SECURING OPEN SOURCE VIRTUAL PRIVATE NETWORKS: A STUDY IN LINUX SECURITY

By

WILLIAM VALELLA

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTERS OF SCIENCE

UNIVERSITY OF FLORIDA

2001

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

SECURING OPEN SOURCE VIRTUAL PRIVATE NETWORKS:
A STUDY IN LINUX SECURITY

By

William Valella

December 2001

Chairman: Dr. Manuel Bermudez
Major Department: Computer and Information Science and Engineering

Small businesses continue to search for ways to save money while increasing productivity. Finding a solution to this oxymoron may appear to be beyond the grasp of most. However, with the growing open-source community and the rapid adoption of the Linux operating system it is possible to save money and increase productivity. Linux is a free operating system and is widely available both in retail stores and on the Internet. As a result, a great number of programmers and researchers have begun developing, maintaining, and freely distributing many of the same types of programs previously only available at high cost.

By using these "freeware," open source programs, small businesses can now connect branch offices and remote users to the home-office network. This thesis introduces many of the open-source tools available for developing and securing Virtual Private Networks that can provide low-cost connectivity for small businesses. The thesis focuses on creating a production system using low-cost PC hardware and freely

distributed software. In addition, it provides an in-depth discussion of the security implications of remote connectivity and guides the reader through the steps involved in addressing each security issue. The thesis also reports on the experience gained while deploying a production Linux Virtual Private Network solution and the experience of maintaining such a system.

CHAPTER 1
INTRODUCTION

Consider a time when access to a file left in a directory on a corporate network would prove invaluable at another location. The company denies access to the corporate network from anywhere other than the physical location of the network—from a workstation directly connected to the network. Using an ftp client to get the file from the corporate ftp server would make life much easier. However, the file contains sensitive information and, according to corporate policy, cannot reside on a server providing publicly accessible services such as ftp. Opening a telnet connection to the corporate network is no longer an option since telnet is inherently an insecure protocol.

In addition, it is not economical for the company to hardwire everyone into the trusted network. The manpower does not exist to deploy such a hardwired network, and even if it did, the maintenance alone would be prohibitively expensive. Outsourcing the connection to a third party would take care of the deployment and maintenance, but the financial resources are normally insufficient to foot the bill for it. That does not leave much in the way of options for corporate connectivity. Can the company afford not to offer remote connectivity without sacrificing productivity? How can the company use a low-cost, high-speed Internet connection such as a DSL or Cable modem to provide an encryption/authentication tunnel into the corporate trusted network? An option must exist to provide remote connectivity for small businesses and its branch offices.

Virtual Private Networks

A Virtual Private Network (VPN) can provide a low-cost, low-maintenance alternative for telecommuters and branch offices of corporate entities. As a result, companies can consider a potential productivity increase with all users having access to the corporate LAN from just about anywhere. A Virtual Private Network is a set of private networks using secure communications across a public infrastructure like the Internet. With the current availability of highly secure authentication and encryption schemes it is possible to implement a tunnel protected by encryption and authentication that passes packets from one trusted network to another. In fact, many vendors such as Cisco, Nokia, and Watchguard Technologies provide such services through hardware connected to any dedicated Internet connection.

The precursor to the VPN can be called the private data network – a secure private medium using dedicated local office wiring and dedicated leased circuits to connect remote sites or branch offices; all at an extremely high cost. The associated cost involves the client now managing the network and all its associated elements, investing capital in network-switching infrastructure to pass and route packets to the appropriate subnets and bridges, hiring trained and experienced staff, and assuming complete responsibility for the provisioning and on-going maintenance of the network service (Ferguson & Huston, 1998).

The base motivation for not employing a private data network is the economics of communication. Rhetorically, the question is just what is data worth and how much is anyone willing to invest in its protection. Many small-to-medium businesses do not have security policies or a standard means for evaluating their own network security.

> "Often, the business of engineering a product obscures the need to protect the design from the unknown or unexpected Peeping-Tom. Further, their capital is first and foremost used in engineering and marketing. Commonly if a corporation does not have a significant web-presence, the thought of network and Internet security is bottom of the list. As small companies find it more economical to have branch offices rather than expand locally, connecting remote sites is essential to sharing data. Most companies, however, are not willing to invest the capital in a security employee or in a managed connectivity solution. So what is the answer? Provide a secure, easy to manage, low-cost VPN implementation that does not require a large capital investment or additional staff." (Chae, 1998)

By far the biggest selling point for Virtual Private Networks is the associated cost savings. Using the Internet to distribute network services over long distances means avoiding having to purchase expensive leased lines to branch offices or partner companies. In addition, paying for long-distance charges on dial-up modem or ISDN calls between distant sites becomes a thing of the past. Finally, implementing a VPN solution means not having to invest in additional wide-area network (WAN) equipment and instead leveraging an existing Internet installation."

InternetWeek research on VPN usage shows that of 29% 200 IT managers currently use a VPN solution and 55% plan to implement one in 6 months to a year. Sixty-eight percent of IT mangers plan to use VPN authentication and access control on extranet applications. Remote site connections were at the top of the usage list with 41% intending to connect at least one remote site and 67% between one and ten sites. Ninety percent of those surveyed felt Security was the most important requirement of VPN implementations, with ease of management or tools provided a close second. Fifty-two percent of managers intend to implement their own VPN solution rather than purchase a bundled service such as a managed VPN from an Internet Service Provider (InternetWeekOnline, n.d.).

According to Packet Magazine, IP is seemingly the corporate standard for data communications across local and wide-area networks, and intranets have become a critical means for sustaining and increasing productivity and decreasing time to market, which in turn places unprecedented demands on WAN technologies to ensure quality of service and carry advanced network services such as security. With such importance placed on intra-network communications, the need for cost-effective options for connecting users over distributed networks steps forward as a problem needing immediate attention. Managers want a way to quickly and economically set up a secure intranet using leased lines and the "insecure" Internet.

Virtual private networks can provide "secure" connection using encrypted connections between users' distributed sites. More importantly, the VPNs forward IP traffic so that in most cases they can provide a secure path for network applications that requires no changes to the actual application itself to support multiple platforms.

> "The principle motivation in establishing a VPN of this type is that perhaps the majority of communications between devices within the VPN community may be sensitive in nature…, yet the total value of the communications system does not justify the investment in a fully private communications system which uses discrete transmission elements." (Ferguson & Huston, 1998)

Perhaps the most pervasive type of VPN is one in which distributed subnetworks—whether separated geographically or administratively—falling under a common administrative or corporate domain must be interconnected using a shared domain that is outside of administrative control.

Virtual Private Network Design Issues

A high-quality Virtual Private Network needs to support multiple OS platforms, remain open-sourced but with roots firmly planted in a supported standard, provide IP

fragmentation support, strong security, a bastion firewall, key management, policy management, and authentication, and be scalable across a multitude of conditions. In addition, these conditions must come at a cost most small businesses can manage.

Further, VPN implementations must take into account that the service provided operates through a medium considered a mutually hostile environment. In such an environment as the Internet, any vulnerability exposing the trusted partners to access by third parties will, in most cases, be exploited in a hostile fashion. A Virtual Private Network rarely exists as a completely isolated communication network. Each VPN has some external connection that allows controlled connectivity with other VPNs and the broader untrusted community. As a result, the greatest consideration in the design of a virtual private network involves the tradeoff between secure privacy and the need for external access.

Virtual Private Network Implementations

In past implementations of virtual private networking, attempts were made to provide secure communications through the various operating system layers. Securing the application layer meant providing secure communications on a program-by-program basis. On the other side, providing secure communications through the link layer proved extremely cost prohibitive. Using the network layer often proved successful but at the cost of interoperability across platforms. Of the four layer implementations, the transport layer proved the most versatile and remains at the forefront of VPN technology (Ferguson & Huston, 1998). What follows is a summary description of current and past implementations categorized by layer.

Application Layer

Privacy can be provided based on an application-by-application basis using cryptoAPIs such as the Generic Security Service API. The GSS API is an application-programming interface that provides security services to callers in a generic fashion. It is supportable by a range of underlying mechanisms and technologies, allowing source level portability of applications to different environments. CryptoAPI provides an abstraction layer that isolates the application from the algorithm used to protect the data. The application will refer to contexts and keys and make calls to special functions listed in the API that act as drivers for the encryption servers installed on the machine. These servers are Cryptographic Service Providers (CSP) and are the modules that do the actual work of encoding and decoding data. As a result, an application can be developed incorporating the CryptoAPI and can provide secure communications on an application-by-application basis.

Cryptographic Service Providers are not the only means for providing application-based secure communications. Protocols such as Pretty Good Privacy (PGP) can provide end-to-end encryption and authentication for files and email. PGP utilizes certificate and key registrations that offer the user the option to encrypt their own data and publish a public key to the global community. In addition, the user can register the public key with an authorization authority so other users can verify the validity of the key. Other examples of application-layer security include Domain Name System Security (DNSSec) that allows secure zone transfers between different administrative name servers; Secure remote access provided by using secure shell protocols such as SSH or Stanford's Secure Remote Password.

Link Layer

Link layer secure communications typically involve the use of Asynchronous Transfer Mode (ATM), a network technology based on transferring data in cells or packets of a fixed size, and Frame Relay Virtual Connections. These technologies can provide Private Virtual Circuits that allow discrete VPNs to be constructed across a single Frame Relay network. However, such implementations are often cost prohibitive and complex to implement and manage. Multi Protocol Over ATM (MPOA) is one example of such technologies. MPOA uses standardized ATM switching to provide high performance, scalable routing functionality to restrict and grant access to destinations on managed networks and from the Internet. This implementation is very reliable if constructed and managed well, however, because of its sole reliance on ATM means it can be difficult and costly to implement.

Network Layer

At the network layer, secure communications typically utilize the operating system's routing implementations and services to encrypt and then transfer secured information to other similarly implemented operating systems. The virtual private network daemon (Vpnd) connects two networks on a network level either via TCP/IP or a leased line attached to a serial interface. All data between the two networks are encrypted using the un-patented free Blowfish encryption algorithm. Blowfish is a symmetric block cipher that takes a variable-length key, from 32 bits to 448 bits. The random key length makes it ideal for both domestic and exportable use since government restrictions allow only a certain level of encryption for export – typically 56 bits or less. Bruce Schneier designed Blowfish in 1993 as a fast, free alternative to existing encryption algorithms such as DES and IDEA. However, vpnd is not intended as a replacement for existing

secured communications software like ssh or tunneling facilities of the operating system. Rather, it is intended as a means of securing transparent network interconnection across potentially insecure channels. Such a network layer implementation would require further application or transport layer securities.

Border Gateway Protocol (BGP) is an inter-autonomous system routing protocol designed for TCP/IP Internets. BGP supports transit policies via controlled distribution of routing information. In plain terms, it is a method to control route propagation. The use of the BGP communities attribute allows a VPN provider to attach a community attribute to the BGP Network Layer Reachability Information. In this way, configuration control allows route information propagation in strict accordance with community profiles. Subscribers to this type of VPN cannot detect the presence of other subscribers even though multiple interwoven streams of subscriber data traffic pass unprotected in the core of the service provider's network

Transport Layer

Netscape's Secure Sockets Layer (SSL) can allow a client and server to authenticate each other and negotiate encryption and keys before an application transmits data. Thus, it can provide secure communication for any application that can interact with the transport layer calls for SSL. Transport Security Layer (TSL) is the next generation of SSL 3.0.

VPN tunneling involves data packets that are first encrypted for security, and then encapsulated in an IP package by the VPN and tunneled through the Internet. Such implementations include L2TP, SOCKS 5, and IPSEC and can provide tunneling functions that can be widely used as the building blocks for VPN security.

Layer-2 Tunneling Protocol (L2TP) is a combination of Microsoft's PPTP and Cisco's Layer-2 Forwarding (L2F). L2TP supports any routed protocol and any WAN backbone technology (frame relay, ATM, x.25 and SONET). Further information on L2TP is available in RFC2661.

SOCKS5, also known as Authenticated Firewall Traversal, creates network proxies at the session layer. It is a protocol for handling TCP traffic through a proxy server that can be used with virtually any TCP application, including Web browsers and FTP clients. Further it provides a simple firewall and IP masquerading by checking and filtering incoming and outgoing packets and hides the IP addresses of client applications. It is a mechanism where a secure proxy data channel can be established in a client/server environment.

Internet Protocol Security (IPSEC) is a suite of protocols that provide security features for IP VPNs. IPSEC provides a means of ensuring the confidentiality and authenticity of IP packets. IPSEC works by encapsulating IP data packets into new IP packets fitted with authentication and security headers. It utilizes strong cryptography such as RSA and DES3 to provide both authentication and encryption services. Authentication ensures that packets are from the right sender and have not been altered in transit; while, encryption prevents unauthorized reading of packet contents. IPSEC was developed by the Internet Engineering Task Force and will be required as part of Ipv6.

IPSEC Implementation

The premise of IPSEC is to provide security in the form of authentication and encryption at the IP level. IPSEC implements its security using the ESP, AH, and IKE protocols. The Encapsulating Security Payload (ESP) protocol encrypts and authenticates data. The encryption is provided via a block cipher. A block cipher is a method of

encrypting data in which a cryptographic key and algorithm are applied to a block of data as a group rather than as one bit at a time. Packet authentication is done via the a hashed message authentication code (HMAC) construct that uses a hashing algorithm such as MD5 or SHA and a key to verify the integrity of the data. The Authentication Header (AH) protocol provides a packet authentication service that implements verification of connectionless integrity, data origin authentication, and optional anti-replay service. The Internet Key Exchange (IKE) protocol is responsible for negotiating connection parameters, including keys, for the ESP and AH protocols. IPSEC uses the Diffie-Hellman key agreement that is based on the discrete logarithm problem and can employ the RSA algorithm to authenticate gateways for Diffie-Hellman key negotiation.

A positive aspect of IPSEC is that it can protect any protocol running above IP and any medium that IP runs over. Conversely, higher-level security implementations such as PGP, SSH, SSL, or TLS can only protect a single protocol, and lower-level services can only protect a single medium. In this respect, IPSEC provides flexibility not had by other protocols by providing security at the IP level instead of other levels.

Though IPSEC is extremely well supported and flexible, it does have some limitations that must be considered. IPSEC does not secure the system itself, so other physical and software security must protect the integrity of the system. In addition, IPSEC is not end-to-end between applications and users and additional security implementations must be used to provide user-to-user security. Further, IPSEC authenticates machines not authenticate users. Therefore, strong authentication controls messages directing from machine to machine while user authentication must be done at the application level.

IPSEC cannot stop denial of service attacks and traffic analysis is still possible while using IPSEC. Since the headers of the encrypted packets are not themselves encrypted, a very skilled cracker can attempt to gather information such as source and destination addresses and packet size from the headers. The cracker would not, however, be able to gain information from the payloads of the packets themselves.

Even with these limitations, IPSEC's benefits can outweigh any limitations that may exist. IPSEC authentication of the underlying communication can prevent man-in-the-middle attacks by preventing forging keys. IPSEC is also transparent to the end users; therefore, all security considerations can be made at a central point of implementation.

Finally, all the major open-source operating systems – Linux and BSD-derived UNIX variants – support IPSEC. In addition, many commercial OS vendors such as Microsoft, Apple, and Sun support IPSEC. The overall appeal of the IPSEC protocol and its implementation at the transport layer is its interoperability since it doesn't specify a proprietary way to perform authentication and encryption.

Outline of Thesis

In this thesis we present the design and implementation of a production Linux VPN server. Within this implementation, we will describe all aspects of a typical Linux networked system. This will include, installation of the Linux operating system, user and group additions and modifications, password selection, file system structure, network configurations, and server daemon services. The focus during the implementation of the Linux VPN server will be on securing the system and the data contained therein. Securing a Linux system includes all aspects of the Linux system including physical and network security. The following chapters will introduce the reader to the Linux operating

system and provide an in-depth exploration of the steps necessary to secure a Linux operating system server to function robustly in a production environment.

In addition, the installation and securing of various server services will be covered with a focus on manageability of and integration with a VPN server. The FreeS/Wan IPSEC software has been chosen for its ease of manageability and integration with common server services.

CHAPTER 2
LINUX

Introduction to Linux

Millions of users worldwide use Linux for a variety of applications. Linux is used as a robust server operating system by many web sites and ISPs., and many graphic artists turn to it as an economical design workstation. Further, Linux is the development platform of choice for a large number of C programmers. Much of this can be attributed to its ability to run on less expensive PC hardware. Linux is a modern operating system running on 32-bit architectures. It uses preemptive multitasking and protected memory while supporting multiple users and networking (Schenk, n.d.).

Above all, Linux is a free UNIX-like clone that runs popular server software, and is becoming increasingly popular in the desktop arena. The success of the Linux operating system can be attributed to its flexibility, reliability, and economics.

The source code to the Linux kernel is "copylefted" under the GNU Public License, meaning that the system must be freely distributed with source code available, and anyone may freely modify that source code provided that any modifications they choose to distribute are distributed with the source code included. Since the source to Linux is open and available, it is easy to search and customize. This flexibility has enabled Linux to run on everything from hand-helds and embedded systems to clusters of hundreds of servers and mainframes. Open source accelerates the development process. It breaks down the barriers between developers and users, and removes obstacles in developer-to-developer communication (Jordan, n.d.)

The proof of the open source model is in the results: Apache holds roughly 60% market share among web servers, and that market share is growing. Sendmail holds roughly 80% market share among mail transfer agents. Linux is the fastest growing server-class operating system (Schenk, n.d.).

Each new version of the Linux operating system is rapidly viewed and tested by thousands of programmers world-wide and thus bugs are easily identified. Linux's fundamental architecture also creates a more reliable system. Systems using protected memory and preemptive multitasking are inherently more stable. Protected memory prevents an error in one application from bringing down the entire system, and genuine multitasking means that a bottleneck in one application does not hold up the rest of the system. Linux also maintains a clean separation between user processes and kernel processes. While other server class operating systems use protected memory, protected memory does no good if faulty applications are allowed to invade kernel space with their processes.

Since Linux is free and open-sourced, the initial investment in Linux is low. Linux requires no seat licenses, and has no usage fees associated with the operating system. Linux is also runs on a variety of CPU architectures, meaning it can run on low cost, widely available personal computer hardware. The total cost of ownership is a positive factor in the economics of a server class environment. Updates happen rapidly and openly rather than on the timed-release schedules of most major vendors. Support is available in a variety of forms, from a legion of open source programmers to a number of commercial supporters of Linux. The result is that bugs are identified and fixed rapidly,

new features are brought on line quickly. This means a lower up-front cost that dramatically reduces the total cost of ownership.

History

In the early 1990s, a Finnish Computer Science student named Linus Torvalds decided that he wanted a version of the industrial operating system called UNIX to run on a personal computer. The development started with Minix, a small UNIX-like operating system. Minix was originally written to assist students in learning operating system concepts and programming; because of this, it came with its complete source code. Linus developed and integrated his own source code to add some of Minix's missing functionality and in the end decided to abandon Minix and write his own version of a UNIX operating system. Finally, Linus made his improvements available to the open-source community.

Programmers from around the world responded with bug fixes, more enhancements, suggestions and encouragement. Volunteers added features with each release and eventually all the Minix code was rewritten. As a result, Linux became a completely independent project. Now, Linux has several thousand active developers worldwide, and it is currently the fastest growing server operating system. Several projects are also underway to make Linux more attractive to the desktop market (Schenk).

Linux Virtual Private Network Solution

Using Linux is it is possible to implement a gateway machine to provide secure remote access through IPSEC between two distributed sites. In addition to the connectivity between sites, the Linux gateway can provide site services for LAN

connectivity among the computers at each remote site. These services include DHCP for IP connectivity and administrative web access. Further, the gateway should provide a strong firewall implemented to mask the existence of the gateway on the Internet. The implementation should provide transparent access for each site's users to the remote sites and to the Internet.

Linux can provide a low-cost solution for small businesses and individuals for providing remote access to any IP-based network. The total cost of ownership will remain low and support is available through a variety of resources whether free or fee-based (Jordan, n.d.).

CHAPTER 3
PRE-IMPLEMENTATION CONSIDERATIONS

Hardware Requirements

To create a Virtual Private Network server using the Linux operating system, any standard Pentium-class PC will do. A Pentium II 300MHz or above with 64MB RAM, a 10GB hard disk, a CD-ROM drive, and a floppy drive should be more than sufficient.

In addition to the basic PC, two Ethernet adapters are needed. Use the highest quality cards possible, as lower end adapters tend to not perform as well. For a list of some of the Linux-supported network adapters see the "Linux Networking HOWTO – Chapter 6: Ethernet Information" (Drake, 2000).

Network Addresses

One adapter will function as the connection to the "unsecured" Internet using a reserved IP address from a service provider as shown in Table 3-1.

Table 3-1. External Network (eth0) Configuration

IP address	209.86.84.125
Netmask	255.255.255.224
Default Gateway	209.86.84.124
Primary Nameserver	172.154.232.1
Secondary Nameserver	172.153.242.2

The other adapter will function as the connection to the private intranet using a reserved IP address from the Local Area Network. The IP address range on the Local Area Network should come from a reserved internal address range according to RFC1918—"Address Allocation for Private Internets" by Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, February 1996—available at ftp://ftp.isi.edu/in-

notes/rfc1918.txt. System administrators should choose intranet IP addresses from those listed in Table 3-2.

Table 3-2. Reserved Private Network Allocated IP Addresses

Network Class	Netmask	Network Addresses
A	255.0.0.0	10.0.0.0 - 10.255.255.255
B	255.255.0.0	172.16.0.0 - 172.31.255.255
C	255.255.255.0	192.168.0.0 - 192.168.255.255

The IP address and an appropriate netmask for the external adapter will be allocated from a service provider or from an existing pool of previously requested IP addresses. In addition, it is necessary to have the IP addresses of a primary domain name server and a secondary name server to resolve Internet addresses and the IP address of a router providing access to the Internet to serve as the default gateway for traffic traveling outside the LAN. The term eth0 will refer to the network adapter providing access to the Internet, and eth1 will refer to the LAN network adapter. Only eth0 needs the domain nameserver and default Internet information; eth1 only requires an internal IP and netmask as shown in Table 3-3.

Table 3-3. Internal Network (eth1) Configuration

IP address	192.168.1.1
Netmask	255.255.255.0
Default Gateway	209.86.84.125

Once the network adapters and IP addresses are chosen and before beginning the installation of the operating system, consideration needs to be given to the structuring of the file system and to user access and security.

File System Structure

When developing any Linux-based server, consideration must be given to the layout of the file system. By using a consistent file system structure, a system administrator can readily identify changes within that file system. From a security

standpoint, by restricting the structure of the file system and consistently enforcing that structure, changes made by intruders to the system can be detected more easily and in a timely manner. For the Linux VPN system, the system administrator can use the Filesystem Hierarchy's standard to maintain the integrity of the file system.

The Filesystem Hierarchy Standard (FHS) was developed by the Filesystem Hierarchy Standard Group to allow UNIX- and UNIX clone- based software an users to predict the location of installed files and directories. The standard assumes that the target operating system supports basic UNIX security features such as symbolic linking, group and user permissions, and system and event logging.

The standard divides files into two distinct groups according to sharing permissions and system state. As UNIX-based file systems typically exist in networked and distributed environments, files are first classified based on whether they are shareable or un-shareable. Files relevant to many users or hosts will be distributed or shared to ease access and maintenance while files specific to single hosts or groups of hosts remain unshared and protected from the network and distributed system as a whole.

Secondly, the file system can contain files considered either static or dynamic relative to their specific system state. Static data includes binaries, documentation, run-level templates, or anything that does not change without the intervention of a system administrator. Dynamic data encompasses any data that will change without system administrator intervention such as system logs, error reports, mail, spools, and other user data. Often dynamic data is also referred to as variable – functioning as a storage location holding a value or values that can be replaced or overwritten with new or changing values by some internal or external means.

A common FHS-compliant system will employ the conventions listed in Table 3-4 for data based on sharing and system state (Russell & Quinlan, 2001).

Table 3-4. File System Directory Classifications

	Shareable	Unshareable
Static	/usr /opt	/etc /boot
Dynamic	/var/mail /var/spool/news	/var/run /var/lock

Of course, this is only one of many possible examples of an FHS-compliant system. By following a common FHS-compliant file system securing the file systems on a variety of UNIX-based operating systems becomes more direct and easier to manage. A system administrator can predict the locations of important and private data requiring protection and has the flexibility of scripting the permissions, sharing, and locking of the file system's data.

In open-source, UNIX-based operating systems such as Linux a particular file system structure should and commonly is followed. The file system always starts with the root directory (symbolized as /) and should include the directory structure listed in Figure 3-1 (Russell & Quinlan, 2001).

```
/ -- the root directory
|_ bin          Essential command binaries
|_ boot         Static files of the boot loader
|_ dev          Device files
|_ etc          Host-specific system configuration
|_ lib          Essential shared libraries and kernel modules
|_ mnt          Mount point for mounting a filesystem temporarily
|_ opt*         Add-on application software packages
|_ sbin         Essential system binaries
|_ tmp*         Temporary files
|_ usr*         Secondary hierarchy
|_ var*         Variable data
```

Figure 3-1. Root directory structure

Directories followed by an asterisk (*) should reside on separate partitions from other directories and the root directory and should be symbolically linked, or mounted, as part of the root directory. Further, each directory has specific requirements that must be met in order for the file system structure to remain FHS-compliant.

Filesystem Hierarchy Standard Directory Requirements

The /bin directory must not contain any subdirectories or links to other mounted directories or devices. In addition, only commands required for use in single user mode (when no other file systems are mounted) should exist in the /bin directory. The commands located in the /bin directory are available to both system administrators and users, so extra care should be taken when moving or placing files into this directory.

The /boot directory must contain all data necessary for the boot process to complete successfully including the system kernel. The data stored in /boot must not contain any references to user-mode programs. Configuration files and the map installer, though essential to the boot process, should not reside in the /boot directory since these files belong in the system configuration (/etc) and the system command (/sbin) directories respectively.

All local host configuration files should reside in the /etc directory. In addition, in order to protect the directory from corruption or failed program installs, no binaries should reside in the /etc directory. The /etc should be considered static and unshareable since only the system administrator should access this directory for changes or updates to the host configuration. Further, the /etc directory requires a /opt subdirectory to store host configuration files for add-on application software packages. Each package should create a subdirectory /etc/opt/<package> to store its configuration data. No data should reside in the /etc/opt directory, only subdirectories.

The /mnt directory is provided for temporary mounting of file systems by the system administrator only. No installation program may mount file systems in this directory and the system administrator must implement a security procedure to allow user mounting of devices and file systems either in this directory or in a suitable temporary directory within the /usr directory.

The /opt directory is where add-on applications should reside and must be located in a subdirectory -- /opt/<package>. The packages in the /opt/<package> directory should be static package objects and any binaries or man pages to be invoked by a user must be located in an /opt/<package>/bin or /opt/<package>/man directory respectively. Any subdirectory residing in the /opt directory other than add-on package directories are for local system administrator use only. If an add-on application requires the use of packages that change in normal operation, or are deemed variable, is should reside in an /var/opt/<package> subdirectory of the root directory.

The /sbin directory should contain only utilities and root-only commands for use during system administration such as booting, restoring, recovering, or repairing the file system and, therefore, should remain protected and unshared. Used only after successful mounting of the /usr file system, the /usr/sbin directory contains utilities and root-only commands not used for booting, restoring, recovering, or repairing the system. If the system administrator requires additional utilities and programs, such files and binaries should reside in the /usr/local/sbin subdirectory of /, the root.

Secondary Hierarchy

The /usr directory is the second major section of the file system and must be shared and contain static (read-only) data. The directories, or symbolic links to directories, listed in Figure 3-2 are required in /usr (Russell & Quinlan, 2001).

/usr -- Secondary Hierarchy	
|_bin	Most user commands
|_include	Header files included by C programs
|_lib	Libraries
|_local	Local hierarchy (empty after main installation)
|_sbin	Non-vital system binaries
|_share	Architecture-independent data

Figure 3-2. Directory structure of /usr

The system administrator installs local software in the /usr/local subdirectory structure according to the directories or symbolic links to directories listed in Figure 3-3.

