid connections
iptables -A auth-query -j LOG --log-prefix "Illegal auth query: "
iptables -A auth-query -j DROP

#--------------------------------------------------------------
#
# syslog-message
#
echo syslog-message

# database listener port (UDP Port 514)
iptables -A syslog-message –s $BDRADDR -d $SVCSYSLOG -j ACCEPT

# Log and Deny any other database connection connections
iptables -A syslog-message -j LOG --log-prefix "Illegal syslog message: "
iptables -A syslog-message -j DROP

#--------------------------------------------------------------
#
# db-query
echo db-query

#
# database listener port (TCP Port 1523)
iptables -A db-query -s $WWWSERVER -d $DBSERVER -m state --state NEW -j ACCEPT

# Deny any other squid connections
iptables -A db-query -j LOG --log-prefix "Illegal db query: "
iptables -A db-query -j DROP

#--------------------------------------------------------------
#
# icmp-out
#
# ICMP traffic
echo icmp-out

# Drop outgoing ICMP fragments
iptables -A icmp-out --fragment -j LOG --log-prefix "Frag outgoing ICMP: "
iptables -A icmp-out --fragment -j DROP

# Allow source quench
iptables -A icmp-out -p icmp --icmp-type source-quench -j ACCEPT

# Allow parameter-problem
iptables -A icmp-out -p icmp --icmp-type parameter-problem -j ACCEPT
```

```
# Allow fragmentation needed
iptables -A icmp-out -p icmp --icmp-type fragmentation-needed -j ACCEPT

# Allow outgoing ping from internal network
iptables -A icmp-out -p icmp --icmp-type echo-request \
      -s $SYSADM -m state --state NEW -j ACCEPT

#Drop all other incoming ICMP traffic
iptables -A icmp-out -p icmp -j LOG --log-prefix "Illegal outgoing ICMP"
iptables -A icmp-out -p icmp -j DROP

#-------------------------------------------------------------
#
# icmp-in
#
# ICMP traffic
echo icmp-in

# Drop ICMP fragments
iptables -A icmp-in --fragment -j LOG --log-prefix "Frag ICMP: "
iptables -A icmp-in --fragment -j DROP

# Allow destination unreachable
iptables -A icmp-in -p icmp --icmp-type destination-unreachable -j ACCEPT

# Allow parameter-problem
iptables -A icmp-in -p icmp --icmp-type parameter-problem -j ACCEPT

# Allow source quench
iptables -A icmp-in -p icmp --icmp-type source-quench -j ACCEPT

#Drop all other incoming ICMP traffic
iptables -A icmp-in -p icmp -j LOG --log-prefix "Illegal incoming ICMP"
iptables -A icmp-in -p icmp -j DROP

#-------------------------------------------------------------
#
# external interface source-address-check
#
echo ext-source-check

# Check for source address spoofing
iptables -A ext-source-check -s $EXTADDR -j DROP
iptables -A ext-source-check -s $INTNET -j DROP
iptables -A ext-source-check -s $SVCNET -j DROP
iptables -A ext-source-check -j source-address-check

#-------------------------------------------------------------
#
# service interface source-address-check
#
echo svc-source-check

# Check for source address spoofing
iptables -A svc-source-check -s $SVCADDR -j DROP
iptables -A svc-source-check -s $INTNET -j DROP
iptables -A svc-source-check -j source-address-check

#--------------------------------------------------------------
#
# internal interface source-address-check
#
echo int-source-check

# Check for source address spoofing
iptables -A int-source-check -s $INTADDR -j DROP
iptables -A int-source-check -s $SVCNET -j DROP
iptables -A int-source-check -j source-address-check

#--------------------------------------------------------------
#
# source-address-check
#
# Check for source address spoofing or private addresses
echo source-address-check

iptables -A source-address-check -s $CLASS_A -j DROP
iptables -A source-address-check -s $CLASS_B -j DROP
#iptables -A source-address-check -s $CLASS_C -j DROP
iptables -A source-address-check -s $MULTICAST -j DROP
iptables -A source-address-check -s $RESERVED_NET -j DROP
iptables -A source-address-check -s $LOOPBACK -j DROP

iptables -A source-address-check -d $BROADCAST -j DROP
iptables -A source-address-check -s 0.0.0.0/8 -j DROP

iptables -A source-address-check -s 169.254.0.0/16 -j DROP
iptables -A source-address-check -s 192.0.2.0/24 -j DROP