/usr/local -- Local hierarchy	
|_bin	Local binaries
|_games	Local game binaries
|_include	Local C header files
|_lib	Local libraries
|_man	Local online manuals
|_sbin	Local system binaries
|_share	Local architecture-independent hierarchy
|_src	Local source code

Figure 3-3. Directory structure of /usr/local

By FHS standards, in order for a file system to remain standards-compliant, no other directories except for libraries directories may be in /usr/local after a systems initial installation. The data contained in /usr/local must be protected from changes when updates are performed on the system software. It is possible, if implemented carefully and securely, to share the data in /usr/local among a group of hosts. Further exploration on the sharing requirements of /usr/local will be covered in subsequent sections. The /usr/share subdirectory is for static, hardware independent data. A man/ and misc/ directory must exist as a subdirectory of /usr/share. Online help manuals should reside in /usr/share/man and miscellaneous non-platform-dependent data should reside in /usr/share/misc. (Russell & Quinlan, 2001).

All dynamic data should live in the /var subdirectory. After making this statement, should all other subdirectories be marked and/or mounted in a static (read-only) state? The answer is absolutely, positively, and mostly. An exception to any absolute always exists. Typically, a system administrator will want to provide each user his or her own "sandbox" to play in. By creating each user a directory in the /home subdirectory, a system administrator can segregate users from essential and protected system data and control each user's access to shared data. At logon, the user's environment is constructed from data or symbolic links within their home directory, and their home directory acts as the root for all their data whether static or variable. In addition, temporary subdirectories must exist in a read/write state for commands and programs invoked by users. As a general rule, a system administrator should ensure the bulk of dynamic data lives in /var hierarchy and should segregate users from protected and unshareable data.

In the FHS, the directories, or symbolic links to directories, listed in Figure 3-4 are required in /var

/var -- Variable data	
|_cache	Application cache data
|_lib	Variable state information
|_local	Variable data for /usr/local
|_lock	Lock files
|_log	Log files and directories
|_opt	Variable data for /opt
|_run	Data relevant to running processes
|_spool	Application spool data
|_tmp	Temporary files preserved between system reboots

Figure 3-4. Directory structure of /var

According to the Filesystem Hierarchy Standard, "[s]everal directories are `reserved' in the sense that they must not be used arbitrarily by some new application,

since they would conflict with historical and/or local practice. They are: /var/backups, /var/cron, /var/msgs, /var/preserve" (Russell & Quinlan, 2001).

Linux VPN File System Structure

As implementations provided in later subsections will target the open-source Linux operating system, an additional subdirectory of the root, /, files system should be addressed. Linux employs the proc file system as its de-facto standard for managing system processes and system information. In addition, tightening security requires two additional subdirectories of the root, /chroot and /home. A more detailed look at the security implications of the /chroot subdirectory will come in later sections. The /home directory contains user directories and is created automatically during the Linux install; mount it on a separate partition for extra security. The revised file system structure for a secure Linux VPN system would appear as listed in Figure 3-5.

/ -- the root directory	
|_ bin	Essential command binaries
|_ boot*	Static files of the boot loader
|_ chroot*	security jail
|_ dev	Device files
|_ etc	Host-specific system configuration
|_ home*	User directories
|_ lib	Essential shared libraries and kernel modules
|_ mnt	Mount point for mounting a filesystem temporarily
|_ opt*	Add-on application software packages
|_ proc	Process and system information
|_ sbin	Essential system binaries
|_ tmp*	Temporary files
|_ usr*	Secondary hierarchy
|_ var*	Variable data

Figure 3-5. Linux file system structure

Subdirectories of the root followed by an asterisk, *, should be placed on separate partitions of the hard disk drive. Armed with a definitive file system structure and the right hardware and network information, the system administrator should delay the

system installation further to seriously consider the security implications of user accounts and passwords.

User Accounts and Groups

In the scope of a Linux Virtual Private Network server configuration, the system administrator must make wise choices relating not only to the selection of strong passwords but also relating to the appropriate number of users requiring access to the system. By restricting the number of user and group accounts on a production machine, a system administrator can then identify the addition of or changes made to any account or group on the system. Red Hat Linux creates a default set of users at installation time. The Red Hat default user accounts and any other users created are listed in the /etc/passwd file. Figure 3-6 provides an example RedHat Linux password file.

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/var/ftp:
nobody:x:99:99:Nobody:/:
```

Figure 3-6. RedHat Linux default users

Each entry in the /etc/passwd file is in the form:

<username>:<password>:<UID>:<GID>:<description>:<home directory>:<login shell>

The password field contains only the character ‘x’ that redirects the authentication process to a shadow file containing the passwords. In earlier UNIX generations, passwords were stored in the passwd file. The problem was the /etc/passwd file needed to be readable by other groups than just root to complete the login process. So even though the passwords were “salted” with additional characters and encoded using a hash, since the file was readable, it was susceptible to a dictionary attack. If a cracker could obtain access to the /etc/passwd file she could then encode a dictionary of words and common passwords with the 4096 possible salt values and run a comparison of that dictionary to the encoded passwords. So the authentication module accepts a user’s login and then checks the password in the /etc/shadow file that is only readable through root access or a secured group access (Adams & Erickson, 1999).

Table 3-5. User and Group Accounts

Users	Groups
root	root
bin	bin
daemon	daemon
adm	sys
nobody	adm
named	tty
read	disk
write	mem
httpsd	kmem
admin	wheel
	man
	nobody
	user
	named
	read
	write
	httpsd
	admin

Red Hat also comes with a set of default groups. Linux groups are stored in the /etc/group file and each entry has the format:

<group name>:<blank>:<GID>:<member1,member2,...,membern>

Remove any non-essential user accounts and groups and check the /etc/passwd and /etc/group file for additional users created by applications or intruders. The default users and groups to keep are listed in Table 3-5; all other accounts and groups should be deleted.

Use the userdel command to remove user accounts, and the groupdel command to remove groups:

[root@turtledove /]# userdel <username>, [root@turtledove /]# groupdel <groupname>.

Five additional user accounts and groups are needed in this Linux VPN system configuration: admin, read user, write user, httpsd, and a chroot user, named, to provide a secure jail for specific services and daemons.

Use the following syntax to add a user to the Linux VPN system:

[root@turtledove /]# useradd <username>

Use the following syntax to change or set the password for a user:

[root@turtledove /]# passwd <username>

Use the following syntax to add a group to the system:

[root@turtledove /]# groupadd <group name>

For the httpsd user, use the following to create the user account:

[root@turtledove /]# useradd -c httpsd -u 80 -s /bin/false -r -d / /home/httpd https 2>/dev/null || :

Create the httpsd group in the same manner as all other groups.

Obviously root is the root user account that has uncontrolled access to all data on the system. A system administrator should never logon to a system as root. Never. Instead use an admin account. The admin account does not need truly special permissions, it simply needs read access to all data on the system. With the admin account the system administrator can su (substitute user to root superuser) to gain read/write access to all data. The system administrator should only su when absolutely necessary and should remember to exit the superuser shell with the exit command as soon as possible. Alternatively, a system can be configured to automatically log off the root superuser after a pre-determined amount of time by editing the .bashrc, or the configuration file for whichever shell is used, in the /homes/root directory and adding the line "TMOUT=<seconds to logoff>." After the account remains unused for the specified number of seconds, the user is automatically logged out of the shell. In this particular implementation of a VPN server, IPSEC configurations will use the read and read/write configuration accounts to automatically configure VPN connections. Finally, the chroot account, named, provides a secure user under which to run specific exploit-targeted services and daemons. The chroot account ensures exploit-targeted services will not run under the root account; thereby further restricting access to the system. Typically, the chroot account only has access to the /chroot partition and is in effect placed in a "jail" with no easy escape. The chroot "jail" is not completely foolproof, but is does provide another level of security a malicious entity must traverse before gaining root-level access. Further explanation of the read, write, named, and httpsd accounts will be provided in later relevant sections.

Passwords

Systems Administrators should educate users on methods for wise selection of passwords. This rarely happens. Often a system administrator must force wise password selection or assign passwords for users. The latter is a tedious process and can be avoided by using four basic schemes to enforce wise password selection.

First, the system administrator must educate herself in appropriate selection of passwords and then, if necessary, educate potential users of the system in patterns of "good" password choices. "Good" password choices can include the concatenation of two completely unrelated words that total 7 letters in length; insert a non-alphanumeric character between the two words and then capitalize at least one letter in each word – CloVe$ziT. Optionally, Choose a sentence containing seven words that is easily remembered; use the first letter of each word capitalizing at least two letters and using the punctuation from the sentence – "I was a teenage dirt bag baby!" becomes "IwatdBb!." Create a phonetic spelling of an word that spans eight characters and capitalize three of the eight letters – actual becomes aKcHOol (Spafford, 1991).

Second, the system administrator should enforce strong password selection for all accounts on the system thereby forgoing potential poor password selection. Set the minimum length for passwords to eight characters. Require at least one capital letter, one lower case letter, one number, and one non-alphanumeric character. In addition, when users change passwords, add the following additional checks: do not allow rotated versions of old password; do not allow new password that contain half of the characters from the previous password; do not allow case changes of the password only.

Third, the system administrator should make use of use of a proactive password-checking scheme using a tool such as passwd+ -- a proactive password checker by Matt Bishop (Anonymous, 1998). Such a password-checking tool will use a configuration file to enforce appropriate password selection by requiring a minimum number of characters and the inclusion of capital letters and special characters.

Fourth, require an aging period for passwords that forces password changing and revoke the password and reset it after a pre-established number of concurrent logon failures.

Enabling MD5 hashing of passwords and shadowing passwords in a separate file from /etc/passwd will also strengthen password protection on a Linux system. MD5 is a one-way hashing algorithm developed by RSA Data Security, Inc. (http://www.rsa.com) that takes data of any length and produces a 128-bit message digest. The message digest is considered non-reversible and therefore is thought computationally impossible to determine the data from the message digest (Rivest, 1992). In Password shadowing, the /etc/passwd file does not have encrypted passwords in the password field. Instead, the encrypted passwords are held in a shadow file such as /etc/shadow that is not world-readable (Jackson, 1996).

By creating a rigid structure in the design of the Linux VPN server with respect to the file system and user access, the VPN server can be made more secure. Changes that violate the structure of the production system would then be more prominent and therefore more easily detected. In the next chapter, a structured approach to the installation of the Linux operating system and the packages and binaries it contains will aid in the optimizing the detection of changes to the system by outsiders.

CHAPTER 4
INSTALLING LINUX

After considering and adapting a FHS-compliant file system, deciding on the appropriate number of users needing access, and wisely choosing strong passwords, the next step is the installation of the open-source operating system. By first carefully planning the installation and identifying the necessary packages and binaries to install, the administrator of the system can retain control of access to the files and programs on the system. Further, the identification of new programs or binaries can be made more easily if a rigid rule base is used to restrict what programs may reside on the system.

In this particular implementation of a Virtual Private Network, the target OS is a Red Hat version 6.2 Linux distribution. Any Unix-based open-source operating system will be similar in design and configuration to Red Hat Linux but may require searching for some configuration files. Note, not all options and configuration screens are covered as part of this document. Refer to the Red Hat Linux 6.2: The Official Red Hat Linux Installation Guide (RedHat Documentation Team, 2000) for further installation support. Once the installation program begins, make sure to choose "custom installation" to achieve the most control over the installation process.

Red Hat provides a reasonable installation interface that moves through partitioning the hard disk or disks and building the file system structure. Details beyond the FHS-compliant file system structure will not be provided here and those needing further help with partitioning can reference the "Linux Partition HOWTO" (Harris & Koehntopp, 2000) and the "Partitioning Your Disk for Red Hat Linux" section of the

Red Hat Linux 6.2: The Official Red Hat Linux Reference Guide (RedHat Documentation Team, 2000).

Once the partitioning is complete, select the option to install the Linux Loader (LILO) in the Master Boot Record (MBR) and select the default boot label "linux." Further explanation of the Linux Loader will be covered in Chapter 6.

Next, choose a hostname for the computer. The hostname should be in the form of a fully qualified domain name (FQDN). A fully qualified domain name contains the hostname followed by the domain name. If a registered domain name exists for the organization installing the VPN server simply preface the domain name with the computer name chosen, i.e. turtledove.destinationearth.net. In the network configuration setup specify the IP address, netmask, default gateway, and nameserver information for external network adapter, eth0. Configuration options for eth1 may need to be specified after the initial installation is completed.

Once the network configuration completes, specify the password for the root user and then add the four remaining user accounts—admin, chroot, read-only, read-write—and their appropriate passwords. In the authentication configuration, choose to use shadow passwords and enable MD5 passwords. Do not choose to enable NIS as it is not part of the VPN configuration.

When selecting packages to install ensure the option to select individual packages is checked. By selecting packages individually at install time fewer packages will need to be removed or disabled when locking down the server software. Keep in mind keeping the install to the absolute minimum required packages allows a tighter hold on security measures put in place later. Two important security considerations should be made at

package selection time: by installing only essential services, potential intruders cannot use other services to attack the server and impair or remove the essential services; and a reduction in services installed means fewer log files to search for possible intrusions and errors.

At install time, the packages are broken down into categories roughly according to their function in the system. Tables 4-1 through 4-3 list the packages to install as part of the Linux VPN system.

Table 4-1. Application Packages

Editors	
	any
Internet	
	tcpdump
	traceroute
System	
	bind-utils
	iproute
	mtools
	procinfo
	symlinks
Text	
	m4

Application Packages

Choose a familiar text editor to edit and create configuration files. Any text editor will do; creating and editing configuration files is an indispensable means of securing the Linux VPN system and will allow an easy interface to read package files and daemons that are often used to exploit a Linux-based, Internet-connected server.

Tcpdump is a command-line tool allowing monitoring of network traffic. It can capture and display incoming and outgoing packet headers on any or all network interfaces. In addition, tcpdump can filter and display packet headers based on specific criterion. This package is essential to monitoring and trouble shooting VPN traffic.

Traceroute allows a system administrator to display the route or routes used by IP packets en route to a specified network or Internet host. Traceroute displays the IP address and hostname of machines along the route taken by the IP packets to their destination. This package can aid in debugging network connectivity issues and can pinpoint the location of the problem along the route.

The bind-utils package contains a collection of utilities for querying Domain Name Service name servers to find information about Internet hosts. These utilities will provide a system administrator with IP address resolution from given host names. It can also provide additional information about registered domains and network addresses. Bind-utils is essential for any network-connected workstation or server.

Iproute is a specific tool allowing a host to take advantage of features and characteristics of Linux 2.2.x kernels. In addition, it provides connectivity to the standard network configuration utilities of ifconfig—lists network interface configurations—and route—provides host routing configuration tools

Mtools should be installed to provide seamless access to MS-DOS files. This package allows reading, writing and navigation of MS-DOS file system files. In addition, mtools supports Windows9x style long file names. This tool is more of a convenience tool than an essential file system tool. It can make it easier for a system administrator who uses Windows editors to edit files on a workstation and transfer those files to a Linux system with little or no character and formatting problems.

The procinfo utility allows access to system data in the /proc—kernel file system—directory. It provides a formatted display of the data on the standard output so a

system administrator can easily acquire information about the system from the kernel as it is running, and is thus an essential utility for the Linux system.

Install the symlinks package to aid in the maintenance of symbolic links. Symlinks checks for symlink problems such as dangling symbolic links pointing to nonexistent files and can convert absolute symbolic links to relative symbolic links. This is essentially a maintenance tool and is not necessarily vital to securing a Linux VPN system.

M4 is a macro processor that also includes functions for including named files, running UNIX commands, doing integer arithmetic, manipulating text in various ways, and using recursion. This package is the GNU—a recursive acronym for ``GNU's Not Unix"—implementation of the traditional UNIX macro processor for Linux. M4 is useful for writing text files that can be logically parsed and is used by many programs and packages as part of their build process. This is a required package for autoconf to generate configure scripts.

Table 4-2. Development Packages

Languages	
	cpp
	perl
Libraries	
	glibc-devel
System	
	kernel-headers
	kernel-source
Tools	
	autoconf
	automake
	bison
	libtool
	make
	patch

Devlopment Packages

Install the GNU C-Compatible Compiler Preprocessor (cpp) in order to have a macro processor available to gcc—GNU Compiler Collection—to transform programs before compilation. The C preprocessor provides four individual functionalities: the inclusion of header files; macro expansion; conditional compilation; and line control. This package should be installed to ensure compatibility of other package source compilation.

The Linux system needs the Perl programming language in order to handle perl scripts. Perl will be used primarily for system administration and for interactive web pages using CGI in the Apache web server. Perl—Practical Extraction and Report Language—is a language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information and is a high-level programming language with roots in C, sed, awk, and shell scripting. It's also a good language for many system management tasks.

Glibc-devel contains the header and object files necessary for developing programs that use the standard C libraries. Any Unix-like operating system needs a C library: the library that defines the ``system calls'' and other basic facilities such as open, malloc, printf, exit, and so on. The GNU C library is used as the C library in most new systems with the Linux kernel. In order to compile and create executables from source code the system needs the standard header and object files provided by the glibc-devel package.

Kernel-headers include the C header files for the Linux kernel. The header files define structures and constants that are needed for building most standard programs and for rebuilding the kernel.

Kernel-source contains the source code for the Linux kernel. The source files are needed to build most C programs that depend on constants defined in the kernel source code. In addition, a system administrator can use the source to build a custom kernel to provide additional and better security features.

GNU Autoconf is a package for configuring source code and makefiles and for generating configuration scripts. These scripts are present in a large number of free software packages and are used to detect system features at compilation time. Programmers can create portable and configurable packages with autoconf, so autoconf is mostly needed for producing packages that run on a wide variety of platforms such as Linux. Autoconf requires the m4 package.

Autoconf also requires the automake package. Automake automatically generates make files compliant with the GNU coding standards. It was inspired by the 4.4 BSD make and include files, but aims to be portable and to conform to the GNU standards for Make file variables and targets. Automake is a Perl script and thus requires the perl programming language.

Bison is a general-purpose parser generator that converts a grammar description for an LALR—Look Ahead Left Recursive—context-free grammar into a C program to parse that grammar. Bison is useful for developing a wide-range of language parsers and is upwardly compatible with YACC—Yet Another Compiler Compiler. This package

will be strictly used for development and will not necessarily remain on the finished Linux VPN system.

GNU libtool is a generic library support script that contains a set of shell scripts that automatically configure UNIX and UNIX-like architectures to generically build shared libraries. Libtool hides the complexity of using shared libraries behind a consistent, portable interface.

Make is a tool that controls the generation of executables and other non-source files of a program from a program's source files. Make gets its knowledge of how to build a particular program from a file called the makefile, which lists each of the non-source files and how to compute it from other files. This package allows a system administrator to build and install packages without any significant knowledge about the details of the build process. System administrators will use make to install add-on security and administration programs after the initial system installation.

Table 4-3. System Environment Packages

Base	
	ipchains
	krb5-configs
Daemons	
	bind
	inetd
	iputils
	mod_perl
	routed
	tcp_wrappers
Libraries	
	glib10
	krb5-libs
Shells	
	bash
	tcsh

The patch program applies diff files to original files. Patch takes a patch file containing a difference listing produced by diff and applies those differences to one or more original files, producing patched versions. It is a common way of upgrading applications.

System Environment Packages

The Linux IP Firewalling Chains program in the ipchains package is an update to the Linux kernel packet filtering code. Ipchains allows firewalling and IP masquerading. It is required to administer the IP packet filters in Linux kernel versions 2.1.102 and above. The older Linux firewalling code doesn't deal with fragments, has 32-bit counters, doesn't allow specification of protocols other than TCP, UDP or ICMP, can't make large changes atomically, can't specify inverse rules, and can be tough to manage and prone to user error. Install the ipchains package to setup firewalling for the Linux VPN server and the LAN behind the server.

Kerberos V5 is a trusted-third-party network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Install bind for client name resolution on the LAN. The Berkley Internet Name Domain (BIND) provides the DNS protocol and includes a DNS server to resolve hostnames to IP addresses. The DNS server will allow clients to name resources or

objects and share the information with other network machines. This package must only run on the LAN side of the Linux VPN server and should be isolated in a secure jail since BIND is often a target for known hacker exploits.

The inetd program listens for connections on certain network sockets. When a connection is found on a network socket, inetd decides what service the socket corresponds to and invokes a program to service the request. The server program is invoked with the service socket as its standard input, output and error descriptors. After the program is finished, inetd continues to listen on the socket. Essentially, inetd allows running one daemon to invoke several others, reducing load on the system. Inetd can be a source of many security breaches and should be configured wisely and should employ the use of tcp_wrappers.

The iputils package contains the utility ping that can send a series of ICMP protocol ECHO_REQUEST packets to a specified network host and can return information showing whether a certain network machine is alive and receiving network traffic. Iputils can be useful in diagnosing connectivity problems on the LAN.

The primary advantages of mod_perl are power and speed; it provides full access to the inner workings of the Apache web server and can intervene at any stage of request processing. This allows for customized processing of URI (Uniform Resource Identifiers are short strings that identify resources in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources) to filename translation, authentication, response generation, and logging. With mod_perl it is not necessary to start a separate process, as is often done with web-server extensions. The most widespread such extension, the Common Gateway Interface (CGI), can be replaced

entirely with Perl code that handles the response generation phase of request processing. mod_perl includes two general purpose modules for this purpose: Apache::Registry, that can transparently run existing perl CGI scripts and Apache::PerlRun, which emulates the CGI environment, allowing programmers to write scripts that run under CGI or mod_perl without change. Unlike Apache::Registry, the Apache::PerlRun handler does not cache the script inside of a subroutine. Scripts will be "compiled" every request. After the script has run, it's namespace is flushed of all variables and subroutines. This package is required to use the Webmin administration scripts.

Routed is a daemon invoked at boot time to manage the network routing tables. It uses Routing Information Protocol, RIPv1 (RFC 1058), RIPv2 (RFC 1723), and Internet Router Discovery Protocol (RFC 1256) to maintain the kernel routing table. The routed daemon listens on the udp socket for the route service for Routing Information Protocol packets and sends and receives multicast Router Discovery ICMP messages. If the host is a router, routed periodically supplies copies of its routing tables to any directly connected hosts and networks. It also advertises or solicits default routes using Router Discovery ICMP messages. Routed will provide routing information and discrimination based on destination of packets—to the VPN connection, to the Internet, or to a machine on the LAN.

With the tcp_wrappers package incoming requests for the SYSTAT, FINGER, FTP, TELNET, RLOGIN, RSH, EXEC, TFTP, TALK, and other network services can be monitored and filtered. The wrappers report the name of the client host and of the requested service; the wrappers do not exchange information with the client or server applications, and impose no overhead on the actual conversation between the client and

server applications. Tcp_wrappers are an essential tool in securing a Linux VPN system and care should be used in configuring the service to achieve the best results.

The glib10 package is essential for successful functioning of many programs part of the Red Hat Linux distribution. The package contains GLib version 1.0.6 and provides a library of utility functions for programs that are part of Red Hat. Install the necessary Kerberos libraries from the krb5-libs package.

The shell packages bash2 and tcsh satisfy several dependencies among other programs or packages. The GNU Bourne Again shell (Bash) is a command interpreter that combines features from sh, ksh, and csh shells. Bash is the default shell for Red Hat systems and bash2 includes additional features for the default bash shell. The Samba server and the perl programming language require tcsh, so it must be installed as part of the default installation.

Final Configurations

After the system administrator makes the appropriate configuration options for the installation, the Red Hat installer formats the file system and then installs the selected packages. The final step of the install is to make a boot disk for the Linux system. The system administrator should always keep a boot disk for each system in the network for emergency recover or repair of servers or workstations.

When the initial system install finishes, the installer reboots the new system and additional configuration of the Linux VPN server can continue. Once the system reboot completes, login under the root account to allow the most freedom to customize the installation. As comprehensive and flexible as the Red Hat installer is, not all options are selectable at installation time. In fact the Red Hat installer installs some extra packages

that are unnecessary or are security risks to a Linux VPN system. Thankfully an easy way to remove the extra packages exists; Red Hat uses the RPM (Red Hat Package Manager) utility to install and manage packages included with their distributions. All packages included with the Red Hat distribution are provided in RPM format, therefore removing extraneous programs or applications is no problem. The syntax for installing a package from an rpm distribution is simple:

[root@turtledove /]# rpm –i <package name>.rpm.

To remove a package use the rpm erase syntax:

[root@turtledove /]# rpm –e <package name>.

Remove the apmd package since it is the advanced power mangament daemon and the server will not need to go into power saving or standby mode. Apmd is used primarily for laptop systems to aid in conserving battery power. The gnupg package is GNU's open PGP (Pretty Good Privacy) implementation and since the system administrator will use a workstation for package downloading and verification, PGP is unnecessary and should be removed. The system hardware configured before installation of the Linux OS should remain static throughout the life of the Linux system and thus makes the isapnptools (ISA plug and play) package extraneous; remove it. A second laptop only package is kernel-pcmcia-cs and is unnecessary in a server environment. Unless a tape drive is present in the server systems the mt-st package for moving files to tape and performing tape operations is not needed and should be uninstalled. The pump package is a client-side DHCP utility for receiving network configurations from a DHCP server; the Linux VPN system should use a fixed IP address for both the internal and external network adapters and will not use pump. For systems providing redundancy

services, the raidtools provides utilities for initializing and managing raid arrays. Unless redundancy is part of the system, the raidtools are superfluous ans should be removed.

Two packages vulnerable to intruder exploit should definitely go as soon as possible. Sendmail provides email services, and since this VPN implementation does not require email services, remove it to make one less log to check for suspicious activity. The setserial package contains utilities for getting and setting serial port configurations. If this command is available, an intruder could potentially use it to cause IRQ and other resource conflict that could render the system unusable, and it should be uninstalled.

In the next chapter, the same principles of rigidity in structuring the Linux system are applied to the Linux kernel and to the installation of server services. In addition, several security issues will be identified and the reader will be presented with a means for dealing with each issue as it arises. Referring to the road map presented in Chapter 1, each design issue for the Linux server are dealt with by considering the issues impact on a production VPN server and its robustness and security.

CHAPTER 5
UPDATING THE LINUX DISTRIBUTION

To provide the means for dealing with security issues in the scope of the Linux VPN server, the fact that vulnerabilities are continually discovered and exploited by malevolents must remain forefront in a system administrator's mind. By taking the time to understand the inner-workings of the Linux kernel and by updating vulnerable services, a system administrator can more adequately decide how to secure a production Linux system. The heart of any Linux system lies in the operating system kernel, and great care must be taken while securing the kernel.

Updating the Linux Kernel

The Openwall Project provides security-related options for newer Linux kernels via a kernel patch. The patch, once installed, adds several kernel configuration options when compiling the kernel. The Openwall security options are available and configurable while configuring a new kernel for compilation via the added 'Security options" configuration section.

The patch provides the following security options, as described by the patch's author, for kernels 2.2.x that will be used in the Linux VPN implementation:

Most buffer overflow exploits are based on overwriting a function's return address on the stack to point to some arbitrary code also on the stack. By ensuring the stack area is non-executable, buffer overflow vulnerabilities become harder to exploit. Further, a malevolent can exploit a buffer overflow by pointing to the return address of a function in libc, usually system(). In addition, the Openwall patch changes the default address shared

libraries are memory mapped to in order to make it always contain a zero byte. This change makes it impossible to specify any more data parameters to the function, or more copies of the return address when filling with a pattern. However, the Openwall patch is not a complete solution; rather, it simply provides an extra layer of security. Many buffer overflow vulnerabilities will remain exploitable a more complicated way, and some will even remain unaffected by the patch. The reason for using such a patch is to protect against some of the buffer overflow vulnerabilities that are yet unknown. Also, note that some buffer overflows can be used for denial of service attacks (usually in non-respawning daemons and network clients), and this patch cannot prevent such attacks. Further, a system administrator must still stay abreast of new vulnerabilities as soon as they become known, and address such vulnerabilities even with the patch installed.

Restricting links in the /tmp directory can prevent hackers from using a hard link in an attack by not allowing regular users to create hard links to files they don't own or have permission to access. Hard links to essential system files can be used in exploits of other programs to make changes to the system files. The most well known exploit of this type is in older versions of sendmail's where undeliverable messages are appended to the end of /var/tmp/dead.letter.

> "All users can write to /var/tmp, so local attackers can create a hard link between /etc/passwd and /var/tmp/dead.letter. They then send an undeliverable message to the sendmail server. In the message body, the attacker inserts a user account to be added to the password file (preferably an account with UID 0 or root).
>
> When the message is flagged as undeliverable, it gets appended to /var/tmp/dead.letter, which is now a hard link to /etc/passwd. This results in a new system account with root privileges." (Anonymous, 1998)

In addition to restricting links, restricting writes into untrusted FIFOs (named pipes) can make data spoofing attacks harder. Enabling this option of the Openwall patch disallows writing into FIFOs not owned by the user, unless the owner is the same as that of the directory or the FIFO is opened without the O_CREAT flag.

File descriptors 0, 1, and 2 have a special meaning for the C library and lots of programs. Thus, programmers typically reference the file descriptors by number. As a result, it is normally possible to execute a program with one or more of these fd's closed, and any open(2) calls it might do will provide these fd numbers. The program (or the libraries it is linked with) will continue using the fd's for their usual purposes, in reality accessing files the program has just opened. If such a program is installed SUID and/or SGID, then it may lead to a security problem. Enable this option to ensure that fd's 0, 1, and 2 are always open on startup of a SUID/SGID binary. If any of the fd's is closed, "/dev/null" will be opened for it (Openwall Readme, n.d.).

Download the Openwall patch from the Openwall project website at http://www.openwall.org.To apply the patch to the kernel source located in /usr/src/linux do the following:

```
[root@turtledove /]# cd /usr/src/linux
[root@turtledove linux/]# patch –p1 < /usr/src/linux-2.2.19-ow1/
                                    linux2.2.19-ow1.diff
```

In addition, add the following line to the /etc/syslog.conf file to log security alerts separately:

```
kern.alert /var/log/alert
```

Once the kernel patch is applied, the options will be available in the kernel configuration options.

To install the ipchains package, change to the ipchains directory untarred into the /usr/src directory and issue the following commands:

```
[root@turtledove ipchains/]# make all
[root@turtledove ipchains/]# make install
```

The ipchains package is now installed and further ipchains firewall configurations will be discussed in Chapter 15.

Compiling and Installing the New Kernel

The Linux kernel handles interaction between programs and hardware by allocating memory, talking directly to the hardware, and dividing processor time. The kernel has many options that can be termed "user-configurable" by recompiling the kernel. Before compiling the kernel, a configuration script allows the user to choose which options to enable or disable in the new kernel configuration. Once the configuration is finished, the kernel is compiled into an image file and can be used by modifying the lilo.conf file to include access to that image.

Once the kernel archive is unpacked, the directory /usr/src/linux should contain the downloaded kernel source for Linux kernel 2.2.19. Change into the /usr/src/linux directory and issue the following command to begin configuring the new kernel:

```
[root@turtledove linux/]# make menuconfig
```

The menuconfig script provides access to the kernel options in a text-based menu. The options appear in hierarchies and can be navigated through using the keyboard. In Appendix A is a list of the kernel configuration options for the Linux VPN implementation. Each option is presented with a [Y/n/?] response. The appropriate responses are listed in bold and underlined to aid in readability. The list was created using the non-menu configuration utility "make config" and is presented in Appendix A

in that form. All options listed in Appendix A are available through the menuconfig utility. Some options not directly related to the Linux VPN implementation have been omitted for conciseness. Omissions are noted with the following text: [omission: <reason>].

If the help descriptions are not sufficient to determine the function of a particular option, refer to the configuration text by Axel Boldt for a more thorough discussion of each option (Boldt, 2000).

To compile the kernel, first make sure all kernel dependencies are in place and then clean any old sources left behind by previous kernels or configurations:

```
[root@turtledove linux/]# make dep
[root@turtledove linux/]# make clean
```

Now compile the kernel into a compressed image file. Use the following command:

```
[root@turtledove linux/]# make bzImage.
```

The new kernel is in /usr/src/linux/arch/i386/boot/ and is labelled “bzImage.” Change to the /boot directory and copy new “bzImage” file into the /boot directory as vnlinuz-2.2.19: [root@turtledove boot/]# cp /usr/src/linux/arch/i386/boot/bzImage vmlinuz-2.2.19.

The final step is to add an entry to lilo.conf to boot to the new kernel image. Make certain not to remove the entries related to the previous kernel image in case the new image does not work as expected. Add the following lines to the /etc/lilo.conf file:

```
image = /boot/vmlinuz-2.2.19        # the image file to boot
  root = /dev/hda1                  # the location of the root partition
  label = linux                     # image name listed or used at boot: prompt
  read-only                         # mount root partition as read-only
```

Change the label for the old kernel image to something like “linuxold” and save the lilo.conf file. Make sure to tell lilo about the changes by calling the /sbin/lilo script to parse the new configuration. If the new kernel functions, no further changes are necessary at this point. If the new kernel fails, reboot and type the label of the old linux kernel image at the lilo prompt to boot to the known-good kernel.

After removing unnecessary and vulnerable packages, visit the Red Hat update site http://www.redhat.com/support/errata/rh62-errata-general.html to gather up the Red Hat provided updates to the Red Hat 6.2 distribution. Create a subdirectory in the /tmp directory named RPMS using the following syntax:

```
[root@turtledove /]# mkdir /tmp/RPMS
```

```
man-1.5i-0.6x.1.i386.rpm
logrotate-3.5.2-0.6.i386.rpm
rpm-4.0.2-6x.i386.rpm
inetd-0.16-7.i386.rpm
glibc-2.1.3-22.i386.rpm
glibc-devel-2.1.3-22.i386.rpm
pam-0.72-20.6.x.i386.rpm
modutils-2.3.21-0.6.2.i386.rpm
bash-1.14.7-23.6x.i386.rpm
ncurses-5.0-12.i386.rpm
iputils-20001010-1.6x.i386.rpm
traceroute-1.4a5-24.6x.i386.rpm
sysklogd-1.3.31-17.i386.rpm
gpm-1.19.3-0.6.x.i386.rpm
textutils-2.0e-6.i386.rpm
```

Figure 5-1. RedHat Update Packages

Gather the packages in Figure 5-1 from ftp://updates.redhat.com/6.2/en/os/i386/ and place them into the /tmp/RPMS/ directory. Get the OpenSSH RPMs listed in Figure 5-2 to provide secure shell logins and secure shell ftp from the OpenBSD ftp site: ftp://ftp.openbsd.org/pub/OpenBSD/ OpenSSH/portable/rpm/RH62/. Also get the OpenSSL RPMs to provide a secure web server from http://redhat.pacific.net.au/rawhide/

i386/RedHat/RPMS/. Make certain to put the OpenSSH and OpenSSL packages listed in Figure 5-2 into the /tmp/RPMS/ directory:

openssh-2.9p2-1.i386.rpm
openssh-askpass-2.9p2-1.i386.rpm
openssh-server-2.9p2-1.i386.rpm
openssh-clients-2.9p2-1.i386.rpm
openssl-0.9.6a-6.i386.rpm
openssl-devel-0.9.6a-6.i386.rpm
openssl-perl-0.9.6a-6.i386.rpm

Figure 5.2. OpenSSH and OpenSSL packages

Note: OpenSSH requires OpenSSL, so install OpenSSL before installing OpenSSH. Grab the latest perl distribution, ActivePerl-5.6.1.626-i686-linux-thread-multi.rpm, from the Activestate web site, http://aspn.activestate.com/ASPN/Downloads/ActivePerl/ .

Once /tmp/RPMS/ contains all of the necessary Red Hat updates and other packages, use the following syntax to install each package:

[root@turtledove /]# rpm –Uvh /tmp/RPMS/*

The –Uvh command line options tell the RPM manager to upgrade(U) older existing versions of the RPMs being installed or install RPMs that previously did not exist, use verbose(v) mode, and display hashes(h) to show update progress.

In addition to the RPMs available to update the Linux VPN server, several packages unavailable in RPM format at the time of this writing are needed for this VPN implementation. The packages should be downloaded into the /usr/src/ directory; the package names and locations are listed in Figure 5-3.

Once the packages are available in the /usr/src/ directory, use the gzip and tar archive utilities to uncompress the packages. NOTE: If the directory /usr/src/linux

gmp-3.1.1.tar.gz	ftp://ftp.freesoftware.com/pub/gnu/gmp/
apache_1.3.20.tar.gz	http://httpd.apache.org/dist/httpd/
mod_perl-1.25.tar.gz	http://perl.apache.org/dist/
mod_ssl-2.8.4-1.3.20.tar.gz	http://www.modssl.org/
mod_auth_external-2.1.13.tar.gz	http://www.wwnet.net/~janc/
mod_auth_external.html	
freeswan-1.91.tar.gz	ftp://ftp.xs4all.nl/pub/crypto/freeswan/
linux-2.2.19.tar.gz	http://www.kernel.org/pub/linux/kernel/v2.2/
ipchains-1.3.10.tar.gz	http://netfilter.samba.org/ipchains/
bind-9.1.3.tar.gz	ftp://ftp.nerdc.ufl.edu/pub/mirrors/
ftp.isc.org/isc/bind9/9.1.3/	
linux-2.2.19-ow1.tar.gz	http://www.openwall.com/linux/
Net_SSLeay.pm-1.05.tar.gz	http://www.webmin.com/webmin/download/
webmin-0.88.tar.gz	http://www.webmin.com/webmin/download/

Figure 5-3. Additional Linux packages

exists, rename the directory to /usr/src/linux/RH6.2. The syntax for dealing with .tar.gz files is a series of two shell commands:

```
[root@turtledove /]# gunzip <package name>.tar.gz
[root@turtledove /]# tar –xvf <package name>.tar
```

The tar command uses the –xvf switches to extract(x) the archive file(f) in verbose(v) mode. When each package is uncompressed, a new subdirectory labeled with the package name will appear in the /usr/src/ directory.

Once the Linux system has been updated with software revisions dealing with known vulnerabilities, several additional steps must be taken to provide a secure Linux production system. In chapter 6, a discussion of physical and terminal security is provided. While in chapter 7, we address the vulnerabilities of file permissions and system processes. Each of these issues are addressed to assist a system administrator in structuring further rule sets to ensure the robustness of the server and to aid in the detection of possible attacks.

CHAPTER 6
SERVER SECURITY

Physical security

Preventing intrusions into the VPN server must start with physically securing the server. Begin by locating the server in a controlled access area. Ideally, only the system administrator and a small number of assistants should have access to the server area. By locking down the server room, simple console attacks are nearly eliminated. Only a chosen few have access to the server, so an outsider, or even an insider, to the company will not have access to force a reboot of the system to begin his or her attack.

In addition to the need for physically securing the VPN server in a limited access area, the system administrator also needs consider adding as many security measures at the actual system as possible. Most modern computers have a BIOS that allows password protecting the system boot. This is a good idea in most situations; however, administrators must recognize that after a power failure he or she must be there to provide the boot password. This is not always feasible, but whenever it is, a BIOS password should be used. In many computer BIOS configurations a password can be placed on either the boot process or to access the setup menu. If available, place a password on access to the setup menu rather than on the boot process, that way, in the event of a reboot, the machine will reboot normally with the restricted setup.

Another often-overlooked security measure is to disable the floppy drive in the BIOS and then password-protect the BIOS. This will ensure no one will be able to boot

from a floppy without first re-enabling the drive through the BIOS, and will allow a system administrator the safety of booting from a floppy if an emergency should occur.

Terminal Security

Once a user or system administrator ceases activity but does not log out—whether purposefully or not—access to the open terminal becomes a security risk. An intruder could potentially hijack the session and use that terminal connection and the user's identity to attempt break-ins to other remote systems.

The bash shell can automatically log out a user after a predetermined period of inactivity by employing the bash variable TMOUT. To set the default time out for all users, edit the /etc/profile file and add the following line just after the line listing the HISTFILESIZE variable and value:

```
TMOUT=1800
```

Bash will then log out any user who has not made any input in 30 minutes. The TMOUT command takes a value in seconds, so 30 minutes * 60 seconds/minute equals 1800 seconds. If a system administrator prefers to set the logout timeout on an individual user basis, then she should add the TMOUT variable to each users' individual .bashrc file in their home directory.

The bash shell caches commands entered at the shell prompt. Each user's bash history is stored in their home directory in the .bash_history file. Mistakes are sometimes made during a user's session—a superuser command pointing to a sensitive file, or a password entered instead of the command requiring the password—and a wise system administrator will make sure those files are not stored once a user logs off. Add the following line as the very last line of the /etc/profile file:

trap “rm –f ~$LOGNAME/.bash_history” 0

This command will be processed when a user issues the exit command at the shell prompt. It will force (-f) the removal (rm) of the .bash_history file from the user’s home directory, effectively erasing their “tracks.”

In a typical Linux system, the key combination CTRL-ALT-DELETE will force a restart of the system. A user need not be logged in to use the key combination—it is just as effective at the login prompt. Disabling the key combination reboot in situations where physical security is limited is vital, but disabling the key combination is a good idea in any implementation of a secure server.

To disable the CTRL-ALT-DELETE shutdown command, edit the /etc/inittab file and comment out the line that captures the key combination with a # symbol as shown:

#ca::ctrlaltdel:/sbin/shutdown –t3 –r now

After saving the changes, issue the following command to allow the init process to recognize the changes:

[root@turtledove /]# /sbin/init q

One of the reasons an intruder would want to force a reboot is to attempt to start the system in a different run level or in interactive mode in hopes of disabling certain services from running. Do not allow interaction with the boot process by editing the /etc/sysconfig/init file and changing the prompt value to no:

# Set to anything other than ‘no’ to allow hotkey interactive startup…
PROMPT=no

In addition to securing the terminal through init parameters and profile files, the Linux boot loader (LILO) also provides several options for adding boot time security to a Linux-based server

LILO – Linux Loader

Typically, LILO is installed as part of the master boot record and is always located at cylinder 0, head 0, and sector 1, the first sector on the primary disk. When the BIOS boots the machine, it will look at the master boot record for instructions and information on how to boot the disk and load the operating system.

> "The master boot record contains the following structures:
>
> Master Partition Table: This small table contains the descriptions of the partitions that are contained on the hard disk…One of the partitions is marked as active, indicating that it is the one that the computer should use for booting up.
>
> Master Boot Code: The master boot record contains a small initial boot program that the BIOS loads and executes to start the boot process. This program eventually transfers control to the boot program stored on whichever partition is used for booting the PC. "(Kozierok, 2001)

Once the BIOS hands control over to LILO, LILO uses a map file to locate the boot sector and the image of the operating system to start. LILO cannot interact with the file system, as a result the map file gives the exact location of the boot sector and OS image on the physical disk sectors. Once LILO loads, it checks if the shift, control, or alt key is pressed; if so, LILO provides a boot: prompt. LILO will also provide a boot: prompt with a delay if instructed to do so in a configuration file. If none of the special keys are pressed and no prompt delay is specified, LILO boots the default boot image (Veselosky, 1999). LILO can also boot an alternate boot image if specified at the boot: prompt. Once the boot prompt is available, the system administrator can press the tab key to receive a list of possible images to load or just press the enter key to load the default image. LILO is quite flexible, however, its flexibility allows a user with knowledge of it internals to effectively enter a system with his or her preferences. In fact, if a user enters "Linux single" at the boot: prompt, the "single" parameter is passed to the

init process and he or she will have access to the system in "single-user mode." Granted, no services start in single-user mode, but the user now has root access to the system.

Many of the LILO options can be specified and controlled by editing the /etc/lilo.conf configuration file. By editing the configuration file, a system administrator can control access to the various modes and command-line options that can be passed to the kernel at boot time. A typical lilo.conf file might look something like that listed in Figure 6-1 (note: line numbers have been added for reference only; the lilo.conf file should not contain line numbers or extraneous characters except when part of an end-of-line comment prefaced by a '#' character).

```
1  boot = /dev/hda                 # the location of the root partition
2  map = /boot/map                 # the location of the boot sector map
3  install = /boot/boot.b          # the location of the boot sector

5  prompt                          # displays the boot: prompt
6  timeout = 50                    # time before automatically loading default
7  password = jugaloo              # password is set to "jugaloo"
8  restricted                      # restricts command line parameters
9  message = /etc/message_file     # message displayed before LILO:
10 default = linux                 # default image to boot

12 image = /boot/vmlinuz-2.2.19    # the image file to boot
13   root = /dev/hda1              # the location of the root partition
14   label = linux                 # image name listed or used at boot: prompt
15   read-only                     # mount root partition as read-only
```

Figure 6-1. LILO configuration file – /etc/lilo.conf

The /etc/lilo.conf configuration file consists of three types of options containing flag and string variables. The flag variables specify an option to turn on and do not take values. The string variables also specify options to turn on but provide information values as parameters for those options.

In the above example, Lines 1 through 10 define global options that apply to the entire scope of the lilo.conf file. Line 1 contains the boot option that tells LILO where to

install the bootloader--/dev/hda points to the master boot record of the first hard disk, but could easily be located on any disk or on a partition rather than in the master boot record. Line 2 specifies where the sector map should be created and where LILO should look for it. Line 3 specifies what kernel-created file to use as the new boot sector. Of the first three options, only the boot option must be specified. The other two options default to the values listed in the example configuration.

The "prompt" option in Line 5 forces LILO to display the boot prompt without requiring a key-press. Specifying the prompt option without providing a timeout value will cause LILO to wait indefinitely for a boot selection; and unattended reboots are impossible in this configuration. Recognizing the need for unattended reboots, line six specifies the timeout in tenths of a second, so in the example, LILO waits five seconds before loading the default kernel image. At any time before the prompt times out, pressing ENTER will cause LILO to go ahead and load the default kernel image.

The password option in Line 7 configures LILO to prompt the user for a password before loading any kernel image. Passwords are stored unencrypted in the /etc/lilo.conf file, so the file should have permissions allowing access only to the root user (chmod 1600 /etc/lilo.conf). However, always requiring a password at boot time means no unattended reboots, so Line 8 specifies the restricted flag. The password option combined with the restricted option allows for unattended reboots and for loading any image specified in the /etc/lilo.conf file but restricts the use of command line parameters at the boot prompt. If a user wants to specify additional parameters for the kernel or for the init process, a password must be supplied before LILO will boot the specified image.

The restricted option provides a more secure boot without always requiring the system administrator be present for reboots.

If the system administrator wants to display a message prior to the LILO prompt, she can use the message option to specify a file to parse for the message. The maximum message size is 64KB and if the message file changes or is moved, the map file must be rebuilt to accommodate the change and avoid errors with LILO. The final global option specified in Line 10 indicates which image to load if no intervention is made at boot time. If the default option is left out, the first image listed in /etc/lilo.conf is used as the default.

Lines 12 through 15 contain general per-image options and per-image options for the kernel loaded. The image option tells LILO where the kernel image to load is located and the options following the image option specify image and kernel options for that image only. The root option indicates the block device that should be mounted as root, and the read-only option specifies that the root file system should be mounted in read-only mode. Typically, a process later in the system startup will remount the root file system in read-write mode after the volumes have been checked for errors. The last option in the example configuration file is the image label. At the LILO prompt, a user can press the TAB key to get the list of labels for each image loadable through LILO. After modifying the /etc/lilo.conf file, apply the changes using the command:

```
[root@turtledove /]# /sbin/lilo
```

If LILO was able to parse the lilo.conf file successfully, create or update a map for each image listed, and register the images in the boot sectors, then LILO will respond at the command prompt by listing the labels for each image prefaced by “Added” (ex.

"Added linux"). On the next reboot, the new LILO configuration will be used to start the system and load a linux kernel (Almesberger, 1998).

Boot Process

LILO uses the system map to find and boot the selected image, and the kernel calls the parent of all processes, init. Init starts the chain of events leading to the system login prompt by initializing all devices, checking each file system volume, and spawning all services, daemons, and essential system processes. Among these processes is the getty process that provides the terminal services displaying the login prompt and allowing user interaction with the operating system. Init gets its instructions for creating processes from the /etc/inittab configuration file. A sample /etc/inittab file is presented in Figure 6-2 (note: line numbers have been added for reference only; the lilo.conf file should not contain line numbers or extraneous characters except when part of an end-of-line comment prefaced by a '#' character).

Control of what software processes run at boot time and while the system continues to run is determined by choosing a "run-level" configuration. A run-level is a system software configuration that allows only a group of predetermined processes to exist. Choosing a run-level is done by editing the /etc/inittab file or by passing command line arguments to LILO.

```
1  #
2  # inittab    This file describes how the INIT process should set up
3  #            the system in a certain run-level.
4  #
5  # Author:  Miquel van Smoorenburg, miquels@drinkel.nl.mugnet.org
6  #            Modified for RHS Linux by Marc Ewing and Donnie Barnes
7  #
9  # Default runlevel. The runlevels used by RHS are:
10 #   0 - halt (Do NOT set initdefault to this)
```

Figure 6-2. Run level configuration of /etc/inittab

```
11 #   1 - Single user mode
12 #   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
13 #   3 - Full multiuser mode
14 #   4 – unused
15 #   5 - X11
16 #   6 - reboot (Do NOT set initdefault to this)
17 #
18 id:3:initdefault:
20 # System initialization.
21 si::sysinit:/etc/rc.d/rc.sysinit
22
23 l0:0:wait:/etc/rc.d/rc 0
24 l1:1:wait:/etc/rc.d/rc 1
25 l2:2:wait:/etc/rc.d/rc 2
26 l3:3:wait:/etc/rc.d/rc 3
27 l4:4:wait:/etc/rc.d/rc 4
28 l5:5:wait:/etc/rc.d/rc 5
29 l6:6:wait:/etc/rc.d/rc 6
30
31 # Things to run in every runlevel.
32 ud::once:/sbin/update
33
34 # Trap CTRL-ALT-DELETE
35 #ca::ctrlaltdel:/sbin/shutdown -t3 -r now
36
37 # When our UPS tells us power has failed, assume we have a few minutes
38 # of power left.  Schedule a shutdown for 2 minutes from now.
39 # This does, of course, assume you have powerd installed and your
40 # UPS connected and working correctly.
41 pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
42
43 # If power was restored before the shutdown kicked in, cancel it.
44 pr:12345:powerokwait:/sbin/shutdown -c "Power Restored”
46 # Run gettys in standard runlevels
47 1:2345:respawn:/sbin/mingetty --noclear tty1
48 #2:2345:respawn:/sbin/mingetty tty2
49 #3:2345:respawn:/sbin/mingetty tty3
50 #4:2345:respawn:/sbin/mingetty tty4
51 #5:2345:respawn:/sbin/mingetty tty5
52 #6:2345:respawn:/sbin/mingetty tty6
```

Run level configuration of /etc/inittab continued

In the example inittab file listed in Figure 6-2 all commands for init are given in the form: id:run-levels:action:process. Where id is a unique set of one to four characters

identifying the entry in inittab. Init can run Linux in any one of eight possible run-levels. Each Linux distribution can determine what the run-levels will be. Seven of the eight run-levels in RedHat are listed in the comment Lines 10 through 16 of the example init configuration file comments. The final but duplicate RedHat run-level is “S” or “s” for single user mode. The action taken by init can be one of any of the actions listed in table 6-1. The process specified can be any binary executable or shell script.

Line 18 sets the default run-level to three using the initdefault action under the identifier id. Next, in Line 21, init will run the sysinit action regardless of the run-level specified in any other instruction. The sysinit action will start the /etc/rc.d/rc.sysinit process to initialize the system and fork off other processes. In addition, Line 32 instructs init to start the /sbin/update process regardless of the run-level specified in any other instruction. The /sbin/update process periodically updates the file system by flushing data from memory to disk every 30 seconds. Lines 47 through 52 spawn virtual terminals and will respawn them if they stop for any reason. In this particular Linux server implementation, only one terminal will be needed so Lines 48 through 52 have been commented out.

In addition, the /etc/securetty file contains a list of the tty devices on which the root user can login. The login program parses the /etc/securetty file and the root user can login to any virtual terminal, tty, listed. Edit the /etc/securetty file to allow the root user to only login on the first terminal, tty1, by commenting out or removing any reference to other virtual terminals. The other terminals are disabled in the inittab file, however, the getty process can be run at any time. By also disabling the terminals in the /etc/securetty file, root level access will be one step harder to acquire by an intruder.

Table 6-1. Inittab actions defined

respawn	The process will be restarted whenever it terminates.
wait	The process will be started once when the specified runlevel is entered and init will wait for its termination.
once	The process will be executed once when the specified runlevel is entered.
boot	The process will be executed during system boot.[1]
bootwait	The process will be executed during system boot, while init waits for its termination.[1]
off	No action.
ondemand	The process will be executed whenever the specified on demand runlevel is called.
initdefault	Specifies the runlevel which should be entered after system boot. If none exists, init will prompt for one.[2]
sysinit	The process will be executed during system boot. It will be executed before any boot or bootwait entries.[1]
powerwait	The process will be executed when init receives the SIGPWR signal. Init will wait for the process to finish before continuing.
powerfail	Same as powerwait but init does not wait for the process to complete
powerokwait	The process will be executed when init receives the SIGPWR signal provided there is a file called "/etc/powerstatus" containing the word "OK." This means the power has come back again.
ctrlaltdel	The process is executed when init receives the SIGINT signal. This means someone on the system has pressed the "CTRL-ALT-DELETE" key combination.
kbrequest	The process will be executed when init receives a signal from the keyboard handler that a special key combination was pressed on the console keyboard.

[1] The runlevels field is ignored; [2] The process field is ignored

Lines 23 through 29 specify the instructions for init to perform for each run-level. For each run-level init will make a call to /etc/rc.d/rc with the run-level number sent as a command-line parameter; init will wait for the rc process to terminate before continuing to process the inittab instructions. The rc process will execute a series of commands and then process all scripts located in the /etc/rc.d/rc*.d directory where * represents the current run-level. Within the /etc/rc.d/rc*.d directory, the rc process searches and executes scripts whose names begin with the letter 'S.' Each script name will begin with an 'S' followed by a number. The numbers tell the rc process in which order to execute

the scripts, and the ‘S’ instructs the rc process to start the script with “start” as a command-line parameter. In addition to the “start” scripts, there are scripts in each run-level directory that specify what to shut down in what order when leaving a run-level. The “kill” or shutdown script names begin with a ‘K’ instead of an ‘S.’

The files in the /etc/rc.d/rc*.d directories are really just links to the actual scripts located in the /etc/rc.d/init.d. The scripts contained in this directory are instructions for how to start, stop, or restart, a particular subsystem or daemon. So any modifications to existing scripts or the addition of new scripts should be performed in the /etc/rc.d/init.d directory.

One way to secure the Linux VPN server is to remove unneeded services or services vulnerable to known exploits from certain run-levels. One group of thinking suggests that removing the services from the system entirely is the best way to secure the system. Others believe that disabling the services is just as effective and will allow ease of re-enabling the service later--allowing a much larger range of run-level options. To remove the service from the system entirely, use the rpm command with the syntax: [root@turtledove /]# rpm –e <package name>.

Here we identified measures to aid the system administrator in physically securing the Linux system. By now, the reader should have a good understanding of the fact that security is an ongoing issue in any production system and can best be addressed initially with a solid, robust rule set for access to the server. In the next chapter, this concept will be extended to include file permission and system process security.

CHAPTER 7
PROCESS AND FILE SYSTEM SECURITY

Setting Resource Limits on Processes

Even if an intruder cannot gain root-level access to the Linux system, if they can gain access to the system as a user they may be able to mount a Denial of Service attack on the system by spawning multiple processes that consume the process space, memory, and the most CPU cycles. In such a case, no resources are left for the server's main functions and processes. To prevent such a user attack, edit the /etc/security/limits.conf file and add or change the following:

```
*       hard core       0
*       hard rss        5000
*       hard nproc      20
```

The /etc/security/limits.conf file consists of entries in the form <domain> <type of limit> <item to limit> <value>. By adding the lines listed above with the domain *, these rules apply to all users except the root, super, user. The type of limits for each of the entries is a hard limit—a maximum limit. In setting a hard limit on core files to zero, hard core 0, core files cannot be created by users. As a result, attempting to cause a core dump to access valuable information in the dump file is not possible since no core file will be created. The rss item restricts the amount of memory available to users to a hard limit of 5MB, so no one user will be able to consume all available memory on the server by launching many copies of a memory hungry program. Finally, the nproc item limits any user to a maximum of 20 processes, so the system cannot be flooded with processes that consume resources or use up all the file descriptors with a "fork bomb."

In addition, in order to enable the limits at login, the following line must be added as the last line in the /etc/pam.d/login file:

session required /lib/security/pam_limits.so

This line applies whenever a user logs in or out of the system. The pam_limits.so Pluggable Authentication Module (PAM) will read the /etc/security/limits.conf file whenever a user session is opened or closed, and the limits listed in the file will be applied to that user session.

Securing the cron daemon

In multi-user mode, the cron daemon is where all scheduled events are initiated. Cron runs as a background process that wakes up every minute and scans all the stored crontab files—configuration files known as cron tables—to check each of them for commands needing to be executed at the current time. In Red Hat 6.2, the directories /etc/cron.d, /etc/cron.daily, /etc/cron.hourly, /etc/cron.monthly, and /etc/cron.weekly contain the commands for scheduled events. By default, the permissions on these directories and their contained files are –rwxr-xr-x, which allows read and execute access to all users. To tighten down the cron table locations so only the root user and admin group can access the directories and files, change the directory and file permission to –rwxr-x--- by executing the following commands:

```
[root@turtledove /]# chmod –R 750 /etc/cron.hourly
[root@turtledove /]# chmod –R 750 /etc/cron.hourly/*
[root@turtledove /]# chmod –R 750 /etc/cron.daily
[root@turtledove /]# chmod –R 750 /etc/cron.daily/*
[root@turtledove /]# chmod –R 750 /etc/cron.weekly
[root@turtledove /]# chmod –R 750 /etc/cron.weekly/*
[root@turtledove /]# chmod –R 750 /etc/cron.monthly
[root@turtledove /]# chmod –R 750 /etc/cron.monthly/*
```

In addition, the crontab file resides in the /etc directory and directs the cron daemon as to when to execute the commands in the various cron directories. The crontab file should have permissions set to –rw-r--r-- and should be owned by the root user so only the root user and admin group have access to the file. Interestingly, the crontab file is not created or maintained by editing the /etc/crontab file. Instead, the command crontab is used to edit, list, create, or remove a crontab file. If not properly configured, all users can execute the crontab command; even though the users can only create crontab files for themselves, the potential for a seasoned user to use cron to violate the system security still exists. System administrators should deny access to the crontab command in order to prevent scheduling of system unfriendly, or security compromising tasks. When a user invokes the crontab command, it searches for and examines the files /etc/cron.allow and /etc/cron.deny to grant or deny the modification of the crontab file. If the cron.allow exists, each user able to use the crontab command must be listed in the file. If the cron.allow files does not exist but the cron.deny file does exist, the each user able to use the crontab command must not be listed in this file. If neither cron.allow nor cron.deny exist, then only the root user can invoke the crontab command. In the Linux VPN server ensure the cron.allow and cron.deny files do not exist and the correct permissions are set by the crontab command so only the root user and members of the root group can use the crontab command. (Mourani & Madhusudan, 2000)

Console Apps Security Directory

The /etc/security/console.apps directory contains console-equivalent access to common server commands for regular users on the server. A console command is one that is generally only executed at the physical console in the server's location. However,

if files linked to those commands exist in the console.apps directory, then the commands may be executed from a remote terminal session. Disabling access to these commands is a good idea, and only the root user should have access to these commands, which includes the shutdown and halt commands. To disable console-equivalent access to these commands, remove the files from the /etc/security/console.apps directory:

[root@turtledove /]# rm –f /etc/security/console.apps/<command>

The rm command will delete the command files from the console.apps directory. All commands except kbdrate should be deleted from this directory. The kbdrate command is used to reset the keyboard repeat rate and delay time and is therefore relatively harmless. Make sure to check the permissions for the directory so that only root has write access to the folder.

In addition, to completely disable console access for all users other than root, search through the files in the /etc/pam.d directory and comment out any reference to pam_console.so. Use the grep command to find the files containing references to pam_console.so:

[root@turtledove /]# grep –ri pam_console.so /etc/pam.d/

The grep program searches through files and directories and returns lines matching a specified pattern to the standard output. Once grep returns all references to pam_console.so open each referenced file and place a "#" in front of the line(s) containing pam_console.so.

Brief Explanation of Linux File Permissions

Linux, like all UNIX-based derivatives, separates access control on files and directories by the owner, her group, and everyone else. For each file and directory there

is exactly one owner, but there may be many members in the group to which the owner belongs so a system administrator must be aware of how and why groups are constructed. In this Linux implementation, using Red Hat 6.2, each username is also the name of a group, and each user is the only member of his or her group, so group permission errors and maintenance is reduced.

The owner of a file or directory has control of permissions settings of the file or directory and it parent. Permissions are set using three bits assigned to the owner, the group, and everyone else for a total of nine bits plus one bit to indicate a directory, file, or link. The extra bit is important for several reasons, but in relation to securing files and directories the permissions for a directory sometimes have a different meaning than the same set of permissions on files (Bandel, 2000).

The three bits per owner, group, or other, indicate read, write, and execute permissions. Read access allows viewing the contents of a directory or file. Write access allows additions or changes to existing files and the creation of new files. Write permissions also allow deleting and moving directories or files. Execute access allows running of binary programs or shell scripts. Execute permissions combined with read permissions allow searching of directories. To list the contents of the current directory and see the currently set permissions, use the command:

```
[root@turtledove /]# ls –l
```

In the above example, ls is the list contents of a directory command and the command line switch –l specifies a long directory listing. A typical long directory listing with permissions may look like the following:

```
total 1048
-rw-r--r--   1 root     root        2045 Nov 28 05:17 DIR_COLORS
```

```
drwxr-xr-x   2 root     root        4096 Jan  9 14:07 codepages/
drwxr-x---   2 root     root        4096 Jan  9 14:04 default/
-r--------   1 root     root         299 Jan  9 14:07 gshadow
-rw-------   1 root     root        1578 Nov 28 05:19 ipsec.conf
lrwxrwxrwx   1 root     root          12 Jan  9 14:07 redhat-release -> /etc/release
```

From left to right, the first bit in a permissions listing is reserved for directories or links and is set to either ‘d’ or ‘l’ respectively when the object is created. The second through fourth bits are for the owner’s permissions, the fifth through seventh bits are the group permissions, and the final three bits are for the “everyone else” permissions. In the above listing, the DIR_COLORS permissions are -rw-r--r-- and indicates it is a file with the owner retaining read/write permissions and the group and everyone else having read-only permissions. The permissions can also be indicated by using a three digit number where the hundreds digit represents the owner’s permissions, the tens digit the group permissions, and the ones digit everyone else’s permissions. The number used for each digit is one of the eight possible three-bit permutations for the read, write, and execute permissions—where four represents read permission, two represents write permission, and one represents execute permission. See Table 7-1 for an example.

Table 7-1. File Permissions

Number	Permissions
400	Owner has read permission
200	Owner has write permission
100	Owner has execute permission
040	Group has read permission
004	Everyone has read permission
744	

For example, the number 6 would represent –rw indicating read/write permissions. The DIR_COLORS permissions could then be written as 644 indicating -rw-r--r--. Using the three-digit representation, permissions can be set by using the chmod command:

[root@turtledove /]# chmod 0660 DIR_COLORS

The above command would set –rw-rw---- permissions, where the owner and any member of the owner's group have read and write privileges on the DIR_COLORS file.

Notice in the above example the chmod command can take a four-digit number instead of the three-digit number previously explained. A four in thousands position sets the user ID (SUID) upon execution, a two sets group ID (SGID) upon execution or sets manadatory locking, and a one sets the "sticky" bit.

[root@turtledove /]# chmod 1755 codepages

A sticky bit set on a file has no effect, but when a sticky bit is set on a directory, files in that directory can only be linked, unlinked, renamed, or deleted by their owner. The common application of the sticky bit is on directories, such as /tmp, that are world writeable—everyone can write to the directory. Enabling the sticky bit can prevent security attacks involving creating links to delete important system files such as /etc/password or /etc/shadow. When the sticky bit is set, it appears as a t in a long directory listing.

Normally an executed binary runs as the user who invoked the command, but if the SUID bit is set on the binary, the binary will run as the owner of the file. This means any process that runs a binary with the SUID set will have access to system resources based on the user who owns the file. Often, the SUID is the cause of many buffer overflow exploits. In a buffer overflow exploit, the intruder attempts to overwrite memory not allocated to the currently running process by overflowing user input dependent buffers with her own code. Then the process is made to run the dangerous

code. If the process is running as a root process, then the hacker has just gained root privileges to the system.

The SGID bit grants an executable binary the permissions of the file owner's group. If the SGID is set on a directory, all files in the directory will have the SGID set. If the root group owns an executable and the executable is vulnerable to a buffer overflow exploit, then a malevolent may gain root access to the system. A system administrator should be wary of file permissions and their impact on a system's security.

Securing File System Permissions

Begin by changing the default permissions (umask) set during file or directory creation. By setting default permissions for future files and directories, the system administrator ensures users will not receive the wrong permissions that may allow read and write of vital or secure files and directories. The default permissions can be set in the file /etc/profile. Before the line USER= add or change the line to read:

umask 077

The umask is an octal number containing three digits. When setting the default user permissions on files and directories, the umask is logically AND-ed with the octal 0777. By setting the umask to 017, all files and directories created will have permissions set to 0770 thereby denying read, write, and execute permissions to anyone except the owner of the file. Only the root user can change permissions using the chmod command. The system administrator will now have greater control over file permissions and will be less likely to unknowingly set vulnerable directory and file permissions.

Next, edit the /etc/fstab file to disable SUID and SGID permissions, control mounting of devices, and restrict where executables may run. Inside the fstab file is a

listing of the user allowed mounts; any user other than the root user cannot mount any partition or block or character device not listed here. In most cases, users need access to the most partitions listed, however restricting actions a user can perform in those partitions is a good step toward securing the Linux system. Figure 7-1 lists an appropriately configured fstab file for a Linux VPN server implementation:

/dev/hda8	/	ext2	defaults	1 1
/dev/hda1	/boot	ext2	defaults	1 2
/dev/hda10	/chroot	ext2	defaults	1 2
/dev/hda12	/home	ext2	defaults,noexec,nosuid,nodev	1 2
/dev/hda9	/opt	ext2	defaults	1 2
/dev/had11	/tmp	ext2	defaults,noexec,nosuid,nodev	1 2
/dev/hda7	/usr	ext2	defaults	1 2
/dev/hda6	/var	ext2	defaults,nosuid,noexec	1 2
/dev/hda5	swap	swap	defaults	0 0
/dev/hdc	/mnt/cdrom	iso9660	noauto,ro	0 0
/dev/fd0	/mnt/floppy	ext2	defaults,noauto	0 0
none	/proc	proc	defaults	0 0
none	/dev/pts	devpts	gid=5,mode=620	0 0

Figure 7-1. Mountable file system table

The fstab file contains six columns of information specific about the partitions and devices available for mounting. The first column specifies the device type; typically ide drives (hd), SCSI drives (sd), and floppy drives (fd). In the second column is the mount point in the filesystem for the device. Next is the filesystem type specific to the device or OS filesystem. The fourth column specifies the options for users other than root for mounting and using the filesystem. The defaults option implies: rw, suid, dev, exec, auto, nouser, and async.

Users other than root can only mount file systems listed in the fstab file with the user option, and the /home, /tmp, and /var mounts should be protected using the nosuid option to avoid overflow exploits and the noexec option to restrict where users can execute binaries. The /home and /tmp mounts should use the nodev option to ignore

attempts to mount character or block devices. Though these changes to /etc/fstab will not stop a seasoned hacker, it can protect the system from hacker "kiddies"—new, non-hackers with limited file system knowledge using exploit scripts and how-tos found on the internet.

Check and set, if necessary, the appropriate permissions on the important and protected files listed below. Note, use the chmod command to change the permissions to match the listing in Figure 7-2 (Ranch, 2000).

chmod 1600 /etc/inetd.conf	chmod 750 /usr/bin/control-panel
chmod 1600 /etc/lilo.conf	chmod 750 /usr/bin/eject
chmod 1644 /etc/services	chmod 750 /usr/bin/glint
chmod 1644 /etc/passwd	chmod 750 /usr/bin/gpasswd
chmod 1600 /etc/shadow	chmod 750 /usr/bin/ipx*
chmod 1644 /etc/group	chmod 750 /usr/bin/kernelcfg
chmod 1600 /etc/gshadow	chmod 755 /usr/bin/lp*
	chmod 4755 /usr/bin/lpr
chmod –R 700 /etc/rc.d/init.d/*	chmod 750 /usr/bin/mformat
	chmod 750 /usr/bin/mtools
chmod 644 /var/log/wtmp	chmod 750 /usr/bin/netcfg
chmod 644 /var/log/utmp	
	chmod 750 /usr/sbin/am*
chmod 660 /dev/lp*	chmod 750 /usr/sbin/at*
	chmod 750 /usr/sbin/automount
chmod 750 /bin/linuxconf	chmod 750 /usr/sbin/bootp*
chmod 750 /bin/mount	chmod 750 /usr/sbin/crond
chmod 750 /bin/ping	chmod 750 /usr/sbin/dhc*
chmod 750 /bin/rpm	chmod 750 /usr/sbin/dip
chmod 750 /bin/umount	chmod 750 /usr/sbin/edquota
	chmod 750 /usr/sbin/fixmount
chmod 750 /sbin/accton	chmod 750 /usr/sbin/ftpshut
chmod 750 /sbin/badblocks	chmod 750 /usr/sbin/group*
chmod 750 /sbin/ctrlaltdel	chmod 750 /usr/sbin/grp*
chmod 750 /sbin/chkconfig	chmod 750 /usr/sbin/in.*
chmod 750 /sbin/debugfs	chmod 750 /usr/sbin/inetd
chmod 750 /sbin/depmod	chmod 750 /usr/sbin/klogd
chmod 750 /sbin/fdisk	chmod 750 /usr/sbin/logrotate
chmod 750 /sbin/fsck*	chmod 750 /usr/sbin/lp*
chmod 750 /sbin/ftl*	chmod 755 /usr/sbin/lsof
chmod 750 /sbin/getty	chmod 750 /usr/sbin/makemap

Figure 7-2. Default file permissions

chmod 750 /sbin/halt	chmod 750 /usr/sbin/mk-amd-map
chmod 750 /sbin/hdparm	chmod 750 /usr/sbin/named*
chmod 750 /sbin/ide_info	chmod 750 /usr/sbin/nmbd
chmod 750 /sbin/if*	chmod 750 /usr/sbin/newusers
chmod 750 /sbin/init	chmod 750 /usr/sbin/netreport
chmod 750 /sbin/insmod	chmod 750 /usr/sbin/ntp*
chmod 750 /sbin/ipfwadm	chmod 750 /usr/sbin/ntsysv
chmod 750 /sbin/ipx*	chmod 750 /usr/sbin/pppd
chmod 750 /sbin/kerneld	chmod 750 /usr/sbin/pw*
chmod 750 /sbin/killall*	chmod 750 /usr/sbin/quota*
chmod 750 /sbin/klogd	chmod 750 /usr/sbin/rdev
chmod 750 /sbin/lilo	chmod 750 /usr/sbin/rdist
chmod 750 /sbin/mgetty	chmod 750 /usr/sbin/rotatelogs
chmod 750 /sbin/mingetty	chmod 750 /usr/sbin/rpc*
chmod 750 /sbin/mk*	chmod 750 /usr/sbin/rwhod
chmod 750 /sbin/mod*	chmod 750 /usr/sbin/setup
chmod 750 /sbin/netreport	chmod 750 /usr/sbin/showmount
chmod 750 /sbin/pam*	chmod 750 /usr/sbin/snmp*
chmod 750 /sbin/pcinitrd	chmod 750 /usr/sbin/syslogd
chmod 750 /sbin/portmap	chmod 750 /usr/sbin/tcpd*
chmod 750 /sbin/restore	chmod 750 /usr/sbin/time*
chmod 750 /sbin/runlevel	chmod 750 /usr/sbin/tmpwatch
chmod 750 /sbin/syslogd	chmod 750 /usr/sbin/traceroute
chmod 750 /sbin/swapon	chmod 750 /usr/sbin/user*
chmod 750 /sbin/tune2fs	chmod 750 /usr/sbin/vi*
chmod 750 /sbin/uugetty	chmod 750 /usr/sbin/wire-test
chmod 750 /sbin/vgetty	

Default file permissions continued

Find and examine all root owned files and files with the SUID/SGID bit set to make certain root owned SUID files are not writable by other users. To find such files use the command:

[root@turtledove /]# find / -type f \(-perm –04000 –o –perm –02000 \) –ls

For any SUID file not needed, change the file permissions to 700 to ensure no one else can write to the file. In addition, make note of the SUID files and regularly check that no new SUID files have appeared.

Be wary of any world-writable files on the system, and know if and why those files need to be world-writable. These files, particularly system files, can be dangerous

security holes if a malevolent gains access to the system and modifies them. Find all world-writable files and directories:

[root@turtledove /]# find / -perm 2 ! –type l ! –type b ! –type c –ls

The above command will list all files and directories that are world-writable excluding symbolic links (! -type l), block special devices (! –type b), and character special devices (! –type c). These types are excluded from the search since many of them will be world-writable in the course of normal system operation.

R-command files relate to remote access or operation of the system and should be regularly searched for and removed if not a part of the original installation. Use the command:

[root@turtledove /]# find / -grep –e ".rhosts" –e "hosts.equiv"

Neither .rhost files nor host.equiv files should reside on the system in the course of normal operation and therefore need to be removed to prevent insecure accounts vulnerable to intrustion.

System Logs

The system logs are important tools for detecting intrusions. Hackers typically like to remove their traces from the system logs, and anyone skilled enough to get root access to the system can and will cover their tracks. Locking down and splitting up the system logs can provide another level of protection by making it somewhat more difficult to find and access the system logs. Start by modifying /etc/syslog.conf by adding the following lines:

```
#Split logfiles according to function
*.warm;*.err                    /var/log/syslog
auth.*;user.*;daemon.none       /var/log/loginlog
kern.*                          /var/log/kernel
```

All space between the two columns must be tabs not spaces, otherwise the syslog daemon will fail to load. Next, stop the syslog daemon:

```
[root@turtledove /]# /etc/rc.d/init.d/syslog stop
```

Make sure all logfiles in the /var/log directory and its subdirectories have 0600 permissions set. In addition, set the permissions for any subdirectory of /var/log to 0700 to allow the creation of new files in those directories. Add the following lines to the /etc/logrotate.d/syslog file to allow rotation of the new logs:

```
/var/log/kernel {
        postrotate
                /usr/bin/killall –9 klogd
                /sbin/klogd &
        endscript
 }
/var/log/loginlog {
        postrotate
                /usr/bin/killall –HUP syslogd
        endscript
 }

/var/log/syslog {
        postrotate
                /usr/bin/killall –HUP syslogd
        endscript
}
```

Figure 7-3. System log configuration

Restart the syslog daemon:

```
[root@turtledove /]# /etc/rc.d/init.d/syslog start
```

After the syslog daemon restarts, the principle steps to securing the file system are complete. Each of the above recommendations will add another level of security to the Linux VPN system implementation and will ease system administration. A good system

administrator will take the time to regularly review the system logs and check for changes in permissions of protected files that may indicate system intrusions (Ranch, 2001).

In this chapter, a file permission primer was provided to aid in the establishment of a rule set for locking access to vital system files. In addition, limits were set on system processes to aid in the prevention of certain denial of service type attacks. Further, the reader was introduced to the system log as a means for monitoring system activity. With a basic set of rules implemented to deal with basic system security in place, the following chapters will identify further mechanisms to deal with authentication, encryption, and security.

CHAPTER 8
ACCESS AUTHENTICATION

In a VPN environment, authentication is a vital service that must be provided. Through a well-designed authentication mechanism, access to the system can be restricted to users who can provide an appropriate level of information to substantiate they are really who they claim to be. In the Linux environment, Pluggable Authentication Modules can aid the system in thoroughly identifying valid users requesting access to the system.

Pluggable Authentication Modules

The Pluggable Authentication Modules (PAM) for Linux is a suite of shared libraries that control how applications authenticate users. Instead of building support for the changing and evolving authentication schemes directly into a program and then recompiling the program to use that authentication, a program can be compiled once to enable the program as PAM-aware. Once a program is PAM-aware, then control of the authentication method used is done using various authentication modules and can change as authentication schemes evolve. In other words, the modules provide a library of functions that a PAM-aware application may use to request user-authentication (Bandel, 2000).

Any authentication scheme may be used, including stacking authentication schemes to require multiple levels of authentication. A module need only return to the asking program SUCCESS, IGNORE, or FAILURE. The return value is determined by the restrictions specified for the program as listed in its corresponding PAM file in the

/etc/pam.d directory. For example, if the login service is a restricted service requiring authentication from an authentication module, then a file named login and containing the service's restrictions would be in the /etc/pam.d directory. A sample /etc/pam.d/login file is provided in Figure 8-1.

```
#%PAM-1.0
auth            required        /lib/security/pam_securetty.so
auth            required        /lib/security/pam_pwdb.so     shadow md5
auth            required        /lib/security/pam_nologin.so

account         required        /lib/security/pam_pwdb.so

password        required        /lib/security/pam_pwdb.so use_authtok md5 shadow

session         required        /lib/security/pam_pwdb.so
session         required        /lib/security/pam_limits.so
```

Figure 8-1. Configuration of /etc/pam.d/login

A typical service's PAM file will contain as many as four columns: <module-type> <control-flag> <module-path> <arguments>. If a particular service requires authentication, then one or more auth module-type configuration lines will appear in the service's PAM file. In addition, the password module-type can be used to ensure password changes are made when needed. The account module-type simply checks to see that the user attempting to use the program or service is actually permitted to do so. Finally, the session module-type performs actions when a user logs in or out; the actions can include just about any command such as mounting or unmounting directories or logging user activity.

The second column contains the control-flag for each configuration line. The required control-flag indicates that a certain module is required for this service and may not return FAILURE. As the PAM file is parsed, if a module marked required intends to return FAILURE parsing of the remaining configuration lines continues before returning

the failure result. This ensures other required modules will be evaluated and the appropriate actions defined for those modules taken. An optional control-flag is only relevant if no other module of the same module-type is defined. Otherwise, a FAILURE is ignored. For example, if the only listed session module-type has the optional control-flag set, then the FAILURE is returned to the service; but if more than one session module-type is listed, then the one with the optional flag will not determine whether a SUCCESS or FAILURE is returned.

The third column in a configuration line of any service's or program's PAM file contains the full path to the module to be used for authentication. The fourth column would contain any necessary parameters needing to be passed to the module specified (Bandel, 2000).

In the example /etc/pam.d/login file, authentication (auth) module-types are used to ensure only the root user has access to login to the secure terminals listed in the /etc/securetty file, check the /etc/passwd file to verify the user and user ID (UID) exist, and check for the existence of the /etc/nologin file. If the /etc/nologin file exists, then only the root user may login to the system. If all auth configuration-line restrictions are met, then SUCCESS is returned by auth. The account configuration line checks the user's password against the /etc/shadow file and if it matches, account returns SUCCESS. The password configuration line will check the user's password to update any expired authentication tokens. When a user logs in or out the session configuration lines check the pwdb module to determine if the user is authorized to use the service and invoke any limits listed in the /etc/limits.conf file.

The /etc/pam.d/other file is intended to catch other PAM-enabled services that do not have a configuration file in the /etc/pam.d directory. Figure 8-2 lists a commonly configured other file.

```
#%PAM-1.0
auth        required    /lib/security/pam_deny.so
account     required    /lib/security/pam_deny.so
password    required    /lib/security/pam_deny.so
session     required    /lib/security/pam_deny.so
```

Figure 8-2. Configuration of /etc/pam.d/other

This configuration file denies access to any restriced service on the system by calling the pam_deny.so module to always return FAILURE to the calling service.

The Pluggable Authentication Modules provide authentication for any PAM-enabled services on the Linux server. Other PAM-enabled service configurations will be covered as part of each service's individual configuration section.

PAM and Super-User Commands

The su (Substitute User) command allows a user to become another user on the system. For example, since the root user has super-user privileges, root can use the su command to execute commands as any user on the system. This makes administration of a particular user's files easier since files can be created as that user rather than as root and changing the ownership of the new files. The syntax of the su command is su <user>, and typically issuing the su command with no arguments will default to the root user account.

In a secure environment access to the su command is restricted to the root user or to a small, trusted group of users. Add the following lines before the other configuration lines listed in the /etc/pam.d/su file:

```
auth    sufficient      /lib/security/pam_rootok.so
auth    required        /lib/security/pam_wheel.so    group=wheel
```

These two lines allow the root user to always execute commands and allows any member of the "wheel" group to use the su command if they know the root password. In general, it is a good idea to keep the number of users in the wheel group small. Once the additions are made, the /etc/pam.d/su file should look like that listed in Figure 8-3.

```
#%PAM-1.0
auth        required    /lib/security/pam_pwdb.so shadow nullok
auth        required    /lib/security/pam_wheel.so group=root
account     required    /lib/security/pam_pwdb.so
password    required    /lib/security/pam_cracklib.so
password    required    /lib/security/pam_pwdb.so shadow use_authtok nullok
session     required    /lib/security/pam_pwdb.so
session     optional    /lib/security/pam_xauth.so
```

Figure 8-3. Configuration of /etc/pam.d/su

Once the wheel group is added to the /etc/pam.d/su file, add the user accounts needing root-level access to the wheel group:

[root@turtledove /]# usermod –G10 admin write read

The syntax used for adding users to any group is: usermod –G<group1,...,groupN) <username>. The group can be specified by either group name or by the group ID (GID) number. After modifying the /etc/pam.d/su configuration file and adding users needing root access to the wheel group, the substitute user command is restricted and denied to all other users (Mann, n.d.).

With the system configured to restrict access so changes and questionable activity can more easily be identified in a standalone production system, the next step is to secure the network configurations and software services. Once the server is put into production on a network or on the Internet, an entire set of new issues evolves. Extreme care should be taken when configuring any system on a network. What follows in the next chapters are steps to make the server "safer" on a potentially dangerous wire.

CHAPTER 9
NETWORK CONFIGURATION

As a word of caution, placing a server, or computer of any kind, on a network means that server will become vulnerable to an entirely different set of exploits. The best defense to such attacks is to keep computers off of networks altogether. As infeasible as it may be, the only true way to secure the Linux VPN server from network attacks is to isolate it from the network as a whole. Obviously in providing VPN services this is impossible, however, ways exist to make the presence of the server on the network less well known. One certain way to "hide" the server is in the network configuration settings. As a primer, this chapter will discuss basic network configurations and move on to more advanced techniques to thwart commonly known attacks.

```
DEVICE=eth0                 # First ethernet device; use for external net
IPADDR=209.33.129.114       # Static IP address for connection
NETMASK=255.255.255.240     # Subnet mask for external network
GATEWAY=209.33.129.113      # IP address of the external router or modem
ONBOOT=yes                  # Enable the adapter at boot time
```

Figure 9-1. Network configuration of etc/sysconfig/network-scripts/ifcfg-eth0

Network Configuration

To enable the Linux VPN server's networking functions, several configuration files must be set up to allow access to the network hardware and software. To begin with, the ethernet adapters must be set up with static IP addresses and other information needed to communicate using TCP/IP. Edit the /etc/sysconfig/network-scripts/ifcfg-eth0 file to add information about the external network connection similar to that listed in Figure 9-1.

Edit the /etc/sysconfig/network-scripts/ifcfg-eth1 file for the internal network settings to appear similar to the listing in Figure 9-2.

```
DEVICE=eth1                 # Second ethernet device; use for internal net
IPADDR=192.168.6.1          # Static IP address for connection
NETMASK=255.255.255.0       # Subnet mask for internal network
ONBOOT=yes                  # Enable the adapter at boot time
```

Figure 9-2. Network configuration of /etc/sysconfig/network-scripts/ifcfg-eth1

Next, make sure network support is enabled and the server has a unique hostname by editing the /etc/sysconfig/network file like the listing in Figure 9-3.

```
NETWORKING=yes                      # Network support enabled
HOSTNAME=turtledove.destinationearth.net    # Unique server hostname
FORWARD_IPV4=yes                    # Allow forwarding from LAN
to WAN
```

Figure 9-3. Configuration of /etc/sysconfig/network

In addition, edit the /etc/HOSTNAME file to make sure the HOSTNAME variable is set to the correct hostname for the server:

turtledove.destinationearth.net

The HOSTNAME variable will be used in several startup events, so if the hostname is not correct, those startup events may fault or fail.

The Linux server contains a hosts file in the /etc directory. The hosts file contains the IP address to hostname translation of any local hosts not available by DNS. The /etc/hosts file must contain a reference to the local host in order for various programs to function correctly. Make sure the file contains the following entry:

127.0.0.1 locahost.localdomain localhost

This entry refers to the standard IP address of the server and hostname whether or not the server is connected to the network. In addition, the /etc/host.conf file must be configured to provide the resolver service—a basic lookup service for hostnames and IP addresses—

with the correct order in which to lookup host information. Add the entries listed in Figure 9-4 to the /etc/host.conf file.