#--------------------------------------------------------------
#
# dest-address-check
#
# Check destination address
echo dest-address-check

# Block broadcasts
iptables -A dest-address-check -d $SVCNET_BASE -j DROP
iptables -A dest-address-check -d $SVCNET_BROADCAST -j DROP
iptables -A dest-address-check -d $INTNET_BASE -j DROP
iptables -A dest-address-check -d $INTNET_BROADCAST -j DROP
iptables -A dest-address-check -d $EXTNET_BASE -j DROP
iptables -A dest-address-check -d $EXTNET_BROADCAST -j DROP

iptables -A dest-address-check -p ! udp -d $MULTICAST -j DROP

# Deny specific problem ports

iptables -A dest-address-check -p tcp -m multiport \
      --destination-port $BAD_TCP_PORTS --syn -j DROP

iptables -A dest-address-check -p udp -m multiport \
      --destination-port $BAD_UDP_PORTS -j DROP
```

```
##############################################################
##############################################################
#
# Part 3 - Start of filter processing
#
##############################################################
##############################################################
echo
echo Part 3 - Start of filter processing

##############################################################
#
# Initial checks common to all traffic
#
echo
echo Initialize built-in chains
echo

# If TCP: Check for stealth scans or illegal flag combinations
iptables -A INPUT -p tcp -j tcp-state-flags
iptables -A OUTPUT -p tcp -j tcp-state-flags
iptables -A FORWARD -p tcp -j tcp-state-flags

# Check for established connections
iptables -A INPUT -j conn-track
iptables -A OUTPUT -j conn-track
iptables -A FORWARD -j conn-track

# Test for illegal source and destination addresses in incoming packets
iptables -A INPUT -p ! tcp -j source-address-check
iptables -A INPUT -p tcp --syn -j source-address-check
iptables -A OUTPUT -p ! tcp -j source-address-check
iptables -A OUTPUT -p tcp --syn -j source-address-check
iptables -A FORWARD -i $EXTNIC -o $INTNIC -p ! tcp -j ext-source-check
iptables -A FORWARD -i $EXTNIC -o $INTNIC -p tcp --syn -j ext-source-check
iptables -A FORWARD -i $EXTNIC -o $SVCNIC -p ! tcp -j ext-source-check
iptables -A FORWARD -i $EXTNIC -o $SVCNIC -p tcp --syn -j ext-source-check
iptables -A FORWARD -i $SVCNIC -o $EXTNIC -p ! tcp -j svc-source-check
iptables -A FORWARD -i $SVCNIC -o $EXTNIC -p tcp --syn -j svc-source-check
iptables -A FORWARD -i $SVCNIC -o $INTNIC -p ! tcp -j svc-source-check
iptables -A FORWARD -i $SVCNIC -o $INTNIC -p tcp --syn -j svc-source-check
iptables -A FORWARD -i $INTNIC -o $EXTNIC -p ! tcp -j int-source-check
iptables -A FORWARD -i $INTNIC -o $EXTNIC -p tcp --syn -j int-source-check
iptables -A FORWARD -i $INTNIC -o $SVCNIC -p ! tcp -j int-source-check
iptables -A FORWARD -i $INTNIC -o $SVCNIC -p tcp --syn -j int-source-check

#
iptables -A INPUT -j dest-address-check
iptables -A OUTPUT -j dest-address-check
iptables -A FORWARD -j dest-address-check

##############################################################
#
# Traffic filtered by user-defined chains based on traffic flow
#
```

```
# Standard Checks for incoming packets
iptables -A FORWARD -i $EXTNIC -o $SVCNIC -j ext-to-svc
iptables -A FORWARD -i $SVCNIC -o $INTNIC -j svc-to-int
iptables -A FORWARD -i $EXTNIC -o $INTNIC -j ext-to-int
iptables -A INPUT -i $EXTNIC -d $EXTADDR -j EXT-input
iptables -A INPUT -i $SVCNIC -d $SVCADDR -j SVC-input
iptables -A INPUT -i $INTNIC -d $INTADDR -j INT-input

# Standard Checks for outgoing packets
iptables -A FORWARD -i $SVCNIC -o $EXTNIC -j svc-to-ext
iptables -A FORWARD -i $INTNIC -o $SVCNIC -j int-to-svc
iptables -A FORWARD -i $INTNIC -o $EXTNIC -j int-to-ext
iptables -A OUTPUT -o $EXTNIC -s $EXTADDR -j EXT-output
iptables -A OUTPUT -o $SVCNIC -s $SVCADDR -j SVC-output
iptables -A OUTPUT -o $INTNIC -s $INTADDR -j INT-output

###############################################################
# Specific drops of known traffic that you don't want to include in the log file
# Internal NetBIOS broadcasts and telnet scans
iptables -A INPUT -i $INTNIC -p tcp --dport 139 -j DROP
iptables -A FORWARD -p tcp --dport 23 -j DROP
iptables -A INPUT -p tcp --dport 23 -j DROP

# Log anything else that defaults to drop
iptables -A INPUT -j LOG --log-prefix "Default drop: "
iptables -A INPUT -j DROP
iptables -A OUTPUT -j LOG --log-prefix "Default drop: "
iptables -A OUTPUT -j DROP
iptables -A FORWARD -j LOG --log-prefix "Default drop: "
iptables -A FORWARD -j DROP

###############################################################
###############################################################

echo ======= End of initialization ================

exit 0
```

Appendix C – iptables list output

```
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
tcp-state-flags  tcp  --  anywhere             anywhere
conn-track  all  --  anywhere             anywhere
source-address-check !tcp  --  anywhere             anywhere
source-address-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
dest-address-check  all  --  anywhere             anywhere
EXT-input  all  --  anywhere             192.168.2.2
SVC-input  all  --  anywhere             192.168.1.1
INT-input  all  --  anywhere             192.168.3.1
DROP       tcp  --  anywhere             anywhere           tcp dpt:netbios-ssn
DROP       tcp  --  anywhere             anywhere           tcp dpt:telnet
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Default drop: '
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
tcp-state-flags  tcp  --  anywhere             anywhere
conn-track  all  --  anywhere             anywhere
ext-source-check !tcp  --  anywhere             anywhere
ext-source-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
ext-source-check !tcp  --  anywhere             anywhere
ext-source-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
svc-source-check !tcp  --  anywhere             anywhere
svc-source-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
svc-source-check !tcp  --  anywhere             anywhere
svc-source-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
int-source-check !tcp  --  anywhere             anywhere
int-source-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
int-source-check !tcp  --  anywhere             anywhere
int-source-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
dest-address-check  all  --  anywhere             anywhere
ext-to-svc  all  --  anywhere             anywhere
svc-to-int  all  --  anywhere             anywhere
ext-to-int  all  --  anywhere             anywhere
svc-to-ext  all  --  anywhere             anywhere
int-to-svc  all  --  anywhere             anywhere
int-to-ext  all  --  anywhere             anywhere
DROP       tcp  --  anywhere             anywhere           tcp dpt:telnet
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Default drop: '
DROP       all  --  anywhere             anywhere

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
tcp-state-flags  tcp  --  anywhere             anywhere
conn-track  all  --  anywhere             anywhere
source-address-check !tcp  --  anywhere             anywhere
source-address-check  tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN
dest-address-check  all  --  anywhere             anywhere
EXT-output  all  --  192.168.2.2          anywhere
SVC-output  all  --  192.168.1.1          anywhere
INT-output  all  --  192.168.3.1          anywhere
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Default drop: '
DROP       all  --  anywhere             anywhere

Chain EXT-input (1 references)
target     prot opt source               destination
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain EXT-output (1 references)
target     prot opt source               destination
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain INT-input (1 references)
target     prot opt source               destination
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh
```

```
icmp-in    icmp --  anywhere             anywhere

Chain INT-output (1 references)
target     prot opt source               destination
icmp-out   icmp --  anywhere             anywhere
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain SVC-input (1 references)
target     prot opt source               destination
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain SVC-output (1 references)
target     prot opt source               destination
dns-query  udp  --  anywhere             anywhere           udp dpt:domain
dns-query  tcp  --  anywhere             anywhere           tcp dpt:domain
ntp-query  udp  --  anywhere             anywhere           udp dpt:ntp
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain auth-query (4 references)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.1.2         !192.168.3.0/24     tcp flags:SYN,RST,ACK/SYN state NEW
REJECT     tcp  --  anywhere             anywhere           tcp flags:SYN,RST,ACK/SYN reject-with tcp-reset
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal auth query: '
DROP       all  --  anywhere             anywhere

Chain conn-track (3 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere           state RELATED,ESTABLISHED
LOG        all  --  anywhere             anywhere           state INVALID LOG level warning prefix `INVALID packet: '
DROP       all  --  anywhere             anywhere           state INVALID