```
order hosts
multi on
nospoof on
spoofalert on
```

Figure 9-4. Configuration of /etc/host.conf

The order option instructs the resolver service to check the hosts file to resolve any local host to IP address requests. The multi option is set to on so the resolver will return all valid valid addresses for a host that appears in the /etc/hosts file, instead of only the first. The nospoof option enables the resolver service to try and prevent hostname spoofing--after performing a host address lookup, resolver will perform a hostname lookup for that address. If the two hostnames do not match, the query will fail. Finally, the spoofalert option is turned on to log potential spoofing to the system log.

In addition to the /etc/host.conf file, the resolver service uses information stored in the /etc/resolv.conf file to perform lookups of non-local hosts and IP addresses. The /etc/resolv.conf files should contain the internal nameserver IP address followed by all nameserver addresses provided on the Internet Service Provider's network. Refer to Figure 9-5 for a sample configuration.

```
search destinationearth.net      # local domain suffix to search
nameserver 192.168.6.10          # internal DNS
nameserver 204.177.208.1         # ISP primary nameserver
nameserver 204.177.210.39        # ISP secondary nameserver
nameserver ...                   # Any additional namerservers
```

Figure 9-5. Configuration of /etc/resolv.conf

The entries in the /etc/resolv.conf file instruct the resolver to "search" for any hostnames without domain suffixes using the default local domain suffix. If this fails,

then each nameserver is then queried by IP address in listed order until a resolution is made.

Once the network configuration changes are made, issue the following command:

[root@turtledove /]# /etc/rc.d/init.d/network restart

This will restart the network services to allow any changes to take effect. Once the network restarts, verify network connectivity by using the "ping" command to talk to IP addresses and hostnames.

Settings for the /proc/sys Directory

Once the kernel is re-compiled to employ firewall software or to masquerade an interal network from the Internet, a new directory in the /proc directory is created. The /proc/sys directory contains subdirectories in which kernel-tunable paramater files reside. These files are used to alter the state of the running kernel; allowing changes to compiled defaults for certain variables that can alter the system state. The /proc/sys/net/ipv4 directory contains files related to network security kernel options such as ignoring ping requests or filtering rouge packets. Each of these files relates directly to a kernel variable. The /usr/doc/kernel-doc-<version>/sysctl directory contains files describing the valid types and values for the individual variables. Control of the variable values can be managed by adding entries to the /etc/sysctl.conf configuration file. The system control, sysctl, settings are loaded from this file at each boot before the /etc/rc.d/rc.local file is loaded.

In order to combat several known attacks and to provide essential network services for the internal network, several of these parameters need to be set or changed

(Mourani & Madhusudan, 2000). The /etc/sysctl.conf file for the Linux VPN server should contain the parameter specifications listed in Figure 9-6.

```
# Ignore all ICMP(ping)requests
net.ipv4.icmp_ip_echo_ignore_all = 1
# Ignores broadcast requests
net.ipv4.icmp_echo_ignore_broadcasts = 1
# Disable ICMP Redirect Acceptance
net.ipv4.conf.all.accept_redirects = 0
# Disables IP source routing
net.ipv4.conf.all.accept_source_route = 0
# Enables source address verification
# and Enables spoofing protection
net.ipv4.conf.all.rp_filter = 1
# Log spoofed packets, source routed packets, and redirect packets
net.ipv4.all.log_martians = 1
# Disables automatic defragmentation (needed for masquerading, LVS)
net.ipv4.ip_always_defrag = 1
# Disables packet forwarding
net.ipv4.ip_forward = 1
# Enables TCP SYN Cookie Protection
net.ipv4.tcp_syncookies = 1
# Disables the magic-sysrq key
kernel.sysrq = 0
```

Figure 9-6. Configuration of /etc/sysctl.conf

To prevent ping flooding, a basic form of a Denial of Service (DoS) attack where an IP address is flooded with ICMP requests to effectively close down any services provided on that IP address, the net.ipv4.icmp_ip_echo_ignore_all parameter should be set to one so the server will ignore all ICMP (ping) requests. Ignoring an ICMP request is considerably better than simply denying the request—no response from an IP address typically indicates no link at that address, but a denied response is an acknowledgement to the sender that a link does exist.

In addition, set the net.ipv4.icmp_echo_ignore_broadcasts parameter to one to prevent intentional or unintentional ping flooding. When a packet is sent to the network broadcast address, it is sent to all machines on that network. The machines on the

network then respond to that request and the responses can result in network congestion or a DoS attack. Most routers, as a rule, will not forward packets destined for a broadcast address but an exception to that rule most certainly exists. By ignoring broadcast ICMP requests, the Linux server will be less apt to suffer from ping attacks.

When a host uses a non-optimal or stale route to a destination, the routers to inform the host what the correct route should be by returning an ICMP redirect packet. If an attacker is able to forge ICMP redirect packets, he or she may be able to alter the routing table on the host. By altering the host's routing table, the malevolent may gain access to sensitive traffic by diverting traffic via another path. In order to prevent such an attack, set the net.ipv4.conf.all.accept_all_redirects equal to zero.

Like a letter dropped in a mailbox, an IP packet has a destination address and a return address. In Internet communication, the return address is known as the source address where the packet originated. When a packet requires a reply, the reply is sent back to the source address. Often, attackers will create packets with a source address different from their actual IP address. This is known as IP spoofing and it is common to protect the attacker's identity in DoS attacks. The attacker does not require a reply since the real intention is to deny access to any services the destination computer offers. In other cases, the attacker sends a packet with a source address of a trusted internal IP to the external IP of a gateway host. The gateway host, if not prepared, will read the packet as an internal secure communication and may allow the attacker into the system. In order to combat IP spoofing attacks, setting the Linux kernel parameter net.ipv4.conf.all.rp_filter to one enables spoofing protection thereby preventing the server from being the source of spoofed communications (Anonymous, 1998).

IP source routing, where an IP packet contains details about what path to take to its destination, can be potentially problematic. When source routing is enabled, the receiving host is required to reply using the same route as the received packet. This can become a source of attack if a secure host receives a source-routed packet. When the host replies, an attacker can then intercept the reply and make the host believe it is communicating with a trusted host. If this occurs, the security implemented for an insecure connection can be circumvented. To protect the Linux server from this hole, set the net.ipv4.conf.all.accept_source_route to zero. Setting the net.ipv4.conf.all.log_martians parameter to one will send a record of each occurrence of source-routed packets, spoofed-address packets, or redirect requests to the system log.

In order to allow the Linux VPN server to act as a gateway computer that masquerades all internal traffic to the Internet, set the parameter net.ipv4.ip_always_defrag to one. Since not all information sent to the host can always be contained in a single packet, often data is fragmented into multiple packets. The problem with this is that only the first packet contains the port numbers of the communication pipe. It is possible to insert potentially harmful information into the remaining packets. In a normal configuration, fragments are passed through the interface and reassembled later—sometimes with an attacker's code. By enabling IP defragmentation, the packets are reassembled and put into the correct order, discarding any extraneous packets that may have been added, and then passed through the interface.

The net.ipv4.ip_forward parameter should be set to one to enable IP masquerading of Internet requests from the internal network. This protects computers on the internal network so all requests appear to be coming from one and only one host. In

additions, all computers on the internal network have access to Internet resources without becoming vulnerable to attack.

A more serious DoS attack than ping flooding is the SYN attack. Traditionally, systems would employ a SYN-ACK handshake to confirm a TCP connection and move it from the waiting queue. If the SYN-ACK handshake does not complete, an attacker makes a request but ignores the returned packet with the SYN bit set, the server would wait for a long period for the ACK packet that never comes. While waiting for the packet with the ACK bit set, the server would not move the request from the wait queue until a timeout occurs and would allow no one else to connect. An attacker could take advantage of this by continuing to make requests and thereby denying anyone else access to the server. To protect the server, the net.ipv4.tcp_syncookies parameter should be set to one so a SYN attack cannot take place (Bandel, 2000)

Finally, the SysRQ key sequence is a “magic” key combo that the kernel will always respond to regardless of what else it is doing. This can be dangerous if someone can gain access to a terminal session or physical access to the server. The SysRQ sequence combined with a key indicating a command can force shutdown the system without unmounting the file systems and possible causing disk corruption among other things. It is best to disable this option by setting the kernel.sysrq parameter to zero.

Once a good level of secure network services is implemented on the Linux VPN server, it is time to address what services the server should offer, and how those services will interact in a production VPN system. The next chapter introduces several common network services provided in production environments, and offers advice about which

services to provide. Overall, security is still in the forefront of considerations in the design of the VPN system and will be addressed with each offered service.

CHAPTER 10
DAEMON SECURITY

This chapter identifies basic services the VPN server should provide, and also offers solutions to problems that may arise in offering such services. Further, a means for remotely accessing the server will be provided in as secure a manner as possible. Most network environments take advantage of dynamic IP addressing, name resolution services, and remote logins. We begin by discussing the Dynamic Host Configuration Protocol.

```
subnet 192.168.6.0 netmask 255.255.255.0 {          # LAN network and mask
        range 192.168.6.128 192.168.6.254;          # Range of IP addresses
        default-lease-time 86400;                   # Default address lease sec.
        max-lease-time 86400;                       # Maximum lease time sec.
        option routers 192.168.6.1;                 # Default Gateway
        option broadcast-address 192.168.6.255;
        option subnet-mask 255.255.255.0;
        option domain-name-servers 24.221.30.3, 24.221.30.4, 192.168.6.1
        option netbios-name-servers 192.168.1.10, 192.168.1.1;
        option netbios-dd-server 192.168.6.1;
        option netbios-node-type 8;
        option netbios-scope "";
}
```

Figure 10.1. Configuration of /etc/dhcpd.conf

Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol, dhcp, daemon package will allow LAN clients to get their network configuration information from the Linux VPN server. The dhcp package will provide IP addresses, subnetmasks, gateway addresses, and DNS servers to the computers on the LAN. DHCP makes it easier to administer a large network since changes to IP addresses and other configuration data will be propagated to

each workstation without having to "touch" each individual machine. In addition, DHCP will provide information about DNS servers on the other side of the VPN tunnel. Edit the /etc/dhcpd.conf file so it appears similar to listing in Figure 10-1.

Once the server is connected to the LAN and the DHCP server is started, the Linux server will begin handing out IP addresses and other configuration data to all clients on the LAN configured to use DHCP.

Securing Name Resolution Services

As an option, the Linux server can offer hostname to IP address lookups for the internal network. However, a stand-alone DNS server is strongly recommended instead. General configuration of the Berkley Internet Nameserver Daemon (BIND) named daemon will be left for the reader. Help is available from the "DNS HOW TO" by Nicolai Langfeldt and Jamie Norrish, et al at http://www.linuxdoc.org/HOWTO/DNS-HOWTO.html.

For the Linux VPN server to securely offer Name Resolution services on the internal network, the named daemon should be run from a secure partition. Historically, the complex BIND server has been the target of many well-documented exploits to gain root access resulting from bugs in the software. A good security measure is to isolate the named daemon in its own area and not run it as the root user. If a new security exploit comes out and an attacker gains access through the exploit, she will be limited in where she can go and what she can do. (Welte, 2001)

This Linux implementation will only offer DNS services to the internal network, and will forward any requests for names not on the local network to the Internet Service Provider's primary or secondary name servers.

If an appropriate user does not already exist for the named daemon, create one by adding the following line to the /etc/passwd file:

named:x:200:200:Nameserver:/chroot/named:/bin/false

In addition, add a group for the named daemon in the /etc/group file:

named:x:200

In these commands the UID and GID are set to 200; they could easily be set to any available ID number. Notice the shell for the new named user is set to /bin/false since the user will never need to login.

During the initial installation of the Linux VPN server, a separate partition /chroot was created for the chroot jail. The jail name means to change the daemon from running as root to a normal user and separate the daemon from other areas of the system. On the /chroot partition set up a directory structure like the one in Figure 10-2.

```
/chroot
|_int_dns
        |_dev
        |_etc
        |_lib
        |_usr
            |_sbin
        |_var
            |_named
            |_run
```

Figure 10-2. Directory structure of /chroot

Change to the directory /usr/src/bind-9.1.3 that was created when the update packages were downloaded and unzipped and issue the following command to run the configure script:

```
[root@turtledove bind-9.1.3]# ./configure --prefix= --exec-prefix=/usr \
--datadir=/usr/share --includedir=/usr/include --infodir=/usr/share/info \
--mandir=/usr/share/man --disable-threads
```

An explanation of the various options is available in the configure script itself. The options specified will set up the correct compile options for a 2.2.x kernel. Next, compile the named daemon:

```
[root@turtledove bind-9.1.3]# make
```

Once the system has finished compiling the BIND package, install the new version:

```
[root@turtledove bind-9.1.3]# make install
```

Next copy the documentation files to appropriate location. Change into the /usr/src/bind-9.1.3/doc/man/bin directory and copy the man pages as shown below:

```
[root@turtledove bin]#  cp *.1 /usr/share/man/man1/
[root@turtledove bin]# cp *.5 /usr/share/man/man5/
[root@turtledove bin]# cp *.8 /usr/share/man/man8/
```

Recursively change the permissions on the int_dns directory so only the owner has full permissions and the group has read and execute permissions:

```
[root@turtledove /]# chmod –R 750 /chroot/int_dns
```

Set up a null block device and a time keeper for use by the named daemon by using the following command in the int_dns directory:

```
[root@turtledove int_dns]# mknod –m 666 dev/null c 1 3
[root@turtledove int_dns]# cp /etc/localtime /chroot/int_dns/etc/
```

Finally, copy all the required libraries and executable files for the named daemon to run into the int_dns directory:

```
[root@turtledove int_dns]# cp /usr/sbin/named* /chroot/int_dns/usr/sbin
[root@turtledove int_dns]# cp /lib/libc.so.6 /chroot/int_dns/lib
[root@turtledove int_dns]# cp /lib/ld-linux.so.2 /chroot/int_dns/lib
```

Change the permissions on the files in the /int_dns/usr/sbin directory to the same permissions as the /int_dns directory and then remove the named files from the /usr/sbin directory:

```
[root@turtledove int_dns]# chmod 750 /chroot/int_dns/usr/sbin/named*
[root@turtledove int_dns]# rm –f /usr/sbin/named*
```

Lastly, copy any additional necessary configuration files such as the named.conf file to the /chroot/int_dns/etc directory and change the owner and group settings for the entire /chroot/int_dns directory:

```
[root@turtledove int_dns]# chown -R named:named /chroot/int_dns
```

```
#!/bin/sh
# named      This shell script takes care of starting and stopping
#            named (BIND DNS server).
#
# chkconfig: - 55 45
# description: named (BIND) is a Domain Name Server (DNS) \
# that is used to resolve host names to IP addresses.
# probe: true
# Source function library.
. /etc/rc.d/init.d/functions
# Source networking configuration.
. /etc/sysconfig/network
# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0
[ -f /chroot/int_dns/usr/sbin/named ] || exit 0
[ -f /chroot/int_dns/etc/named.conf ] || exit 0
RETVAL=0
# See how we were called.
case "$1" in
  start)
        # Start daemons.
        echo -n "Starting named: "
        daemon /chroot/int_dns/usr/sbin/named -u named –t /chroot/int_dns
                        –c /chroot/int_dns/etc/named.conf
                RETVAL=$?
                [ $RETVAL -eq 0 ] && touch /var/lock/subsys/named
  echo
        ;;
  stop)
```

Figure 10-3. Configuration of /etc/rc.d/init.d/named

The initialization script for the named daemon, /etc/rc.d/init.d/named, should be modified to point to the new location and to start the daemon as the correct user (Mourani

& Madhusudan, 2000). An example named script with changes highlighted is listed in Figure 10-3.

```
        # Stop daemons.
        echo -n "Shutting down named: "
            killproc named
                RETVAL=$?
                [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/named
            echo
            ;;
        status)
                /usr/sbin/ndc status
                exit $?
                ;;
        restart)
                $0 stop
                $0 start
                ;;
        reload)
                /usr/sbin/ndc reload
                exit $?
                ;;
        probe)
                # named knows how to reload intelligently; we don't want linuxconf
                # to offer to restart every time
                /usr/sbin/ndc reload >/dev/null 2>&1 || echo start
                exit 0
                ;;

        *)
            echo "Usage: named {start|stop|status|restart}"
            exit 1
    esac
    exit $RETVAL
```

Configuration of /etc/rc.d/init.d/named continued

To make certain all DNS activities are logged to the system log, change the line starting the daemon syslogd to read:

```
daemon syslogd -a /chroot/int_dns/dev/log -m 0
```

One last step can be taken to protect the server from spoofing. Modify the /chroot/int_dns/etc/named.conf file to disable any queries for domains not owned by or

not covered within the scope of the Linux VPN server. In addition, remove any non-local IP addresses or network addresses from the "allow-recursion" option to decrease the risk of cache poisoning attacks where false data is fed to the DNS server to capture information intended for another host. The best step for reducing attacks through exploits in BIND is to not use BIND on any host attached to the Internet. Choose well-known nameservers to forward lookup requests to, and always update the root.hints file to contain the most current data.

```
#
# inetd.conf    This file describes the services that will be available
#               through the INETD TCP/IP super server.  To re-configure
#               the running INETD process, edit this file, then send the
#               INETD process a SIGHUP signal.
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Echo, discard, daytime, and chargen are used primarily for testing.
#
# To re-read this file after changes, just do a 'killall -HUP inetd'
#
#echo           stream tcp     nowait root    internal
#echo           dgram  udp     wait    root    internal
#discard        stream tcp     nowait root    internal
#discard        dgram  udp     wait    root    internal
#daytime        stream tcp     nowait root    internal
#daytime        dgram  udp     wait    root    internal
#chargen        stream tcp     nowait root    internal
#chargen        dgram  udp     wait    root    internal
#time           stream tcp     nowait root    internal
#time           dgram  udp     wait    root    internal
#
# These are standard services.
#
ssh             stream tcp     nowait root    /usr/sbin/tcpd in.sshd
#ftp            stream tcp     nowait root    /usr/sbin/tcpd  in.ftpd -l –a
#telnet         stream tcp     nowait root    /usr/sbin/tcpd  in.telnetd
```

Figure 10.4. Configuration of /etc/inetd.conf

```
# Shell, login, exec, comsat and talk are BSD protocols.
#shell  stream tcp     nowait root            /usr/sbin/tcpd  in.rshd
#login  stream tcp     nowait root            /usr/sbin/tcpd  in.rlogind
#exec   stream tcp     nowait root            /usr/sbin/tcpd  in.rexecd
#comsat         dgram  udp    wait    root            /usr/sbin/tcpd  in.comsat
#talk   dgram  udp     wait    nobody.tty     /usr/sbin/tcpd  in.talkd
#ntalk  dgram  udp     wait    nobody.tty     /usr/sbin/tcpd  in.ntalkd
#dtalk  stream tcp     wait    nobody.tty     /usr/sbin/tcpd  in.dtalkd
#
# Pop and imap mail services et al
#
#pop-2 stream  tcp    nowait  root    /usr/sbin/tcpd  ipop2d
#pop-3 stream  tcp    nowait  root    /usr/sbin/tcpd  ipop3d
#imap  stream  tcp    nowait  root    /usr/sbin/tcpd  imapd
#
# The Internet UUCP service.
#
#uucp  stream tcp      nowait uucp            /usr/sbin/tcpd  /usr/lib/uucp/uucico    -l
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers." Do not uncomment
# this unless you *need* it.
#
#tftp   dgram  udp     wait    root           /usr/sbin/tcpd  in.tftpd
#bootps         dgram  udp    wait    root           /usr/sbin/tcpd  bootpd
#
# Finger, systat and netstat give out user information which may be
# valuable to potential "system crackers."  Many sites choose to disable
# some or all of these services to improve security.
#
#finger stream tcp      nowait nobody          /usr/sbin/tcpd  in.fingerd
#cfinger stream         tcp    nowait root     /usr/sbin/tcpd  in.cfingerd
#systat stream tcp      nowait guest   /usr/sbin/tcpd  /bin/ps -auwwx
#netstat        stream tcp     nowait guest   /usr/sbin/tcpd  /bin/netstat      -f inet
#
# Authentication
#
# identd is run standalone now
#
#auth   stream tcp      wait    root    /usr/sbin/in.identd in.identd -e -o
#
# End of inetd.conf
#linuxconf  stream tcp wait root /bin/linuxconf linuxconf --http
#swat     stream  tcp    nowait.400      root /usr/sbin/swat swat
```

Configuration of /etc/inetd.conf continued

The inetd Daemon

The “super server”, inetd, will load a network program based upon a request from the network. The /etc/inetd.conf file configures what network programs can be requested and loaded across the network. Each entry tells the inetd daemon what services will listen and accept requests. The /etc/services file lists all possibly available services and the port on which each service listens. Before connecting the Linux server to the network, disable unnecessary services to lock down ports that would normally allow connections for those services. The services can be disabled by finding the configuration line in the /etc/inetd.conf refering to the service and inserting a ‘#’ to comment out the line. A sample inetd.conf file for a Linux VPN server is provided in Figure 10-4.

Notice, the only service available on the Linux VPN server is ssh. The ssh service will provide secured ftp and telnet access to the server so usernames and passwords are not transmitted across the network in plain text.

Once the configuration changes are made to the /etc/inetd.conf file, ensure the owner is root, and if not, use the chown command to make root the file owner. In addition, the permissions on the file should allow only root read and write privileges (600). The final step in configuring the inetd daemon is to activate the changes. Use the following syntax to restart the inetd daemon:

```
[root@turtledove /]# killall –HUP inetd
```

When the inetd daemon has restarted and activated the changes from the /etc/inetd.conf file, any request to an offered service will be sent to the tcpd daemon instead of the actual service’s daemon. By configuring the inetd daemon to send requests to the tcpd, tcp wrappers are employed to secure access to all available network servers.

TCP Wrappers

TCP wrappers is a program that will "wrap" a tcp listener (a service listening on a TCP port) for a service server, such as ftp or telnet, to help protect the Linux server from exploits via that server. TCP-based servers started via the Internet super server daemon, inetd, typically use TCP wrappers. Any TCP server available and listening through the inetd daemon will call the TCP wrapper daemon, tcpd, instead of the actual server's program. Once the requests are passed to the TCP wrapper, the wrapper can then allow or deny access to the original program.

The TCP wrappers daemon is controlled by the /etc/hosts.allow and /etc/hosts.deny files. Each of the files contain entries in the following format:

daemon(s) : client(s) : option : option ...

For each entry the first column of the entry specifies which daemon(s) to protect; the second column of the entry specifies the client(s) to protect by hostname, domain name, or IP address; the final columns are reserved for options that may be configurations or actions to take.

When a request is sent to the TCP wrapper daemon, access to the server will be granted when a daemon, client pair matches an entry in the /etc/hosts.allow file. Access to the server will be denied when the pair matches an entry in the /etc/hosts.deny file. If the pair does not match entries in either file, access is granted.

For the purpose of the Linux VPN server, no options will be specified since the enabled TCP servers do not require options. Begin by editing the /etc/hosts.deny file to

```
# Deny access to everyone
ALL:ALL@ALL
```

Figure 10-5. Configuration of /etc/hosts.deny

contain the lines listed in Figure 10-5. The line entry ALL:ALL@ALL translates to all services and all locations; access to any service will be denied to anyone not listed in the /etc/hosts.allow file. Edit the /etc/hosts/allow to contain the lines listed in Figure10-6.

```
# Allow all access from the administrative workstation
ALL:192.168.6.2              # the IP address of the interal admin computer
sshd:<remote vpn server>     # for access through ftp and secure telnet
```

Figure 10-6. Configuration of /etc/hosts.allow

The Linux VPN server will now allow access to all servers from an IP address on the internal network, and will allow secure ftp and secure telnet access from any remote vpn server. No other hosts or IP addresses will have access to any of the TCP servers offered by the VPN server. To check for configuration errors, execute the tcpdchk utility:

```
[root@turtledove /]# tcpdchk
```

The tcpdchk utility checks for syntax errors in the /etc/hosts.allow and /etc/hosts.deny files and will report any potential and real problems it can find.

Secure Shell Configuration

Ssh (Secure Shell) is a program that allows remote logins to computers over a network. Once logged in, the ssh suite of utilities can also execute commands in a remote machine and move files from one machine to another. A secure shell provides strong authentication and allows for encrypted secure communications over unsecure channels such as the Internet. It is intended as a replacement for remote access programs such as rlogin, rsh, and rcp, which traditionally pass information in clear text. When information such as passwords and usernames are transmitted in clear text, anyone with the ability to sniff packets along the connection used can get the passwords, usernames, and any other information transmitted. Ssh can also provide secure access to normally insecure

programs such as telnet, ftp, and pop3 mail. Additionally, ssh provides secure forwarding of arbitrary TCP connections.

The traditional Berkley Software Designs (BSD) 'r' - commmands (rsh, rlogin, rcp) and telnet and ftp programs are vulnerable to different kinds of attacks because of the use of unencrypted transmissions. Ssh, on the other hand, never sends any information in clear text. Once the ssh protocol establishes a TCP connection with a host providing ssh services, the ssh protocol immediately exchanges public keys to create an encrypted tunnel. Once the tunnel is established a user can perform remote logins and other remote functions while all information is protected within the encrypted tunnel.

In general, ssh can protect against IP spoofing, IP source routing, DNS spoofing, sniffing of cleartext passwords and other data or manipulation of such data by hosts with access to the same wire, and attacks based on listening to X authentication data and spoofed connection to the X11 server. In other words, ssh never trusts the network; a hostile who has taken over the network can only force ssh to disconnect, but cannot decrypted, play back the traffic, or hijack the connection.

The OpenSSH package from OpenBSD provides, in addition to other programs, a secure shell server. The OpenSSH package supports both SSH protocol version 1 and SSH protocol version 2. According to the OpenBSD manual page for sshd, the secure shell daemon, for the server side of secure shell communications, the two protocols differ in their key negotiation and encryption schemes.

> In SSH protocol version 1 “[e]ach host has a host-specific RSA key (normally 1024 bits) used to identify the host. Additionally, when the daemon starts, it generates a server RSA key (normally 768 bits). This key is normally regenerated every hour if it has been used, and is never stored on disk. Whenever a client connects, the daemon responds with its public host and server keys. The client compares the RSA host key against its

> own database to verify that it has not changed. The client then generates a 256 bit random number. It encrypts this random number using both the host key and the server key, and sends the encrypted number to the server. Both sides then use this random number as a session key which is used to encrypt all further communications in the session. The rest of the session is encrypted using a conventional cipher, currently Blowfish or 3DES, with 3DES being used by default. The client selects the encryption algorithm to use from those offered by the server. Next, the server and the client enter an authentication dialog. The client tries to authenticate itself using .rhosts authentication, .rhosts authentication combined with RSA host authentication, RSA challenge-response authentication, or password based authentication. Rhosts authentication is normally disabled because it is fundamentally insecure…"
>
> In SSH protocol version 2 "[e]ach host has a host-specific key (RSA or DSA) used to identify the host. However, when the daemon starts, it does not generate a server key. Forward security is provided through a Diffie-Hellman key agreement. This key agreement results in a shared session key. The rest of the session is encrypted using a symmetric cipher, currently 128 bit AES, Blowfish, 3DES, CAST128, Arcfour, 192 bit AES, or 256 bit AES. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (hmac-sha1 or hmac-md5). Protocol version 2 provides a public key based user (PubkeyAuthentica- tion) or client host (HostbasedAuthentication) authentication method, conventional password authentication and challenge response based methods. " (Tevesk, 2001)

During the install of the OpenSSH packages, the installer will have created the ssh client and server configuration files in the /etc/ssh directory. The /etc/ssh/sshd_config file is the system-wide configuration file for the OpenSSH secure shell server. The secure shell daemon reads configuration data from this file at startup. Edit the /etc/ssh/sshd_config file to appear similar to the one listed in Figure10-7.

The Port option specifies the port number on which sshd listens. The default port is 22 and is listed that way in most 'services' files. The Protocol option specifies that sshd will support both protocol 1 and protocol 2. The ListenAddress option specifies the local addresses sshd should listen on. In this implementation, sshd will listen on both the external and internal network interfaces. The HostKey option specifies the file containing

```
# This is ssh server systemwide configuration file.