Chain db-query (1 references)
target     prot opt source               destination
ACCEPT     all  --  192.168.1.2          192.168.3.6        state NEW
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal db query: '
DROP       all  --  anywhere             anywhere

Chain dest-address-check (3 references)
target     prot opt source               destination
DROP       all  --  anywhere             192.168.1.0
DROP       all  --  anywhere             192.168.1.255
DROP       all  --  anywhere             192.168.3.0
DROP       all  --  anywhere             192.168.3.255
DROP       all  --  anywhere             192.168.2.0
DROP       all  --  anywhere             192.168.2.255
DROP      !udp  --  anywhere             BASE-ADDRESS.MCAST.NET/4
DROP       tcp  --  anywhere             anywhere           multiport dports socks,2000,nfs,squid tcp flags:SYN,RST,ACK/SYN
DROP       udp  --  anywhere             anywhere           multiport dports nfs,4045

Chain dns-query (6 references)
target     prot opt source               destination
ACCEPT     udp  --  192.168.1.2         !192.168.3.0/24     udp spt:domain state NEW
ACCEPT     udp  --  192.168.1.2         !192.168.3.0/24     udp spts:1024:65535 state NEW
ACCEPT     tcp  --  192.168.1.2         !192.168.3.0/24     tcp spts:1024:65535 state NEW
ACCEPT     udp  --  192.168.3.2          192.168.1.2        udp spt:domain state NEW
ACCEPT     udp  --  192.168.3.2          192.168.1.2        udp spts:1024:65535 state NEW
ACCEPT     tcp  --  192.168.3.2          192.168.1.2        tcp spts:1024:65535 state NEW
ACCEPT     udp  --  192.168.1.1          192.168.1.2        udp spts:1024:65535 state NEW
ACCEPT     tcp  --  192.168.1.1          192.168.1.2        tcp spts:1024:65535 state NEW
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal DNS query: '
DROP       all  --  anywhere             anywhere

Chain ext-source-check (4 references)
target     prot opt source               destination
DROP       all  --  192.168.2.2          anywhere
DROP       all  --  192.168.3.0/24       anywhere
DROP       all  --  192.168.1.0/24       anywhere
source-address-check  all  --  anywhere             anywhere

Chain ext-to-int (1 references)
target     prot opt source               destination
```

```
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh
icmp-in    icmp --  anywhere             anywhere

Chain ext-to-svc (1 references)
target     prot opt source               destination
www-query  tcp  --  anywhere             anywhere           multiport dports http,https tcp flags:SYN,RST,ACK/SYN
svc-mail   tcp  --  anywhere             anywhere           tcp dpt:smtp
ntp-query  udp  --  anywhere             anywhere           udp dpt:ntp
syslog-message  udp  --  anywhere             anywhere           udp dpt:syslog
icmp-in    icmp --  anywhere             anywhere
auth-query  tcp  --  anywhere             anywhere           tcp dpt:auth
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain icmp-in (4 references)
target     prot opt source               destination
LOG        all  -f  anywhere             anywhere           LOG level warning prefix `Frag ICMP: '
DROP       all  -f  anywhere             anywhere
ACCEPT     icmp --  anywhere             anywhere           icmp destination-unreachable
ACCEPT     icmp --  anywhere             anywhere           icmp parameter-problem
ACCEPT     icmp --  anywhere             anywhere           icmp source-quench
LOG        icmp --  anywhere             anywhere           LOG level warning prefix `Illegal incoming ICMP'
DROP       icmp --  anywhere             anywhere

Chain icmp-out (4 references)
target     prot opt source               destination
LOG        all  -f  anywhere             anywhere           LOG level warning prefix `Frag outgoing ICMP: '
DROP       all  -f  anywhere             anywhere
ACCEPT     icmp --  anywhere             anywhere           icmp source-quench
ACCEPT     icmp --  anywhere             anywhere           icmp parameter-problem
ACCEPT     icmp --  anywhere             anywhere           icmp fragmentation-needed
ACCEPT     icmp --  dactylis.nrel.colostate.edu  anywhere           icmp echo-request state NEW
LOG        icmp --  anywhere             anywhere           LOG level warning prefix `Illegal outgoing ICMP'
DROP       icmp --  anywhere             anywhere

Chain int-source-check (4 references)
target     prot opt source               destination
DROP       all  --  192.168.3.1          anywhere
DROP       all  --  192.168.1.0/24       anywhere
source-address-check  all  --  anywhere             anywhere

Chain int-to-ext (1 references)
target     prot opt source               destination
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh
icmp-out   icmp --  anywhere             anywhere
www-query  tcp  --  anywhere             anywhere           tcp dpt:http

Chain int-to-svc (1 references)
target     prot opt source               destination
squid-query  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:http-alt
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh
svc-mail   tcp  --  anywhere             anywhere           tcp dpt:smtp
dns-query  udp  --  anywhere             anywhere           udp dpt:domain
dns-query  tcp  --  anywhere             anywhere           tcp dpt:domain
ntp-query  udp  --  anywhere             anywhere           udp dpt:ntp
icmp-out   icmp --  anywhere             anywhere
auth-query  tcp  --  anywhere             anywhere           tcp dpt:auth

Chain log-tcp-state (7 references)
target     prot opt source               destination
LOG        tcp  --  anywhere             anywhere           LOG level warning tcp-options ip-options prefix `Illegal TCP state
DROP       all  --  anywhere             anywhere

Chain ntp-query (4 references)
target     prot opt source               destination
ACCEPT     all  --  192.168.3.2          192.168.1.2        state NEW
ACCEPT     all  --  192.168.2.1          192.168.1.2        state NEW
ACCEPT     all  --  192.168.1.2          dactylis.nrel.colostate.edustate NEW
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal ntp query: '
DROP       all  --  anywhere             anywhere

Chain source-address-check (7 references)
```

```
target     prot opt source               destination
DROP       all  --  10.0.0.0/8           anywhere
DROP       all  --  172.16.0.0/12        anywhere
DROP       all  --  BASE-ADDRESS.MCAST.NET/4  anywhere
DROP       all  --  240.0.0.0/5          anywhere
DROP       all  --  127.0.0.0/8          anywhere
DROP       all  --  anywhere             255.255.255.255
DROP       all  --  0.0.0.0/8            anywhere
DROP       all  --  169.254.0.0/16       anywhere
DROP       all  --  192.0.2.0/24         anywhere

Chain squid-query (2 references)
target     prot opt source               destination
ACCEPT     all  --  192.168.3.0/24       192.168.1.2        state NEW
ACCEPT     all  --  192.168.1.3          anywhere           state NEW
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal Squid query: '
DROP       all  --  anywhere             anywhere

Chain ssh-connect (12 references)
target     prot opt source               destination
ACCEPT     all  --  192.168.3.0/24       192.168.1.0/24     state NEW
ACCEPT     all  --  192.168.3.0/24      !192.168.1.0/24     state NEW
ACCEPT     all  -- !192.168.1.0/24       192.168.3.30        state NEW
ACCEPT     all  --  dactylis.nrel.colostate.edu  192.168.3.1        state NEW
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal ssh connect: '
DROP       all  --  anywhere             anywhere

Chain svc-mail (4 references)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.1.2         !192.168.3.0/24     tcp spts:1024:65535 state NEW
ACCEPT     tcp  -- !192.168.3.0/24       192.168.1.2        tcp spts:1024:65535 state NEW
ACCEPT     tcp  --  192.168.1.2          192.168.3.2        tcp spts:1024:65535 state NEW
ACCEPT     tcp  --  192.168.3.2          192.168.1.2        tcp spts:1024:65535 state NEW
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal Mail connect: '
DROP       all  --  anywhere             anywhere

Chain svc-source-check (4 references)
target     prot opt source               destination
DROP       all  --  192.168.1.1          anywhere
DROP       all  --  192.168.3.0/24       anywhere
source-address-check  all  --  anywhere             anywhere

Chain svc-to-ext (1 references)
target     prot opt source               destination
squid-query  tcp  --  anywhere             anywhere           multiport dports ftp,http,https,nntps,591,gopher,z39.50 tcp
      spts:1024:65535 flags:SYN,RST,ACK/SYN
svc-mail   tcp  --  anywhere             anywhere           tcp dpt:smtp
dns-query  udp  --  anywhere             anywhere           udp dpt:domain
dns-query  tcp  --  anywhere             anywhere           tcp dpt:domain
ntp-query  udp  --  anywhere             anywhere           udp dpt:ntp
icmp-out   icmp --  anywhere             anywhere
auth-query  tcp  --  anywhere             anywhere           tcp dpt:auth
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain svc-to-int (1 references)
target     prot opt source               destination
svc-mail   tcp  --  anywhere             anywhere           tcp dpt:smtp
db-query   tcp  --  anywhere             anywhere           tcp dpt:1523
icmp-in    icmp --  anywhere             anywhere
auth-query  tcp  --  anywhere             anywhere           tcp dpt:auth
ssh-connect  tcp  --  anywhere             anywhere           tcp spts:1024:65535 dpt:ssh