Port 22                                 # Listen for connections on port 22
Protocol 2,1                            # Support SSH protocols 1 and 2
ListenAddress 24.231.21.43              # Server's External IP address
ListenAddress 192.168.6.1               # Server's Internal IP address
HostKey /etc/ssh/ssh_host_key           # File containing private host keys
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
ServerKeyBits 1024                      # Number of bits in server key
                                        # for SSH protocol 1
LoginGraceTime 90                       # Timeout in seconds if nologin
KeyRegenerationInterval 3600            # SSH protocol 1 server key
                                        # regeneration in x seconds
PermitRootLogin no                      # Deny root user login
IgnoreRhosts yes                        # Don't read.rhosts and .shosts files
IgnoreUserKnownHosts yes                # don't trust
# ~/.ssh/known_hosts for
# RhostsRSAAuthentication
StrictModes yes                         # Check file modes and ownership
X11Forwarding no                        # Do not allow X11 forwarding

KeepAlive yes                           # Send keepalive messages
SyslogFacility AUTH                     # Log messages with AUTH code
LogLevel INFO                           # Level of log messages
RhostsAuthentication no                 # Don't authenticate using .rhosts or
# /etc/hosts.equiv
RhostsRSAAuthentication no              # Don't use rhosts or
# /etc/hosts.equiv together with
# RSA authentication
RSAAuthentication yes                   # Use RSA Authentication
PasswordAuthentication yes              # Allow password authentication
PermitEmptyPasswords no                 # Don't allow empty passwords
AllowUsers admin read write             # allow only these users
```

Figure 10-7. Configuration of /etc/ssh/sshd_config

the private host keys (default /etc/ssh_host_key) used by SSH protocol versions 1 and 2. Note that sshd will refuse to use a file if it is group/world-accessible, so the permissions for the files specified should be 700. The Linux server implementation will support typical host keys as well as RSA and DSA generated keys. The ServerKeyBits option defines the number of bits in the ephemeral protocol version 1 server key. The minimum

value is 512 and here it is doubled create a longer key. The LoginGraceTime is the time in seconds allowed for a successful login. After this time if the user has not successfully logged in, the server will disconnect and terminate the session. The KeyRegeneration-Interval is used only in protocol version 1. If the ephemeral server key has been used, it is automatically regenerated after the specified number of seconds. The purpose of regeneration is to prevent decrypting captured sessions by later breaking into the machine and stealing the keys. In addition, the key is never written to a file. The PermitRootLogin option specifies whether the root user can login. If this option is set to “no” root is not allowed to login. IgnoreRhosts set to yes specifies that .rhosts and .shosts files will not be used in RhostsAuthentication, RhostsRSAAuthentication or HostbasedAuthentication. However, the files /etc/hosts.equiv and /etc/shosts.equiv are still used. Setting the IgnoreUserKnownHosts to “yes” tells the secure shell daemon to ignore the user's $HOME/.ssh/known_hosts during RhostsRSAAuthentication or Hostbased-Authentication. The strict StrictModes option specifies whether sshd should check file modes and ownership of the user's files and home directory before accepting login. This is a good idea since novices sometimes accidentally leave their directory or files world-writable, and, as a reminder, the sshd daemon will not open a connection if the files are world-readable or writable. The X11Forwarding option is disabled since this implementation will not use any X11 programs. Enabling the KeepAlive option allows the system to send keepalive messages to the other side so the death of the connection or crash of one of the machines will be properly noticed. If keepalives are not sent, sessions may hang indefinitely on the server and unecessarily consume server resources. The sshd daemon can log messages to the system log when a login occurs. Setting the

SyslogFacility option to AUTH enables this function. The LogLevel option gives the verbosity level that is used when logging messages from sshd. Disabling the RhostsAuthentication option does not allow authentication using rhosts or /etc/hosts.equiv files only. This is a good idea, since these files are notoriously insecure. Turning off the RhostsRSAAuthentication option does not allow authentication using rhosts or /etc/hosts.equiv authentication together with successful RSA host authentication. Again, the files are not considered secure. The RSAAuthentication option enables using pure RSA authentication for SSH protocol one. The Password-Authentication option allows password authentication, however keep in mind, even though the tunnel is encrypted, the passwords are sent through the tunnel in clear text form. Do not enable PermitEmptyPasswords. When password authentication is allowed, the PermitEmptyPasswords option set to no allows login to accounts with empty password strings. The AllowUsers keyword is followed by a list of user names, separated by spaces, that can login to the system through ssh. In this implementation, the list of users is crucial. No other users should be allowed remote access to this host.

The OpenSSH package also comes with its own secure shell client. In the Linux VPN implementation, the ssh client will be used only in scripted sessions to dynamically configure VPN connection. Other clients such as TeraTerm are available for download for the system administrator's workstation. The OpenSSH client supports both SSH protocols, and the manual page describes the implementations in detail:

> In SSH protocol version 1 "if the machine the user logs in from is listed in /etc/hosts.equiv or /etc/shosts.equiv on the remote machine, and the user names are the same on both sides, the user is immediately permitted to log in. Second, if .rhosts or .shosts exists in the user's home directory on the remote machine and contains a line containing the name of the client machine and the name of the user on that machine, the user is permitted to

log in. This form of authentication alone is normally not allowed by the server because it is not secure. The second authentication method is the rhosts or hosts.equiv method combined with RSA-based host authentication. It means that if the login would be permitted by $HOME/.rhosts, $HOME/.shosts, /etc/hosts.equiv, or /etc/shosts.equiv, and if additionally the server can verify the client's host key (see /etc/ssh_known_hosts and $HOME/.ssh/known_hosts in the FILES section), only then login is permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing and routing spoofing. [Note to the administrator: /etc/hosts.equiv, $HOME/.rhosts, and the rlogin/rsh protocol in general, are inherently insecure and should be disabled if security is desired.] As a third authentication method, ssh supports RSA based authentication. The scheme is based on public-key cryptography: there are cryptosystems where encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key. RSA is one such system. The idea is that each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key. The file $HOME/.ssh/authorized_keys lists the public keys that are permitted for logging in. When the user logs in, the ssh program tells the server which key pair it would like to use for authentication. The server checks if this key is permitted, and if so, sends the user (actually the ssh program running on behalf of the user) a challenge, a random number, encrypted by the user's public key. The challenge can only be decrypted using the proper private key. The user's client then decrypts the challenge using the private key, proving that he/she knows the private key but without disclosing it to the server. Ssh implements the RSA authentication protocol automatically… If other authentication methods fail, ssh prompts the user for a pass-word. The password is sent to the remote host for checking; however, since all communications are encrypted, the password cannot be seen by someone listening on the network."

In SSH protocol version 2 "[w]hen a user connects using the protocol version 2 different authentication methods are available. Using the default values for PreferredAuthentications, the client will try to authenticate first using the hostbased method; if this method fails public key authentication is attempted, and finally if this method fails keyboard-interactive and password authentication are tried. The public key method is similar to RSA authentication described in the previous section and allows the RSA or DSA algorithm to be used…The session identifier is derived from a shared Diffie-Hellman value and is only known to the client and the server. If public key authentication fails or is not available a password can be sent encrypted to the remote host for proving the user's identity. Additionally, ssh supports hostbased or challenge response authentication. Protocol 2 provides additional mechanisms for confidentiality (the traf- fic is encrypted using 3DES, Blowfish, CAST128 or Arcfour) and integrity

> (hmac-md5, hmac-sha1). Note that protocol 1 lacks a strong mechanism for ensuring the integrity of the connection. "(Ylonen, 2001)

Since all ssh activity initiating from the Linux VPN activity will be initiated from the server itself, a system-wide configuration should be setup. Edit the /etc/ssh/ssh_config to appear as in Figure 10-8.

```
# $OpenBSD: ssh_config,v 1.9 2001/03/10 12:53:51 deraadt Exp $
# This is ssh client systemwide configuration file.  See ssh(1) for more
# information.  This file provides defaults for users, and the values can
# be changed in per-user configuration files or on the command line.
# Configuration data is parsed as follows:
#  1. command line options
#  2. user-specific file
#  3. system-wide file
# Any configuration value is only changed the first time it is set.
# Thus, host-specific definitions should be at the beginning of the
# configuration file, and defaults at the end.
# Site-wide defaults for various options

Host *
Port 22
Protocol 2,1
ConnectionAttempts 3
KeepAlive yes
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication yes
FallBackToRsh no
UseRsh no
BatchMode no
CheckHostIP yes
StrictHostKeyChecking no
```

Figure 10-8. Configuration of /etc/ssh/ssh_config

Most of the options specified in the /etc/ssh/ssh_config file are similar to the options used in the /etc/ssh/sshd_config file. Refer to the previous section for information about those options.

Several of the options are specific only to the ssh client. The ConnectionAttempts option specifies the number of attempts made to connect to a secure shell server before giving up. The FallBackToRsh and UseRsh options are disabled since the protocols are not installed in this implementation. In addition, the rsh protocol is inherently insecure and should not be used even if installed on the system. If the BatchMode option is set to yes password querying will be disabled. While this can be useful when scripting ssh sessions, if someone were to gain access to usernames on the system, access can be gained without a password. With the CheckHostIp option set to "yes", ssh will check the host IP address in the known_hosts file in order to detect if a host key changed due to DNS spoofing. Finally, the StrictHostKeyChecking option set to no enables automatic adding of new host keys to the known_host file.

```
#%PAM-1.0
auth        required    /lib/security/pam_pwdb.so shadow nodelay
auth        required    /lib/security/pam_nologin.so
account     required    /lib/security/pam_pwdb.so
password    required    /lib/security/pam_cracklib.so
password    required    /lib/security/pam_pwdb.so shadow nullok use_authtok
session     required    /lib/security/pam_pwdb.so
session     required    /lib/security/pam_limits.so
```

Figure 10-9. Configuration of /etc/pam.d/sshd

The OpenSSH package also provides a secure file transfer program, sftp. The sftp file will be used to transfer configuration files securely between VPN servers to enable dynamic configurations when changes need to be made.

After configuring the secure shell client and server, the Pluggable Authentication Module needs to know how to handle authentication for the secure shell daemon. Edit the /etc/pam.d/sshd file to include the configuration entries listed in Figure 10-9. For

information about the entries listed see the section on Pluggable Authentication Modules in Chapter 8.

Lastly, “wrap” the sshd daemon so that the TCP wrappers program will start and stop the OpenSSH server. Edit the inetd.conf file by commenting out both the telnet and ftp configuration lines. Add the following line:

```
ssh             stream tcp      nowait root     /usr/sbin/tcpd  sshd –i
```

This entry will allow the super server, inetd, to listen for connections for the sshd server. Note, the –i option is vital because it signals sshd that it is being run from the inetd server.

With the basic services identified and implemented, the remaining chapters of this document will deal directly with establishing secure mechanisms for remote management of the Linux system, and will detail the design of the VPN functionality of the system. Finally, the server will be locked down using a packet firewall and by removing unnecessary binaries from the production system.

CHAPTER 11
SECURE SOCKETS LAYER

OpenSSL

The secure socket layer, SSL, is a presentation layer service located between the transport and application layers of the operating system. It is platform and application independent. As such, SSL is responsible for the management of a secure communications channel between clients employing such applications as ftp, http, and mail services, and a secure server providing those services. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client and vice-versa. In addition, the secure socket layer allows for the establishment of an encrypted connection between a client and a server across an insecure medium.

SSL provides a strong cryptography mechanism for encrypting data transferred between a client and a server. The SSL protocol uses a combination of asymmetric public-key and symmetric key encryption. A symmetric key encryption system is when both the client and server use the same key for encryption and decryption. Symmetric key encryption is much faster than public-key encryption since only one key is used, but public-key encryption provides better authentication techniques, hence the reason for using a combination of both. Public Key Infrastructure, PKI, is a system to verify and authenticate the validity of each party involved in an Internet transaction. PKI is an asymmetric key system that consists of a pair of keys consisting of a public key and a private key. The public key is sent out to clients and the private key stays local to the

server and is never made public. Data that is encrypted with the public key can be decrypted only with the private key, and data encrypted with the private key can be decrypted only with the public key. Whitfield Diffie and Martin Hellman invented public key cryptography in 1976, and for this reason, it is sometimes called Diffie-Hellman encryption.

How PKI Works

In an Internet transaction, the client's Internet browser, or other Internet application, randomly creates a public and private key pair for the SSL session. The client uses the generated private key to encrypt a message, thereby providing for a means of source authentication of the message. The encrypted message is then encrypted with the server's public key to provide confidentiality since only the server is able to do the initial decryption of the message using its private key. The server then uses the client's public key to decrypt the encrypted message. Only the client has access to its private key, and therefore, the server is assured that the encrypted message was actually sent from the client.

A message digest is also applied at the time of the initial encryption of the message. The message digest is used to verify no one has tampered with the message's contents. To create a message digest, the sender applies a hash function, the part of the private key known as the fingerprint, to the message. The digest is appended to the original message, thereby providing a message signature. No matter how long the message, the signature's length is constant. Making any change in the message will change the length of the signature and will provide proof of the changes in the message (OpenSSL core team, n.d.).

How SSL Works

The SSL protocol includes the SSL record protocol and the SSL handshake protocol. The SSL record protocol defines the format, such as TCP, used to transmit data. During the establishment of an SSL connection, the SSL handshake protocol uses the SSL record protocol to exchange a series of messages between an SSL-enabled server and an SSL-enabled client.

An SSL session always begins with an exchange of messages called the SSL handshake. The handshake allows the server to authenticate itself to the client using public-key techniques, and then allows the client and the server to cooperate in the creation of symmetric keys used for rapid encryption, decryption, and tamper detection during the session that follows.

During the handshake certificates in the form of asymmetric keys are exchanged between the client and server. The server then sends its public key to the client; if the server requires client authentication via a certificate, the client will send its public key to the server. The dates on the certificates are verified as valid and then are checked for a digital signature of a trusted certificate authority. If either check fails, the Internet application will issue a warning explaining the failure to the user. At this point, it is up to the user whether to trust the server certificate.

Next, the client generates a random symmetric key to use for encryption. The random symmetric key is encrypted using the server's public key and sent to the server. Once, decrypted by the server, the new symmetric key is used for encrypting the data sent between the client and server. In addition, message encryption algorithm and a hash function are negotiated for connection integrity purposes. This negotiation process could

be carried out on either the client or server side. For additional security, the server can require certain encryption algorithms the client must use. (Netscape Communications Corporation, 1999). Apendix B lists the possible cipher specifications as defined in the OpenSSL protocol suite.

Server Certificates

Typically, the certificates provided during the SSL handshake protocol are issued and digitally signed by a well-known, trusted certificate authority such as Verisign. In this Linux VPN implementation, SSL services will be provided only on the LAN side, so the Linux server will also act as a certificate authority. To set up the Linux server to not require a Certificate Authority, CA, use the OpenSSL program to create a private key and a "self-signed" certificate.

Create a private key using RSA authentication using the following command:

```
[root@turtledove /]# openssl genrsa –out /etc/ssl/keys/serverrsa.key 1024
```

Optionally, create a private key using the triple des encryption standard in addition to the rsa authentication by using the following command:

```
[root@turtledove /]# openssl genrsa -des3 -out /etc/ssl/keys/serversrsa.key
1024
```

If choosing to use triple des encyption, a password must be supplied during the key generation. In addition, the password is required each time the SSL server starts. To avoid manually entering the password each time the server starts, create a file with the first line containing only the password. Secure the file permissions on the file so only the root user and the wheel group can read the file. Point the server to this file when the server is started.

Next, create a Certificate Signing Request (CSR) to obtain a signed certificate. The purpose is to send the certificate authority enough information to create the certificate without sending the entire private key or compromising any sensitive information. Use the following command to generate the CSR:

```
[root@turtledove /]# openssl req -new -key /etc/ssl/keys/serverrsa.key –out \
                     /etc/ssl/keys/servercsr.csr
```

Sign the certificate by creating a self-signed certificate using the following command:

```
[root@turtledove /]# openssl req –new –key /etc/ssl/keys/serverrsa.key –x509 \
                     –days 1095 –out /etc/ssl/keys/servercrt.crt
```

Now secure these keys and certificates by changing the permissions so only the root user can read or write the files. The permissions should be set to 600, giving root read and write permissions.

Using SSL on the LAN side of the server will help to protect the server from attacks initiating from the internal network. In addition, it will provide a secure means for the system administrator to interact with the Linux server (Ng, 2001).

CHAPTER 12
APACHE WEB SERVICE

Apache is a very common, powerful, efficient and freely available Web server. The Apache Web server is HTTP/1.1 compliant web server implementing the latest protocols, including HTTP/1.1 (RFC2616). In addition, it is highly configurable and extensible with third-party modules can be customized by writing modules using the Apache module API and provides full source code an unrestrictive license. In the Linux VPN implementation, Apache will listen only on the internal LAN interface in order to provide system administrators convenient access to common system features through the Webmin CGI scripts. This implementation requires the mod_perl package, the mod_ssl package, and OpenSSL.

Apache is the most widely used HTTP-server in the world today. It surpasses all free and commercial competitors on the market, and provides a myriad of features; more than the nearest competitor could give you on a UNIX variant. It is also the most used web server for a Linux system. A web server like Apache, in its simplest function, is software that displays and serves HTML pages hosted on a server to a client browser that understands the HTML code. Mixed with third party modules and programs, it can become powerful software, which will provide strong and useful services to a client browser (Sigle, 2001).

Apache Modules

To include Perl programming language support in your Apache web server,

change into the mod_perl source directory, /usr/src/mod_perl-1.25/, and use the following commands to add the mod_perl source to the Apache source:

```
[root@turtledove mod_perl-1.25/]# perl Makefile.PL \
                EVERYTHING=1 \
                APACHE_SRC=../apache_1.3.12/src \
                DO_HTTPD=1
[root@turtledove mod_perl-1.25/]# make
[root@turtledove mod_perl-1.25/]# make install
```

The above commands use perl to configure the make file for the mod_perl package to include all options, “EVERYTHING”, and apply the source to the Apache source directory listed. The “make” command compiles the mod_perl package, and the “make install” command installs the mod_perl module.

The mod_SSL package is an Apache web server module that provides strong cryptography for web sessions via the Secure Socket Layer protocol library provided by OpenSSL. In order to allow Apache to configure and respond to secure requests, the mod_ssl package must be installed. To install the mod_ssl package, change into the /usr/src/mod_ ssl-2.8.4-1.3.20 directory and use the following command to make the Apache source aware of the mod_ssl package:

```
[root@turtledove mod_ ssl-2.8.4-1.3.20/]# ./configure \
                --with-apache=/usr/src/apache_1.3.20 \
                --with-crt=/etc/ssl/keys/servercrt.crt \
                --with-key=/etc/ssl/keys/serverrsa.key
```

This command adds the mod_ssl source to the Apache source tree so it will be compiled in with the Apacher server. The “--with-apache” option specifies the location of the Apache source directory, the “--with-crt” option specifies the location of your existing signed certificate (public key) for SSL encryption, and the “--with-key” option specifies the location of your existing private key for SSL encryption.

Once mod_ssl source has been added to the Apache source and Apache compiled, SSL configuration entries must be added to the Apache configuration file. The mod_ssl package provides its own custom commands that can be defined both globally for all sites provided by the server and for individual virtual hosts. The global configuration for SSL will begin with a check to see if the mod_ssl module is available to Apache: "<IfModule mod_ssl.c>." If the mod_ssl module is available, the configuration directives are loaded by Apache at startup. Figure 12-1 provides a sample of one possible global configuration for Apache.

Initially, the SSLEngine option is turned off in the global context. The SSLEngine option will be turned on only for specific virtual hosts on specific IP addresses and port numbers. This directive toggles the usage of the SSL/TLS Protocol Engine. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts.

When Apache starts up it has to read the various certificate and private key files of the SSL-enabled virtual servers. Since the private key files are usually encrypted for security reasons, mod_ssl needs to query the administrator for a Pass Phrase in order to decrypt those files. Here the SSLPassPhraseDialog is set to builtin. This is the default where an interactive terminal dialog occurs at startup requiring the system administrator to manually enter the pass phrase for each encrypted private key file. Mod_ssl will attempt to reuse any previously entered passwords before prompting for a new one. If one of the previous passwords works, then the prompt is not needed.

The SSLSessionCache option configures the storage facility, or cache, for SSL sessions that can speed up parallel request processing. Since modern clients can make up

```
##########################################################
##  SSL Global Context Configuration
##
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts
##       (unless overridden by virtual hosts)
##
<IfModule mod_ssl.c>
##  SSL Support
##  When we also provide SSL we have to listen to the
##  standard HTTPS port - 443
##
Listen 192.168.1.32:443
SSLEngine off
SSLPassPhraseDialog  builtin
SSLSessionCache  dbm:/var/log/ssl/ssl_scache
SSLSessionCacheTimeout  300
SSLMutex  file:logs/ssl_mutex
SSLRandomSeed startup file:/dev/urandom 512
SSLRandomSeed connect builtin
SSLCipherSuite
ALL:!ADH:!NULL:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
SSLLog      /var/log/ssl/ssl_engine_log
SSLLogLevel info
CustomLog /var/log/ssl/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
SSLProtocol all
</IfModule>
#
#        END OF SSL GLOBAL CONTEXT CONFIGURATION
##########################################################
```

Figure 12-1. Apache web server SSL global context configuration

to four parallel requests for data, the mod_ssl module can respond to those requests through different pre-forked server processes to avoid unneccessary session handshakes. In this implementation, the mod_ssl module makes use of a DBM type database hashfile on the local system to synchronize the server processes.

To ensure that unused or abandoned SSL sessions are closed and disabled, the SSLSessionCacheTimeout directive will turn off the SSL Protocol Engine after a

predetermined number of seconds. In addition, the SSLSessionCacheTimeout will provide session cache cleaning of stale information.

The SSLMutex option configures the SSL engine's semaphore that is used for mutual exclusion of operations that have to be done in a synchronized manner between the pre-forked Apache server processes. Here the SSLMutex option is configured to use the lock file in the Apache logs directory.

The SSLRandomSeed option configures one or more sources for seeding the Pseudo-Random Number Generator (PRNG) in OpenSSL. This directive can be used to configure Apache's initial startup and when any connection attempt is made.

> "The following source variants are available: builtin This is the always available builtin seeding source. It's usage consumes minimum CPU cycles under runtime and hence can be always used without drawbacks. The source used for seeding the PRNG contains the current time, the current process id and (when applicable) a randomly choosen 1KB extract of the inter-process scoreboard structure of Apache. The drawback is that this is not really a strong source and at startup time (where the scoreboard is still not available) this source just produces a few bytes of entropy. So you should always, at least for the startup, use an additional seeding source.
>
> file:/path/to/source
>
> This variant uses an external file /path/to/source as the source for seeding the PRNG. When bytes is specified, only the first bytes number of bytes of the file form the entropy (and bytes is given to /path/to/source as the first argument). When bytes is not specified the whole file forms the entropy (and 0 is given to /path/to/source as the first argument). Use this especially at startup time, for instance with an available /dev/random and/or /dev/urandom devices (which usually exist on modern Unix derivates like FreeBSD and Linux).
>
> But be careful: Usually /dev/random provides only as much entropy data as it actually has, i.e. when you request 512 bytes of entropy, but the device currently has only 100 bytes available two things can happen: On some platforms you receive only the 100 bytes while on other platforms the read blocks until enough bytes are available (which can take a long time). Here using an existing /dev/urandom is better, because it never blocks and actually gives the amount of requested data. The drawback is

> just that the quality of the received data may not be the best." (Engelschall & Ralf, 2001)

The complex SSLCipherSuite directive uses a colon-separated string of OpenSSL cipher specifications to configure the cipher suites clients can negotiate in the SSL handshake phase. The SSL cipher specification should contain at least the following four attribute algorithms: Key Exchange, Authentication, Cipher/Encryption, MAC Digest.

The default cipher-specification string is "ALL:!ADH:RC4+RSA:+HIGH: +MEDIUM:+LOW:+SSLv2:+EXP." This string translates to the following: Use all ciphers (ALL) but remove any ciphers that do not authenticate (!ADH). Next, in order, use ciphers employing RC4 encoding and RSA key exchange (RC4+RSA), use ciphers employing Triple-DES encoding (+HIGH), use ciphers employing 128 bit encryption (+MEDIUM), use low strength ciphers employing single DES, but not export ciphers (+LOW), use SSL protocol version 2 ciphers (+SSLv2), and, lastly, use export only ciphers (+EXP).

This implementation uses the default cipher-specification string with one addition: do not use ciphers employing no encryption (!NULL). For secure communications, all handshaking should be done with encryption of information passed between the client and server.

The SSLLog directive sets the path and name for the dedicated SSL protocol engine logfile. Either an absolute path or relative path can be used. Generally, error type messages are sent to the general Apache error log file as specfied in the Apache configuration file, so information logging is more than adequate for this implementation. Therefore, the SSLLogLevel directive is set to info to log major processing steps. The log files should be stored in a subdirectory of the /var directory. The /var directory is used

for dynamic data and is segregated from other directories on a separate partition, so create a subdirectory /var/logs/ssl/ to store the logs and set the access permissions so only root can write. By doing this, the logs should be safeguarded so they cannot be used for symlink attacks on a real server.

In this implementation of the Linux VPN server, only the system administrator needs access to the web services provided on the LAN side. Therefore, it is a good idea to log all the requests made to the server through the mod_ssl CustomLog directive. The CustomLog directive will create a log of all SSL requests made to the server and should be checked regularly for suspicious activity. In addition, it can be a useful tool to see what ciphers and protocols are being used during the SSL handshake.

Finally, the SSLProtocol directive can be used to control the SSL protocols mod_ssl uses to allow client connectivity. Clients can only connect with one of the protocols specified in the SSLProtocol directive.

In this implementation, the server will accept all available SSL protocols supported by the mod_ssl module: SSLv2, SSLv3, and TLSv1.

To add additional client authentication features for the Apache secure services, the mod_auth_external module will be used to authenticate the client after the SSL handshake. The mod_auth_external module allows Apache to call an external password authentication program; this way, Apache need not have access to the password or shadow password files. The mod_auth_external module will be configured to pass the user's login and password to the external authentication program via a pipe. The external authentication program will access the shadow password file to authenticate the client. If successful, the external authentication program will return SUCCESS to Apache and

Apache will service the request from the client. For security reasons, do not allow the external authentication file to directly access the shadow password file. Rather, allow the pluggable authentication modules to do the work instead.

Install the mod_auth_external module

Copy the mod_auth_external source file from the unzipped directory to the Apache extra modules directory:

```
cp mod_auth_external-2.1.9/mod_auth_external.c \
        apache_1.3.20/src/modules/extra
```

Next, make modifications to the pwauth source code before the install. Edit the config.h file to enable PAM support as shown in Figure 12-2.

```
/* #define SHDOW_NONE          /**/
/* #define SHADOW_BSD          /**/
#define SHADOW_SUN             /**/
/* #define SHADOW_JFH          /**/
/* #define SHADOW_MDW          /**/
/* #define SHADOW_AIX          /**/
#define PAM                    /**/
/* #define PAM_SOLARIS_26      /**/
```

Figure 12-2. Modify pwauth source code

Edit the Makefile to include the libraries required for PAM support:

```
CC=gcc
LOCALFLAGS= -g
# For PAM on Redhat Linux
LIB=-lpam –ldl
```

Finally, issue the “make” command in the source directory to build the executable binary. Once the executable is built, move it to the /usr/local/libexec/pwauth/ directory. Make sure the permissions are set so only root can access the program. Once this is complete, the interface to PAM must be written. Create the file /etc/pam.d/pwauth and add the lines listed in Figure 12-3.