Chain syslog-message (1 references)
target     prot opt source               destination
ACCEPT     all  --  192.168.2.1          192.168.1.2
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal syslog message: '
DROP       all  --  anywhere             anywhere

Chain tcp-state-flags (3 references)
target     prot opt source               destination
log-tcp-state  tcp  --  anywhere             anywhere           tcp flags:FIN,SYN,RST,PSH,ACK,URG/NONE
```

```
log-tcp-state  tcp  --  anywhere             anywhere           tcp flags:FIN,SYN/FIN,SYN
log-tcp-state  tcp  --  anywhere             anywhere           tcp flags:SYN,RST/SYN,RST
log-tcp-state  tcp  --  anywhere             anywhere           tcp flags:FIN,RST/FIN,RST
log-tcp-state  tcp  --  anywhere             anywhere           tcp flags:FIN,ACK/FIN
log-tcp-state  tcp  --  anywhere             anywhere           tcp flags:PSH,ACK/PSH
log-tcp-state  tcp  --  anywhere             anywhere           tcp flags:ACK,URG/URG

Chain www-query (2 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             192.168.1.3        state NEW
LOG        all  --  anywhere             anywhere           LOG level warning prefix `Illegal WWW query: '
DROP       all  --  anywhere             anywhere
```

Cited References:

(1) Zovi, Dino Dai. "Kernel Root Kits and Loadable Kernel Modules." 2001.
http://rr.sans.org/threats/rootkits.php (August 28, 2002)

(2) Swab, Kevin. "SMTP Gateway Virus Filtering with Sendmail and AmaViS." 2001.
http://rr.sans.org/email/amavis.php (August 28, 2002)

(3) Unknown. "SSL/TLS Strong Encryption: How-To" 2002.
http://httpd.apache.org/docs-2.0/ssl/ssl_howto.html (August 28, 2002)

(4) Unknown. "Security Overview" 2002.
http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scoverv.htm (August 28,2002)

(5) Antoine, Vanessa, .et .al "Router Security Configuration Guide" 2002.
http://nsa1.www.conxion.com/cisco/guides/cis-2.pdf (August 28,2002)

(6) Ballew, Scott M. *Managing IP Networks with Cisco Routers.*
Sebastopol, CA: O'Reilly and Associates, 1997.

(7) Unknown. "Packet Filtering for Firewall Systems" 2002.
http://www.cert.ort/tech_tips/packet_filtering.html (August 28, 2002)

(8) Ziegler, Robert L and Carl B. Constantine.. *Linux Firewalls, Second Edition.* Indianapolis, IN: New Riders, 2002.

(9) Andreasson, Oskar. "IPTables Tutorial" 2001.
http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.html#MATCHES
(August 28, 2002)

(10) Zwickey, Cooper and Chapman. *Building internet Firewalls. Second Edition.* Sebastopol, CA: O'Reilly and Associates, 2000.

(11) McClure, Scambray and Kurtz. *Hacking Exposed. Third Edition*
Berkley, CA: McGraw Hill, 2001.

(12) Northcutt, Stephen and Judy Novak. *Network Intrusion Detection, An Analyst's Handbook, Second Edition*. Indianapolis, IN: New Riders, 2000.

(14) Albitz, Paul and Cricket Liu. *DNS and BIND, 3rd Edition.*
Sebastopol, CA: O'Reilly and Associates, 1998

(15) Garfinkel, Simson and Gene Spafford. *Practical Unix and Internet Security 2nd Edition.*
Sebastopol, CA: O'Reilly and Associates, 1996

(16) Unknown. *SANS GIAC Course Notes: Firewall and Perimeter Protection 2.1.5*

(17) Lowe Kwan. “Kernel Rebuild Procedure” 2002.
http://www.digitalhermit.com/linux/kernel.html (August 28th, 2002)

(18) Unknown. *SANS GIAC Course Notes: Incident Response and Hacker Exploits*

(19) Smith, Christian J. “Covert Shells” 2000. http://rr.sans.org/covertchannels/covert_shells.php
(August 28th, 2002)