```
#%PAM-1.0
auth        required    /lib/security/pam_pwdb.so shadow nullok
auth        required    /lib/security/pam_nologin.so
account     required    /lib/security/pam_pwdb.so
```

Figure 12-3. Configuration of /etc/pam.d/pwauth

The pwauth program is an external suid-root program that is run by mod_auth_external to do authentications through PAM. To configure Apache to use pwauth for secure authentications, add the following lines to the http.conf file

```
AddExternalAuth pwauth /full/path/to/pwauth
SetExternalAuthMethod pwauth pipe
```

Place the secure authentication directives above the SSL Global Context Configuration directives. Now Apache can provide secure access to web pages and provide secure authentication of clients attempting to view the secure pages.

Compiling and Installing Apache

To install the Apache web server onto the Linux server, change into the source directory, /usr/src/apache_<version>, and use the following commands:

```
[root@turtledove apache_1.3.20/]# configure
[root@turtledove apache_1.3.20/]# make
[root@turtledove apache_1.3.20/]# make install
[root@turtledove apache_1.3.20/]# rm -f /usr/sbin/apachectl
[root@turtledove apache_1.3.20/]# rm -f /usr/man/man8/apachectl.8
[root@turtledove apache_1.3.20/]# rm -rf /home/httpd/icons/
[root@turtledove apache_1.3.20/]# rm -rf /home/httpd/htdocs/
```

The “configure” command will configure the Makefile for the Linux server install. The “make” command compiles all files in the source tree into executable binaries, and the “make install” command installs the binaries and any supporting files into the appropriate locations. The “rm -f” commands remove extraneous files from the install directories.

Define a Secure Virtual Host

The next step in providing secure web services is to create a virtual host that will respond to requests through the SSL protocol. Configuring a virtual host for Apache is relatively straightforward, so the focus of this discussion will be on the implementation of the SSL services provided through the virtual host. The global SSL directives for all hosts offered by the Apache server are defined as above before defining a specific virtual host. Traditionally, a secure host will listen for secure communication requests only on port 443. Most web browsers will default to making requests on port 443 when a URL is addressed using the secure http protocol, https. In any case, secure services can be offered on any port number, and in order to contact the host, the port number, preceeded by a colon, would need to be appended to the end of a URL. For example, https://www.destinationearth.net:12431, would be a valid means to request secure services on port 12341. In the Apache virtual host directive the site is not addressed using a URL, rather it is addressed by the IP address and port number to listen for requests: <VirtualHost 192.168.6.1:443>. The format and directives for a secure virtual host in this implementation are listed in Figure 12-4.

The most importation directives for SSL are the SSLEngine, SSLCertificateFile, and SSLCertificateKeyFile. As described previously, the SSLEngine directive turns on the SSL protocol for only the virtual host containing the directive. The SSLCertificateFile directive points Apache to the certificate file, saved in the /etc/ssl/keys directory, to be used for secure handshaking. The SSLCertificateKeyFile directive points Apache to the private key file, also contained in the /etc/ssl/keys directory. The private key file should not be readable or writeable by anyone other than root. The permission

should have been set correctly after creating the files, but a quick look at the file permissions is a good idea to ensure the security of the private key.

```
<VirtualHost 192.168.6.1:443>
        ServerName secure.destinationearth.net
        ServerAlias destinationearth.net www.destinationearth.net
        # primary content
        DocumentRoot        /home/httpsd/configuration/html
        ScriptAlias /cgi-bin /home/httpsd/configuration/cgi-bin
        # alias for Apache icons
        Alias /icons/ /var/www/icons/
        # SSL Directives
        SSLEngine on
        SSLCertificateFile /etc/ssl/keys/servercrt.crt
        SSLCertificateKeyFile /etc/ssl/keys/serverrsa.key
        # End SSL Directive
        SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
</VirtualHost>
```

Figure 12-4. Apache Secure Virtual Host Configuration

The Apache web server should not run as super-user, and indeed in most recent versions it cannot be run that way. Earlier the httpsd user and group were created for the Apache server. The Apache directives, User and Group, are thus set to “httpsd.”

The SSLRequireSSL directive denies access to SSL-enabled virtual hosts or directories unless HTTP over SSL (HTTPS) is enabled for the current connection. When this directive is present all requests are denied that are not using SSL.

The SSLRequire directive specifies criteria that have to be fulfilled in order to allow access to a virtual host or directory. Here, the server will only accept requests using a minimum cipher key size of 128 bits or greater. Requests must be made from a browser supporting 128-bit encryption, otherwise access to the server will be denied.

In addition, many configuration directives exist for the configuring the Apache server, however, these directives fall well beyond the scope of this document. For further

information regarding the configuration of the Apache server, see the "Apache Overview HOWTO" by Daniel Lopez Ridruejo available from http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO.html and the Apache server reference manual available from http://httpd.apache.org/docs/.

Starting Apache

Create the script file /etc/rc.d/init.d/httpd to start and stop the Apache web server (Bowen, Coar, Grip-Jansson, Marlowe, & Chinnappan, 2000). To create the httpd script file issue the command: touch /etc/rc.d/init.d/httpd. Next, open the file and add the listing from Figure 12-5.

To make the script executable and ensure only the root user can run it, change the permissions on the script file:

```
[root@turtledove /]# chmod 700  /etc/rc.d/init.d/httpd
```

Finally, ensure that Apache will start at boot time by creating the symbolic links in each necessary run level:

```
[root@turtledove /]# chkconfig --add httpd
```

Note: the -startssl option will start Apache in SSL mode. Do not remove this option, otherwise, secure communications will not be possible.

Securing Apache

To add additional security to the Apache server, change the file permissions as listed below:

```
[root@turtledove /]# chmod 511 /usr/sbin/httpd
[root@turtledove /]# chmod 750 /etc/httpd/conf/
[root@turtledove /]# chmod 750 /var/log/httpd/
```

```
#!/bin/sh
    # Startup script for the Apache Web Server
    # chkconfig: 345 85 15
    # description: Apache is a World Wide Web server.  It is used to serve \
    #          HTML files and CGI.
    # processname: httpd
    # pidfile: /var/run/httpd.pid
    # config: /etc/httpd/conf/httpd.conf
    # Source function library.
    . /etc/rc.d/init.d/functions
    # See how we were called.
    case "$1" in
    start)
    echo -n "Starting httpd: "
    daemon httpd -startssl
    echo
    touch /var/lock/subsys/httpd
    ;;
    stop)
    echo -n "Shutting down http: "
    killproc httpd
    echo
    rm -f /var/lock/subsys/httpd
    rm -f /var/run/httpd.pid
    ;;
    status)
    status httpd
    ;;
    restart)
    $0 stop
    $0 start
    ;;
    reload)
    echo -n "Reloading httpd: "
    killproc httpd -HUP
    echo
    ;;
    *)
    echo "Usage: $0 {start|stop|restart|reload|status}"
    exit 1
    esac
    exit 0
```

Figure 12-5. Configuration of /etc/rc.d/init.d/httpd

This sets the binary program readable by only root, and executable by everyone. Further, these commands lock down the /etc/httpd/conf/ and /var/log/httpd from others outside the root user and the httpsd user and group.

To test the Apache install, point a web browser to the following address: https://my-web-server/private/. Where <my-web-server> is the fully qualifed host name of the server, and </private/> is the protected directory.

CHAPTER 13
WEBMIN

The Webmin package will be used for remote configuration of the Linux VPN server. It provides a web-based interface with modules that allow interaction with system files and settings. This service will be provided exclusively on the internal network, and only the IP address of the system administator's workstation will be permitted access. In addition, SSL services will be used to require password authentication before using the service. Webmin can use SSL to secure connections between a SSL-enabled web browser and the Webmin server. Before installing the Webmin package, install the

Table 13-1. Webmin Configuration

Prompt	Response
Config file directory	/etc/webmin
Log file directory	/var/webmin
Full path to perl	/usr/bin/perl
Operating System	4 (Redhat Linux)
Version	9 (Redhat Linux 6.2)
Webserver port	8443
Login name	admin
Login password	<password>
Webserver hostname	turtledove.destinationearth.net
Use SSL	y
Start Webmin at boot time	n

Net::SSLeay perl module downloaded earlier and extracted to /usr/src/Net_SSLeay.pm-1.05/:

```
[root@turtledove /]# cd /usr/src/Net_SSLeay.pm-1.05
[root@turtledove Net_SSLeay.pm-1.05]# perl Makefile.PL
[root@turtledove Net_SSLeay.pm-1.05]# make install
```

NOTE: since the OpenSSL package was intalled from an RPM, it may be necessary to use the following command to find the OpenSSL library properly:

[root@turtledove Net_SSLeay.pm-1.05]# perl Makefile.PL /usr

To test the Net_SSLeay.pm install, use the command “perl -e 'use Net::SSLeay” to see if any error messages are output (Sampo, 2001). If no errors are output, then the SSL support that Webmin needs is properly installed.

```
<VirtualHost 192.168.6.1:443>

        SetEnv WEBMIN_CONFIG /etc/webmin
        SetEnv WEBMIN_VAR /var/webmin
        SetEnv MINISERV_C0NFIG /etc/webmin/miniserv.conf

        ServerName secure.destinationearth.net
        ServerAlias destinationearth.net www.destinationearth.net

         # primary content

        DocumentRoot        /home/httpsd/configuration/html/webmin-0.88
        ScriptAlias /cgi-bin /home/httpsd/configuration/html/cgi-bin

         # alias for Apache icons

        Alias /icons/ /var/www/icons/

        # SSL Directives
        SSLEngine on
        SSLCertificateFile /etc/ssl/keys/servercrt.crt
        SSLCertificateKeyFile /etc/ssl/keys/serverrsa.key
        # End SSL Directives

        SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0

</VirtualHost>
```

Figure 13-1. Apache Webmin virtual host

Installation

To install the Webmin package extracted to /usr/src/ webmin-0.88, use the following set of commands:

[root@turtledove /]# cp –R /usr/src/webmin-0.88 /home/httpsd/ configuration/html

```
[root@turtledove /]# cd /home/httpsd/configuration/html/webmin-0.88
[root@turtledove webmin-0.88]# ./setup.sh
```

As the “setup.sh” shell script runs, it will prompt for several items. The prompts and the corresponding responses are listed in Table 13-1.

The setup script will give the URL and port number to use to access Webmin. Enter the URL, substituting the IP address for the hostname, into a web browser and login at the login prompt with the username and password specified at install time. Once logged in, the browser will load the main Webmin page, on which is an icon for each module installed.

To allow Apache to server the Webmin pages, add the virtual host configuration listed in Figure 13.1 to the httpd.conf file.

Next, add the directory configuration listed in Figure 13.2 to the httpd.conf file.

```
<Directory /home/httpsd/configuration/html/webmin-0.88>
   SSLRequireSSL
   SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
   DirectoryIndex index.cgi
   Options Includes, ExecCGI
   AllowOverride None
   order deny,allow
   deny from all
   allow from all
   AuthName "Webmin-0.88 administration"
   AuthType Basic
   AuthUserFile /etc/webmin/miniserv.users
   AuthExternal pwauth
   require valid-user
</Directory>
```

Figure 13-2. Apache Webmin web directory

Configuration

To allow the Webmin programs to function correctly, make all the Webmin programs owned by root and setuid with the commands:

```
chown -R root:root /usr/local/webmin-0.88
chmod -R 6755 /usr/local/webmin-0.88
```

Finally, add the -U flag to the perl line in all the Webmin scripts. This can be easily done with the following command run from the webmin-0.88 directory:

```
find . -name "*.cgi" -o -name "*.pl" | perl perlpath.pl "/usr/bin/perl -U" –
```

Restart Apache and login to Webmin at the virtual host configured for Apache: 192.168.6.1:443. Note that the Webmin Users and Webmin Configuration modules will not work, as they configure miniserv.pl and cannot deal with Apache.

CHAPTER 14
FREES/WAN IPSEC

FreeS/WAN IPSEC Configuration

The FreeS/WAN IPSEC package is the heart of the Linux VPN server. This package will negotiate and establish a secure virtual private network between Linux servers across the unsafe Internet. The FreeS/WAN package makes use of the character device random to generate encryption schemes. Typically, the number of random 32 bit words used to generate a random number is set low for speed's sake. Instead, edit the /usr/src/linux/drivers/char/random.c file and change the randomness pool size to 512 32 bit words rather than 128: #define POOLWORDS 512

Save the changes to the file, and the increase in randomness pool size will be compiled into the new kernel after adding support for the IPSEC protocol.

Next, change into the /usr/src/freeswan-1.91 directory and issue the following command:

```
[root@turtledove freeswan-1.91]# make menugo
```

This command invokes the kernel configuration utility menuconfig and adds support for the IPSEC protocol to the kernel. While configuring the kernel, make certain the following options are configured correctly (Note: the correct configuration options are bold and underlined):

```
Loadable module support
Enable loadable module support (CONFIG_MODULES) [Y/n/?]
IP: tunneling (CONFIG_NET_IPIP) [N/y/?]
```

A common tool for attackers is a "rootkit", a set of tools used once they have become root. A rootkit is a collection of tools (programs) that an attacker can use to mask intrusion and obtain administrator-level access to a computer or network. If an intruder can obtain user-level access via password cracking or through exploiting a known vulnerability, then she can install a rootkit. The goal of the rootkit is to collect userids and passwords to other machines on the network. With these passwords, the attacker hopes to gain root or privileged access and to introduce assorted additional compromises so that they "own" the system despite most recovery efforts. A rootkit may consist of utilities that also: monitor traffic and keystrokes; create a "backdoor" into the system for the hacker's use; alter log files; attack other machines on the network; and alter existing system tools to circumvent detection. With modules disabled, an attacker cannot install a rootkit module or other bogus module. The only way to achieve the same effects is to install a new kernel and force a reboot which is considerably more likely to be noticed (Retallack, 2001).

Save the configuration even if no changes are made to ensure that the FreeS/WAN changes are actually seen by the system. Once the configuration is saved, the ipsec utilites are compiled. Next, compile the new kernel. The ipsec utilities are installed automatically, but the new kernel image must be installed. See the section on compiling the kernel for a refresher on how to compile and install a new kernel.

The “make menugo” also automatically builds an RSA authentication key pair, a public key and the matching private key, in the /etc/ipsec.secret file. The public key does not need to be kept secure. In fact, once the public key is created, a copy is provided to any client wanting to communicate with the Linux VPN server. On the other hand, the

private key must be kept secure on the Linux server; no one else should have access to the server's private key. For FreeS/WAN, both the public and private keys are stored in the /etc/ipsec.secrets file. This file must be kept secure at all times since it holds the server's private key. The public keys for systems communicating with the VPN server are placed in the /etc/ipsec.conf file. Security for the /etc/ipsec.conf file is less vital since the public key is made publicly available, and is worthless without the private key. This does not mean that the file's permissions should be lax; it also contains information about the VPN connections and so should be secured from non-root users.

Setting up RSA Authentication Keys

The RSA authentication key pair generated by the "make menugo" is not sufficient for a production system and is intended only for testing purposes. To generate a new RSA key pair use the following command:

```
[root@turtledove /]# ipsec rsasigkey --verbose 2048 > /tmp/#rsakey#
```

This will generate a 2048-bit signature key using the /dev/random character device and put it in the file /tmp/#rsakey#. The file contents can be inserted as is into an entry in the ipsec.secrets file and the public key can then be extracted and placed into the ipsec.conf file. The /tmp/#rsakey# will contain both the public and private keys.

Create a new /etc/ipsec.secrets file and copy the entire contents of the #rsakey# file into the new file. Wrap the entire contents of the file with the wrapper shown below:

```
: RSA {
  <contents of the #rsakey# file>
   }
```

It is important to note the ":" must be unindented and all other lines, including the "}", must be indented. The spaces are needed to separate tokens so ":RSA" would not be a

valid entry for the ipsec.secrets file. See Figure 14-1 for a sample /etc/ipsec.secrets file

(Note: the public key is highlighted in bold typeface):

```
# This file holds shared secrets or RSA private keys for inter-Pluto
# authentication.  See ipsec_pluto(8) manpage, and HTML documentation.

# Shared secret (an arbitrary character string, which should be both long
# and hard to guess, enclosed in quotes) for a pair of negotiating hosts.
# Must be same on both; generate on one and copy to the other.
#10.1.1.1 10.1.1.10 #"jxR5kS55l2ulmn11WU3uU21U1j3ku
WS3ljkS4uWkk3nRV2uWljjUmkSuSj1WVkW#Uu1mWmk31l"

# RSA private key for this host, authenticating it to any other host
# which knows the public part.  Put ONLY the "pubkey" part into connection
# descriptions on the other host(s); it need not be kept secret.
: RSA {
        # 2048 bits, Wed Jun 28 12:59:57 2000
        # for signatures only, UNSAFE FOR ENCRYPTION
        #pubkey=0x0103af2bff44f443007bc6d3d56fd1f8037414f4040c1a
70979d27f78bf886418a3587044110adae378e75c927e90897d77285ae2ae0bb
5ce3ea3ff383dae36d87aa07c9461e7e19e78c2d8ce40d55e433b42b8c80560c2
dae8c5730399093e22ec5bd389447ef0fad9697f60957c17917341ad970ecbe9a
273d421e9b645525278b99fdd185161146cc8390a5d7a65216a334e2e0dec886
082e39356b45bedef03f0a586691529a035b4cb5838c29a9251ff1506f9dba149
9c22b6db4b157e450fb91a464378c0581f27dc7ee86db2429991d0df65abeab6
510d4789b7f0ff5196d3c15cb1fb90a4f81b3bccfde93f4c9d9ec289a3a7099ad9
906422af98b2a87f5
        #IN KEY 0x4200 4 1 AQOvK/9E9EMAe8bT1W/R+AN0FPQEDBpwl5
0n94v4hkGKNYcEQRCtrjeOdckn6QiX13KFrirgu1zj6j/zg9rjbYeqB8lGHn4Z5
4wtjOQNVeQztCuMgFYMLa6MVzA5kJPiLsW9OJRH7w+tlpf2CVfBeRc0Gt
lw7L6aJz1CHptkVSUni5n90YUWEUbMg5Cl16ZSFqM04uDeyIYILjk1a0W+
3vA/ClhmkVKaA1tMtYOMKaklH/FQb526FJnCK220sVfkUPuRpGQ3jAWB8
n3H7obbJCmZHQ32Wr6rZRDUeJt/D/UZbTwVyx+5Ck+Bs7zP3pP0ydnsKJo6
cJmtmQZCKvmLKof1
        # (0x4200 = auth-only host-level, 4 = IPSec, 1 = RSA)
        Modulus: 0xaf2bff44f443007bc6d3d56fd1f8037414f4040c1a70979d27f
78bf886418a3587044110adae378e75c927e90897d77285ae2ae0bb5ce3ea3ff383
dae36d87aa07c9461e7e19e78c2d8ce40d55e433b42b8c80560c2dae8c573039909
3e22ec5bd389447ef0fad9697f60957c17917341ad970ecbe9a273d421e9b645525
278b99fdd185161146cc8390a5d7a65216a334e2e0dec886082e39356b45bedef0
3f0a586691529a035b4cb5838c29a9251ff1506f9dba1499c22b6db4b157e450fb9
1a464378c0581f27dc7ee86db2429991d0df65abeab6510d4789b7f0ff5196d3c15c
b1fb90a4f81b3bccfde93f4c9d9ec289a3a7099ad9906422af98b2a87f5
        PublicExponent: 0x03
```

Figure 14-1. Configuration of /etc/ipsec.secrets

```
# everything after this point is secret
        PrivateExponent: 0x74c7ff834d820052848d38f536a557a2b8a2ad5d66f5b
a68c54fb2a5aed65c23af582b6073c97a5ef930c54605ba8fa1ae741c95d23ded46d5
4d0291ecf3afc6afdb8414541145081e5ded5e3942cd22c7b300395d73c9b2e4cad10
b0d417483d37b0d854a0a73b9baa4063a80fb64cd673ba09dd466c4d38169bced8e1
8c5069f076f8832227f752b6b3aa36a6cdefbc6620b8017c917cf92d09da5241b2340
e74f22fd116c38e693b629abb75a4f0f71846c069e35abff1df17ca4e25c16d411f08b
2860b8beea30093e0a7ac8e3d123d17f11d2d595a8577278128a46012ff7b6ed884f8
fec4897b2c9937acc5adbc0e68fed2aa26c3b4d7834375eac18b3
        Prime1: 0xf29a11a5479676a1b909fc7e29c452bc5c952caae8cbab02c0e620
3a14a717633b569860c57bcf2f5d04924a1a76da79984fc656e502df337595f385a43
bd9629bc346efdc22d53659b0cf9642c1e567c13622aa04164cb72213c26330bec99
75da8804b8d42c344ec8d47186515f54207e4c83366c855c2d9daacbb485b9e57
        Prime2: 0xb8d89893834710fb0965d1645cea756d2ebaa2f3bc0cb9741c165
e93f40f23fa740b19b4f2fbded2121fb2c07c2ad40f2eba073ce7c1389008ed8634601e
555adaf84c8b1ecd8e5cdc0941e0e034ee77a61d9515fe54b7dd2f3925000fcd4e0dead
ae687b46fa9cfe3a35a68f6ac4c4e42e9fa2349ddb57d38332feb34ccc493
        Exponent1: 0xa1bc0bc3850ef9c12606a8541bd837283db8c871f087c7572b4
4157c0dc4ba42278f104083a7df74e8adb6dc11a491a6658a8439ee01ea224e63f7ae6
d7d3b9712822f4a92c1e37991208a642c81439a80cec1c6ad643324c162819775d486
64e91b00325e2c822df308da104363f8d6afeddaccef30392c913c73278592698f
        Exponent2: 0x7b3b10625784b5fcb0ee8b983df1a39e1f27174d28087ba2bd6
43f0d4d5f6d51a2b211234ca7e9e1616a772afd71e2b4c9d15a289a80d06005f3aecd9
5698e3c91faddb2148909933d5b814095789efa6ebe6363fee3253e1f7b6e000a88deb
3f1e7445a784a7135426ce6f0a472dd8981f1516cdbe923a8d0221ff223332db7
        Coefficient: 0x80079ef72e2bd39d36fb2ff6acb881758ea7a8082638077f24a
36911f2d38a5bc7d51e06ecfcdc511ed8544a935e3a6a6f00723e4b107749cad66085f
bd06833d27aba0f7b636e6ede3ef9fe79c28e5eccf39000245cc81a1f32ee9cf2d5c5bf
af0d3cf331d5ea223859d122ffb26ea7c071eb0d78d924ab0ad186545dfac3e3
        }
# do not change the indenting of that "}"
```

Configuration of /etc/ipsec.secrets continued

Even though the new keys are stronger than the orginal keys, the FreeS/WAN generated RSA keys are suitable only for authentication, and must not be used for encryption. Encryption for the VPN tunnel will be provided by using the block cipher—a symmetric cipher that operates on fixed-size blocks of plaintext, giving a block of ciphertext for each—Triple DES. Triple DES (3DES) uses three DES—the Data Encryption Standard which is a block cipher with 64-bit blocks and a 56-bit key

encryptions on a single data block—passes with at least two different keys, to get higher security than is available from a single DES pass (FreeS/WAN documentation).

Exchanging authentication keys

The public key used to negotiate IPSEC connection is the line contained in the output of the rsasigkey command starting with "#pubkey=0x". Public keys remain public, and the FreeS/WAN configuration system is designed to work even if an enemy knows all the public keys used. However, for the integrity of the system to remain sound, remember to always authenticate the public key. If an attacker can forge a public key and the server is tricked into trusting the forged key, then the attacker can gain access to the previously secure communications. For example, consider the following example from the FreeS/WAN documentation.

> "[A husband] wants to communicate with his mistress, keeping messages secret from his wife:
>
> · If the wife obtains the mistress' public key, that is not a problem. As long as she does not get the private key, she can neither read things sent to the mistress nor authenticate herself as the mistress.
>
> · If the mistress has any sense, she protects her private key carefully. So long as she does that, and the husband encrypts his messages correctly, there should be no (cryptographic!) problem.
>
> · However, imagine that the wife is somewhat devious. She generates a public/private key pair and sends the husband that public key, forging the message to look as if it came from the mistress. Of course this fails if the husband has enough sense to check the key's validity before using it.
>
> · However, if the husband blindly accepts that key without verification, it is extremely unlikely that he will be pleased with the results.
>
> · If he accepts that key, the wife can read every message he sends to it.

- She can also pose as the mistress and send him whatever messages she likes. “(FreeS/WAN documentation, n.d.).

Authentication of a public key’s origin is an essential step in providing a secure VPN server. Exchange keys encrypted with a message digest to confirm the integrity of the key and from whom it came.

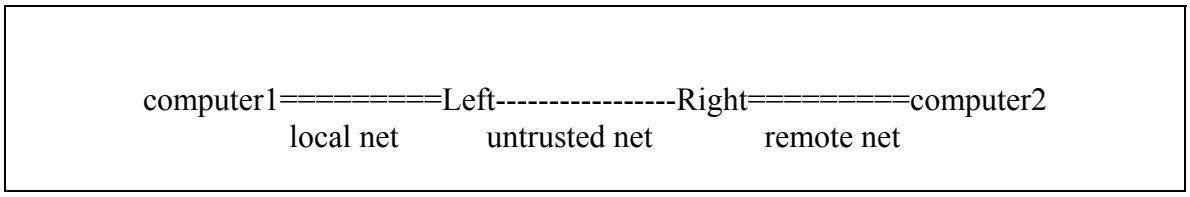

Figure 14.2. Virtual private network connection

IPSEC configuration file

For example purposes, consider the network configuration listed in Figure 14.2. The network consists of two Linux servers, Left and Right, two networks behind the Linux servers, and two computers, computer1 and computer2, on separate networks, separated by an insecure, or untrusted network such as the Internet. The goal is to connect the Left server and the Right server through an IPSEC VPN and forward traffic between the network behind each server. The /etc/ipsec.conf file is used to provide configuration parameters for a connection to the IPSEC server. A sample configuration file is listed in Figure 14-3. (Note: the RSA public keys have been shortened for readablility).

The config setup section describes the machine configuration for the server. The interfaces variable tells the KLIPS—Kernel IP Security, the Linux FreeS/WAN project's changes to the Linux kernel to support the IPSEC protocols—IPSEC code in the Linux kernel which network interface to use. The interface(s) used for communication between VPN servers should be listed in the following format: interfaces="ipsec0 =eth0.” If the

```
########################################################################
# /etc/ipsec.conf - FreeS/WAN IPSEC configuration file
########################################################################

# basic configuration
config setup
# THIS SETTING MUST BE CORRECT or almost nothing will work;
# %defaultroute is okay for most simple cases.
        interfaces="ipsec0=eth0"
        # Use auto= parameters in conn descriptions to control startup actions.
        plutoload=%search
        plutostart=%search
        # Close down old connection when new one using same ID shows up.
uniqueids=yes

# defaults for subsequent connection descriptions
conn %default
# How persistent to be in (re)keying negotiations (0 means very).
        keyingtries=0
        # Use RSA keys for authentication
        authby=rsasig

# connection from home office to remote office in Arrington
conn home-arrington
        # Left security gateway, subnet behind it, next hop toward right.
        leftrsasigkey=0x01034db9045c0fd89ac1e7c2787c79632747...
        left=209.86.84.125
        leftnexthop=209.86.84.124
        leftsubnet=192.168.6.0/24
        leftfirewall=yes

        # Right security gateway, subnet behind it, next hop toward left.
        rightrsasigkey=0x0103af2bff44f443007bc6d3d56fd1f80374...
        right=207.33.129.114
        rightnexthop=207.33.129.113
        rightsubnet=192.168.3.0/24
        rightfirewall=yes

        auto=add
```

Figure 14-3. Configuration of /etc/ipsec.conf

correct interfaces are not specified, then nothing will work with the IPSEC protocol and no VPN connections can be established.

FreeS/WAN's Pluto is an IKE, IPsec Key Exchange, protocol and daemon that is used to automatically build shared security associations between VPN servers. According to the FreeS/WAN documentation, a "Security Association (SA) is an agreement between two network nodes on how to process certain traffic between them. This processing involves encapsulation, authentication, encryption, or compression."

IKE is used to negotiate Security Associations between connecting servers. If both servers can find an agreeable set of characteristics for a Security Association through the Pluto IKE daemon, and both recognize each other's authenticity, they can set up a Security Association. Refer to RFC 2409, http://www.ietf.org/rfc/rfc2409.txt, for further information about the Internet Key Exchange protocol.

The plutoload and plutostart variables will contain a list of connections to be automatically loaded into memory or started when Pluto starts. Rather than specify each individual connection to load or start, the plutoload and plutostart variables can be set to "%search". In this case, the any connection with auto=add in its connection definition is then loaded, and any connection with auto=start is started.

Be aware, that only one of the Linux servers should be configured to automatically start VPN connections when Pluto starts. Having a server attempt to rebuild tunnels to systems that are unavailable will waste considerable resources. In genral, the server in the home office should have auto=add for all connections in its ipsec.conf file, and the remote server should use auto=start to initiate connection negotiations. This will ensure that the home office server will be able to continue providing services to other VPN connections without bogging down trying to contact a down remote server.

The uniqueids variable should be set to “yes” so when a connection is dropped and re-established Pluto will automatically remove the old connection to allow the new connection to function correctly.

The conn default section specifies the default parameters to apply to all connections. The keyingtries variable tells Pluto how persistent to be in (re-) keying negotiations. This variable takes an integer value where “0” instructs the Pluto daemon to be persistent, retry forever, in re-keying the connections. The authby variable is set to authenticate server using PKI negotiation with RSA signatures.

The conn section describes the network and keying parameters for each server in the connection defined. A string describing the connection should follow the conn key word. The string should be just informative enough to describe the type of connection made. For example, “home-arrington” would suggest a connection from the home office to the remote office in Arrington. In the connection description section, it does not matter which server is the left connection and which server is the right connection. For clarity’s sake the server defined in the left connection should be the central server to which all other VPN connections are made. In this case, the left connection would be the home office server.

The left variable defines the IP address of the external network adapter, eth0. The leftnexthop variable points to the default gateway left should send packet addressed to right. Typically, this is the IP address of the router providing access to the Internet. It is important to note that FreeS/WAN’s kernel instructions, KLIPS, bypasses the normal routing machinery, so KLIPS must have the routing information even though the routing services already have the information. The leftsubnet variable lists the IP addresses for

the internal network that left is protecting. The leftfirewall variable must be set to "yes" if masquerading is set up on the server since a firewall of that type will suppress IP forwarding of non-routable IP addresses. The auto variable works as specified in config setup section of this document. Finally, the leftrsasigkey variable contains the entire public key for the left server. For each left connection variable, there is a corresponding right connection variable.

"Note that a connection to a subnet behind left does not include left itself. The tunnel described…protects packets going from one subnet to the other. It does not apply to packets that either begin or end their journey on one of the gateways. If you need to protect those packets, you must build separate tunnel descriptions for them.

It is a common error to attempt testing a subnet-to-subnet connection by pinging from one of the gateways to the far end or vice versa. This does not work, even if the connection is functioning perfectly, because traffic to or from the gateway itself is not sent on that connection" (FreeS/WAN documentation, n.d.).

To enable the machine on the other end of the VPN connection, after installing the FreeS/WAN package and generating the RSA keys, copy the /etc/ipsec.conf to the other server. Once the servers reboot, Pluto should take over to load and start the VPN connection. For additional information about configuring the FreeS/WAN package, refer to the FreeS/WAN documentation available at the FreeS/WAN website.

Once the left and right server pass the authentication stage, the connections are set to be automatically (re-) keyed; the keys will change regularly so an opponent who gets one key cannot gain access to a large amount of data. However, when an opponent does obtain a private key he cannot automatically gain access to any encryption keys or any

data. Be aware that once the authentication mechanism is subverted there is no way to prevent the attacker from getting keys and data, but the attacker still has to work for them. In this case, new public and private keys must be generated and exchanged to prevent further system intrusion.

Securing the IPSEC Configuration and Key Files

Check the modes and permissions of the /etc/ipsec.secrets file to be sure that the super-user "root" owns the file, and its permissions are set to block all access by others bot included in the wheel group:

[root@turtledove /]# chmod 660 /etc/ipsec.secrets

This will add another layer of protection for the private keys stored on the server. In addition, make sure to remove the #rsakey# file created in the /tmp directory.

Next, edit the /etc/services file and add the following (if not there already):

```
isakmp        500/tcp    isakmp
isakmp        500/udp    isakmp
```

Configuring Remote Gateways for IPSEC Communication

The system administrator can use the read and write accounts created on each Linux gateway to remotely enable a similarly configured Linux gateway to provide VPN access to the local network. The SSH and SFTP tools were included in the Linux VPN implementation to facilitate secure configuration of remote systems.

To begin with, gather the network configuration information from the remote location. Be sure to get the external network IP address, the default gateway and subnet mask. In addition, make note of the internal LAN subnet addresses and subnet masks. This information will be used for connecting the local and remote gateway together through the VPN.

Use the following steps on the local machine to remotely configure a new VPN gateway:

1. Generate a set of public and private keys for the remote gateway machine.
2. Create a new ipsec.secrets file for the remote gateway name ipsec.secrets.remotenetwork where “remotenetwork” corresponds to the name of the remote network.
3. Copy the existing ipsec.conf file to a new file named ipsec.conf.remotenetwork.
4. Edit the ipsec.conf.remotenetwork file and remove any connection descriptions related to other established VPNs.
5. Add a new connection description for the local gateway and the remote gateway. Make sure to include the appropriate public keys.
6. Once the files are configured appropriately, contact the system administrator at the remote site for secure shell access and read and write account passwords.
7. Log into the remote machine using SSH and the write user account and password. Initiate a SFTP session with the remote system and copy the IPSEC configuration files to the remote system.
8. Edit the firewall configuration to add the new VPN connection to the allowed services.
9. Once the transfer of files and firewall changes have been made, issue a remote restart command.
10. Once the remote gateway machine restarts, the IPSEC protocols will negotiate a connection.
11. Verify the VPN connection by pinging an address on the remote network’s LAN.

The VPN connection should then be complete.

CHAPTER 15
FINAL CONFIGURATION

The final steps for securing the Linux VPN server involve locking down all network access and then re-opening access to only a few, pre-determined services. A packet filtering firewall will be used to deny access to all other services. In addition, all source code, compilers, and compiler tools will be removed from the system. Combined with a final check to lock down file permissions, the removal of all compilers will deny access to tools an attacker could use to gain further access to the system.

Firewall

A filtering firewall works at the network level. Data is only allowed to leave the system rules specified in a configuration file allow it. As packets arrive at both the internal and external interfaces they are filtered by the type, source address, destination address, and port information contained in each packet. The firewall applies rules to decide whether to forward or block packets crossing between the two interfaces. Since very little data is analyzed and logged, filtering firewalls take less CPU and create less latency in your network. Filtering firewalls are more transparent to the user. The user does not have to setup rules in their applications to use the Internet.

A simple firewall setup is sometimes called a bastion firewall and is often the main line of defense against attack from external sources. The firewall's purpose is to enforce the site's security policies. Security policies reflect access control and authenticated use of private or protected services, programs and files on the network computers.

A strong firewall should consist of rules limiting connections to specific system ports. There are two classes of port numbers. The ranges from 0 through 1023 are reserved ports. These ports are reserved and well known in order to provide particular standardized services. An incoming connection to one of these ports is routed to the appropriate service by inetd, portmap, or some other server such as sshd or httpd listening on that port. The ports are defined in /etc/services and are well established in the network community (Ziegler, 1999).

The range from 1024 through 65535 represents the system's unprivileged ports. When a client program initiates a connection to a server, a port is selected from the unprivileged pool on the client's end. The combination of the client machine's IP address, port number, and transport protocol (TCP or UDP) defines the client's socket. On the server side, the combination of host IP address, the server's well-known service port number, and the transport protocol (TCP or UDP) form the server's socket. This socket pair uniquely defines the connection between client and server (Ziegler, 1999).

The tool `ipchains` is used to establish firewall rules and talk to the kernel to tell it what packets to filter based on those rules. Ipchains uses a chain approach to creating rules. Within ipchains there are three built-in chains that define rules for input to the firewall, output from the firewall, and forwarding traffic between the internal and external interfaces. New rules are appended to the ends of each chain to filter, allow, or forward specific packets. More information about the specific options for the ipchains tool are available from the IPCHAINS-HOWTO document (Russell, 2000).

Installing the VPN firewall

Create an /etc/rc.d/init.d/rc.firewall script and and /etc/firewall.conf file using the configurations listed in Appendix D. Change the permissions on both files so they are owned by the root and readable and writeable by the wheel group. Upon reboot, the firewall will be implemented and the system should be secure to external traffic.

Final Lockdown

Remove all source files and directories from the /usr/src and /var/temp directories. In addition, use the rpm command to remove the C compilers and libraries installed earlier. These procedures will complete the configuration of the LINUX VPN server and will ready the server to for production.

Power down the server and connect it to the external network and the internal LAN. Power on the system and look for any messages relating to firewall configuration and VPN errors. If errors appear, remove the connection to the external network and review previous chapters to resolve the errors.

Conclusion

Once a system administrator takes the time to analyze and develop rules by which to implement a Linux VPN server, her job of detecting changes to the system becomes easier. In protecting any server from attack, whether originating internally or externally, the most important requirement is regular close inspection of the system. This close inspection of the system makes identifying differences in the system from one period to the next a much more maintainable task. Vulnerabilities and exploits are constantly being discovered and, in my opinion, no one can stay abreast of each one. Rather, the familiarity with the system will allow a system administrator to better protect a system by

noticing changes or anomalies in the system quickly is a much more robust means of defending against attacks.

APPENDIX A
KERNEL CONFIGURATION OPTIONS

Code maturity level options
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [**N**/y/?]
Processor type and features
Processor family (386, 486/Cx486, 586/K5/5x86/6x86, Pentium/K6/TSC/CyrixIII, PPro/6x86MX) [PPro/6x86MX]
Intel IA32 CPU microcode support (CONFIG_MICROCODE) [**N**/y/m/?]
Model-specific register support (CONFIG_X86_MSR) [**N**/y/m/?]
CPU information support (CONFIG_X86_CPUID) [**N**/y/m/?]
Maximum Physical Memory (1GB, 2GB) [**1GB**]
Math emulation (CONFIG_MATH_EMULATION) [**N**/y/?]
MTRR (Memory Type Range Register) support (CONFIG_MTRR) [**N**/y/?]
Symmetric multi-processing support (CONFIG_SMP) [Y/**n**/?]
Loadable module support
Enable loadable module support (CONFIG_MODULES) [Y/**n**/?]
General setup
Networking support (CONFIG_NET) [**Y**/n/?]
PCI support (CONFIG_PCI) [**Y**/n/?]
PCI access mode (BIOS, Direct, Any) [**Any**]
PCI quirks (CONFIG_PCI_QUIRKS) [Y/**n**/?]
PCI bridge optimization (experimental) [Y/**n**/?]
Backward-compatible /proc/pci (CONFIG_PCI_OLD_PROC) [Y/**n**/?]
MCA support (CONFIG_MCA) [**N**/y/?]
SGI Visual Workstation support (CONFIG_VISWS) [**N**/y/?]
System V IPC (CONFIG_SYSVIPC) [**Y**/n/?]
BSD Process Accounting (CONFIG_BSD_PROCESS_ACCT) [**N**/y/?]
Sysctl support (CONFIG_SYSCTL) [**Y**/n/?]
Kernel support for a.out binaries (CONFIG_BINFMT_AOUT) [**Y**/n/?]
Kernel support for ELF binaries (CONFIG_BINFMT_ELF) [**Y**/n/?]
Kernel support for MISC binaries (CONFIG_BINFMT_MISC) [**Y**/n/?]
Parallel port support (CONFIG_PARPORT) [**N**/y/?]

[omission: specific to user preference]

Plug and Play support
Plug and Play support (CONFIG_PNP) [**N**/y/?]
Block devices
Normal PC floppy disk support (CONFIG_BLK_DEV_FD) [**Y**/n/?]
Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support (CONFIG_BLK_DEV_IDE) [**Y**/n/?]

[omission: specific to hardware configuration]

Additional Block Devices
Loopback device support (CONFIG_BLK_DEV_LOOP) [**N**/y/m/?]
Network block device support (CONFIG_BLK_DEV_NBD) [**N**/y/m/?]

[omission: specific to hardware configuration]

Networking options
Packet socket (CONFIG_PACKET) [Y/**n**/?]
Kernel/User netlink socket (CONFIG_NETLINK) [**N**/y/?]
Network firewalls (CONFIG_FIREWALL) [N/**y**/?]
Socket Filtering (CONFIG_FILTER) [**N**/y/?]
Unix domain sockets (CONFIG_UNIX) [**Y**/n/?]
TCP/IP networking (CONFIG_INET) [**Y**/n/?]
IP: multicasting (CONFIG_IP_MULTICAST) [**N**/y/?]
IP: advanced router (CONFIG_IP_ADVANCED_ROUTER) [**N**/y/?]
IP: kernel level configuration support (CONFIG_IP_PNP) [**N**/y/?]
IP: firewalling (CONFIG_IP_FIREWALL) [N/**y**/?] **Y**
IP: firewall packet netlink device (CONFIG_IP_FIREWALL_NETLINK) [**N**/y/?]

```
IP: transparent proxy support (CONFIG_IP_TRANSPARENT_PROXY) [N/y/?]
IP: masquerading (CONFIG_IP_MASQUERADE) [N/y/?]
IP: optimize as router not host (CONFIG_IP_ROUTER) [N/Y/?]
IP: tunneling (CONFIG_NET_IPIP) [N/Y/?]
IP: GRE tunnels over IP (CONFIG_NET_IPGRE) [N/y/?]
IP: aliasing support (CONFIG_IP_ALIAS) [N/y/?]
IP: TCP syncookie support (not enabled per default) (CONFIG_SYN_COOKIES) [N/Y/?]
IP: Reverse ARP (CONFIG_INET_RARP) [N/y/?]
IP: Allow large windows (CONFIG_SKB_LARGE) [Y/n/?]
The IPX protocol (CONFIG_IPX) [N/y/?]
Appletalk DDP (CONFIG_ATALK) [N/y/?]
QoS and/or fair queueing (CONFIG_NET_SCHED) [N/y/?]

        [omission: specific to hardware configuration]

Network device support
Network device support (CONFIG_NETDEVICES) [Y/n/?]

        [omission: specific to hardware configuration]

Character devices
Virtual terminal (CONFIG_VT) [Y/n/?]
Support for console on virtual terminal (CONFIG_VT_CONSOLE) [Y/n/?]
Standard/generic (dumb) serial support (CONFIG_SERIAL) [Y/n/?]
Support for console on serial port (CONFIG_SERIAL_CONSOLE) [N/y/?]
Extended dumb serial driver options (CONFIG_SERIAL_EXTENDED) [N/y/?]
Non-standard serial port support (CONFIG_SERIAL_NONSTANDARD) [N/y/?]
Unix98 PTY support (CONFIG_UNIX98_PTYS) [Y/n/?]
Maximum number of Unix98 PTYs in use (0-2048) (CONFIG_UNIX98_PTY_COUNT) [128]
Mouse Support (not serial mice) (CONFIG_MOUSE) [Y/n/?]

        [omission: specific to hardware configuration]

Filesystems
Quota support (CONFIG_QUOTA) [N/y/?]
Kernel automounter support (CONFIG_AUTOFS_FS) [Y/n/?]
Amiga FFS filesystem support (CONFIG_AFFS_FS) [N/y/?]
Apple Macintosh filesystem support (experimental) (CONFIG_HFS_FS) [N/y/?]
DOS FAT fs support (CONFIG_FAT_FS) [N/Y/?]
        MSDOS fs support (CONFIG_MSDOS_FS) [N/Y/?]
        VFAT (Windows-95) fs support (CONFIG_VFAT_FS) [N/Y/?]
ISO 9660 CDROM filesystem support (CONFIG_ISO9660_FS) [Y/n/?]
Microsoft Joliet CDROM extensions (CONFIG_JOLIET) [N/Y/?]
Minix fs support (CONFIG_MINIX_FS) [N/y/?]
NTFS filesystem support (read only) (CONFIG_NTFS_FS) [N/y/?]
OS/2 HPFS filesystem support (read only) (CONFIG_HPFS_FS) [N/y/?]
/proc filesystem support (CONFIG_PROC_FS) [Y/n/?]
/dev/pts filesystem for Unix98 PTYs (CONFIG_DEVPTS_FS) [Y/n/?]
ROM filesystem support (CONFIG_ROMFS_FS) [N/y/?]
Second extended fs support (CONFIG_EXT2_FS) [Y/n/?]
System V and Coherent filesystem support (CONFIG_SYSV_FS) [N/y/?]
UFS filesystem support (CONFIG_UFS_FS) [N/y/?]
Network File Systems
Coda filesystem support (advanced network fs) (CONFIG_CODA_FS) [N/y/?]
NFS filesystem support (CONFIG_NFS_FS) [Y/n/?]
SMB filesystem support (to mount WfW shares etc.) (CONFIG_SMB_FS) [N/y/?]
NCP filesystem support (to mount NetWare volumes) (CONFIG_NCP_FS) [N/y/?]
Partition Types
BSD disklabel (BSD partition tables) support (CONFIG_BSD_DISKLABEL) [N/y/?]
Macintosh partition map support (CONFIG_MAC_PARTITION) [N/y/?]
Minix subpartition support (CONFIG_MINIX_SUBPARTITION) [N/y/?]
SMD disklabel (Sun partition tables) support (CONFIG_SMD_DISKLABEL) [N/y/?]
Solaris (x86) partition table support (CONFIG_SOLARIS_X86_PARTITION) [N/y/?]

        [omission: specific to user preference]

Console drivers
VGA text console (CONFIG_VGA_CONSOLE) [Y/n/?]
Video mode selection support (CONFIG_VIDEO_SELECT) [N/y/?]

        [omission: specific to hardware configuration]
```

Security options
Non-executable user stack area (CONFIG_SECURE_STACK) [**Y**]
Restricted links in /tmp (CONFIG_SECURE_LINK) [**Y**]
Restricted FIFOs in /tmp (CONFIG_SECURE_FIFO) [**Y**]
Restricted /proc (CONFIG_SECURE_PROC) [**N**]
Special handling of fd 0, 1, and 2 (CONFIG_SECURE_FD_0_1_2) [**Y**]
Enforce RLIMIT_NPROC on execve(2) (CONFIG_SECURE_RLIMIT_NPROC) [**N**]
Destroy shared memory segments not in use (CONFIG_SECURE_SHM) [**N**]
Kernel hacking
Magic SysRq key (CONFIG_MAGIC_SYSRQ) [**N**/y/?]

APPENDIX B
OPENSSL CIPHER SPECIFICATIONS

Table B-1. SSL RSA Ciphers

Cipher-Tag	Protocol	Key Ex.	Auth.	Enc.	MAC	Type
DES-CBC3-SHA	SSLv3	RSA	RSA	3DES(168)	SHA1	
DES-CBC3-MD5	SSLv2	RSA	RSA	3DES(168)	MD5	
IDEA-CBC-SHA	SSLv3	RSA	RSA	IDEA(128)	SHA1	
RC4-SHA	SSLv3	RSA	RSA	RC4(128)	SHA1	
RC4-MD5	SSLv3	RSA	RSA	RC4(128)	MD5	
IDEA-CBC-MD5	SSLv2	RSA	RSA	IDEA(128)	MD5	
RC2-CBC-MD5	SSLv2	RSA	RSA	RC2(128)	MD5	
RC4-MD5	SSLv2	RSA	RSA	RC4(128)	MD5	
DES-CBC-SHA	SSLv3	RSA	RSA	DES(56)	SHA1	
RC4-64-MD5	SSLv2	RSA	RSA	RC4(64)	MD5	
DES-CBC-MD5	SSLv2	RSA	RSA	DES(56)	MD5	
EXP-DES-CBC-SHA	SSLv3	RSA(512)	RSA	DES(40)	SHA1	export
EXP-RC2-CBC-MD5	SSLv3	RSA(512)	RSA	RC2(40)	MD5	export
EXP-RC4-MD5	SSLv3	RSA(512)	RSA	RC4(40)	MD5	export
EXP-RC2-CBC-MD5	SSLv2	RSA(512)	RSA	RC2(40)	MD5	export
EXP-RC4-MD5	SSLv2	RSA(512)	RSA	RC4(40)	MD5	export
NULL-SHA	SSLv3	RSA	RSA	None	SHA1	
NULL-MD5	SSLv3	RSA	RSA	None	MD5	

Table B-2. SSL Diffie-Hellman Ciphers

Cipher-Tag	Protocol	Key Ex.	Auth.	Enc.	MAC	Type
ADH-DES-CBC3-SHA	SSLv3	DH	None	3DES(168)	SHA1	
ADH-DES-CBC-SHA	SSLv3	DH	None	DES(56)	SHA1	
ADH-RC4-MD5	SSLv3	DH	None	RC4(128)	MD5	
EDH-RSA-DES-CBC3-SHA	SSLv3	DH	RSA	3DES(168)	SHA1	
EDH-DSS-DES-CBC3-SHA	SSLv3	DH	DSS	3DES(168)	SHA1	
EDH-RSA-DES-CBC-SHA	SSLv3	DH	RSA	DES(56)	SHA1	
EDH-DSS-DES-CBC-SHA	SSLv3	DH	DSS	DES(56)	SHA1	
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES(40)	SHA1	export
EXP-EDH-DSS-DES-CBC-SHA	SSLv3	DH(512)	DSS	DES(40)	SHA1	export
EXP-ADH-DES-CBC-SHA	SSLv3	DH(512)	None	DES(40)	SHA1	export
EXP-ADH-RC4-MD5	SSLv3	DH(512)	None	RC4(40)	MD5	export

Table B-3. OpenSSL Cipher Specification Tags

Algorithm	Tag	Description
Key Exchange:		
	kRSA	RSA key exchange
	kDHr	Diffie-Hellman key exchange with RSA key
	kDHd	Diffie-Hellman key exchange with DSA key
	kEDH	Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)
Authentication:		
	aNULL	No authentication
	aRSA	RSA authentication
	aDSS	DSS authentication
	aDH	Diffie-Hellman authentication
Cipher Encoding:		
	eNULL	No encoding
	DES	DES encoding
	3DES	Triple-DES encoding
	RC4	RC4 encoding
	RC2	RC2 encoding
	IDEA	IDEA encoding
MAC Digest:		
	MD5	MD5 hash function
	SHA1	SHA1 hash function
	SHA	SHA hash function

Table B-4. OpenSSL Aliases

SSLv2	all SSL version 2.0 ciphers
SSLv3	all SSL version 3.0 ciphers
TLSv1	all TLS version 1.0 ciphers
EXP	all export ciphers
EXPORT40	all 40-bit export ciphers only
EXPORT56	all 56-bit export ciphers only
LOW	all low strength ciphers (no export, single DES)
MEDIUM	all ciphers with 128 bit encryption
HIGH	all ciphers using Triple-DES
RSA	all ciphers using RSA key exchange
DH	all ciphers using Diffie-Hellman key exchange
EDH	all ciphers using Ephemeral Diffie-Hellman key exchange
ADH	all ciphers using Anonymous Diffie-Hellman key exchange
DSS	all ciphers using DSS authentication
NULL	all ciphers using no encryption

APPENDIX C
APACHE WEB SERVER CONFIGURATION

```
##
## httpd.conf -- Apache HTTP server configuration file
##
# This is the main server configuration file. See URL http://www.apache.org/
# for instructions.
# Do NOT simply read the instructions in here without understanding
# what they do, if you are unsure consult the online docs. You have been
# warned.
# Originally by Rob McCool

# BindAddress: You can support virtual hosts with this option. This option
# is used to tell the server which IP address to listen to. It can either
# contain "*", an IP address, or a fully qualified Internet domain name.
# See also the VirtualHost directive.

BindAddress 192.168.6.1

# HostnameLookups: Log the names of clients or just their IP numbers
#   e.g.   www.apache.org (on) or 204.62.129.132 (off)
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on.

HostnameLookups off

# Port: The port the standalone listens to. For ports < 1023, you will
# need httpd to be run as root initially.

Port 443

# ServerRoot: The directory the server's config, error, and log files
# are kept in.
# NOTE!  If you intend to place this on a NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation,
# you will save yourself a lot of trouble.

ServerRoot /etc/httpd

# ServerType is either inetd, or standalone.
```

```
ServerType standalone

# User/Group: The name (or #number) of the user/group to run httpd as.

User httpsd
Group httpsd

# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.

ErrorLog /var/log/httpd/error_log

# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.

LogLevel warn

# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Please read the file README.DSO in the Apache 1.3 distribution for more
# details about the DSO mechanism and run `httpd -l' for the list of already
# built-in (statically linked and thus always available) modules in your httpd
# binary.
#
# Example:
# LoadModule foo_module libexec/mod_foo.so
#
# Documentation for modules is in "/home/httpd/manual/mod" in HTML format.

LoadModule env_module          modules/mod_env.so
LoadModule config_log_module   modules/mod_log_config.so
LoadModule agent_log_module    modules/mod_log_agent.so
LoadModule referer_log_module  modules/mod_log_referer.so
LoadModule mime_module         modules/mod_mime.so
LoadModule negotiation_module  modules/mod_negotiation.so
LoadModule status_module       modules/mod_status.so
LoadModule info_module         modules/mod_info.so
LoadModule includes_module     modules/mod_include.so
LoadModule dir_module          modules/mod_dir.so
LoadModule cgi_module          modules/mod_cgi.so
LoadModule asis_module         modules/mod_asis.so
LoadModule imap_module         modules/mod_imap.so
```

```
LoadModule action_module          modules/mod_actions.so
LoadModule userdir_module         modules/mod_userdir.so
LoadModule proxy_module           modules/libproxy.so
LoadModule alias_module           modules/mod_alias.so
LoadModule rewrite_module         modules/mod_rewrite.so
LoadModule access_module          modules/mod_access.so
LoadModule auth_module            modules/mod_auth.so
LoadModule anon_auth_module       modules/mod_auth_anon.so
LoadModule db_auth_module         modules/mod_auth_db.so
LoadModule digest_module          modules/mod_digest.so
LoadModule expires_module         modules/mod_expires.so
LoadModule headers_module         modules/mod_headers.so
LoadModule usertrack_module       modules/mod_usertrack.so
LoadModule setenvif_module        modules/mod_setenvif.so

# Extra Modules
LoadModule perl_module            modules/libperl.so
LoadModule external_auth_module   modules/mod_auth_external.so

LoadModule ssl_module             /usr/lib/apache/libssl.so

#  Reconstruction of the complete module list from all available modules
#  (static and shared ones) to achieve correct module execution order.
#  [WHENEVER YOU CHANGE THE LOADMODULE SECTION ABOVE
UPDATE THIS, TOO]

ClearModuleList

AddModule mod_env.c
AddModule mod_log_config.c
AddModule mod_log_agent.c
AddModule mod_log_referer.c
AddModule mod_mime.c
AddModule mod_negotiation.c
AddModule mod_status.c
AddModule mod_info.c
AddModule mod_include.c
AddModule mod_dir.c
AddModule mod_cgi.c
AddModule mod_asis.c
AddModule mod_imap.c
AddModule mod_actions.c
AddModule mod_userdir.c
AddModule mod_proxy.c
AddModule mod_alias.c
AddModule mod_rewrite.c
```

```
AddModule mod_access.c
AddModule mod_auth.c
AddModule mod_auth_anon.c
AddModule mod_auth_db.c
AddModule mod_auth_external.c
AddModule mod_digest.c
AddModule mod_expires.c
AddModule mod_headers.c
AddModule mod_usertrack.c
AddModule mod_so.c
AddModule mod_setenvif.c

# Extra Modules
AddModule mod_perl.c

AddModule mod_ssl.c
AddExternalAuth pwauth /usr/lib/apache/pwauth
SetExternalAuthMethod pwauth pipe

##############################################################
##  SSL Global Context Configuration
##
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts
##       (unless overridden by virtual hosts)
##

<IfModule mod_ssl.c>

##  SSL Support
##  When we also provide SSL we have to listen to the
##  standard HTTPS port - 443
##
Listen 192.168.1.32:443

SSLEngine off

SSLPassPhraseDialog  builtin

SSLSessionCache  dbm:/var/log/ssl/ssl_scache

SSLSessionCacheTimeout  300

SSLMutex  file:logs/ssl_mutex

SSLRandomSeed startup file:/dev/urandom 512
```

SSLRandomSeed connect builtin

SSLCipherSuite
ALL:!ADH:!NULL:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP

SSLLog /var/log/ssl/ssl_engine_log

SSLLogLevel info

```
CustomLog /var/log/ssl/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

SSLProtocol all
</IfModule>
#
END OF SSL GLOBAL CONTEXT CONFIGURATION
##

The LockFile directive sets the path to the lockfile used when Apache
is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or
USE_FLOCK_SERIALIZED_ACCEPT. This directive should normally be left at
its default value. The main reason for changing it is if the logs
directory is NFS mounted, since the lockfile MUST BE STORED ON A LOCAL
DISK. The PID of the main server process is automatically appended to
the filename.
#
LockFile /var/lock/httpd.lock

PidFile: The file the server should log its pid to
PidFile /var/run/httpd.pid

ScoreBoardFile: File used to store internal server process information.
Not all architectures require this. But if yours does (you'll know because
this file is created when you run Apache) then you *must* ensure that
no two invocations of Apache share the same scoreboard file.
ScoreBoardFile /var/run/httpd.scoreboard

UseCanonicalName: (new for 1.3) With this setting turned on, whenever
Apache needs to construct a self-referencing URL (a url that refers back
to the server the response is coming from) it will use ServerName and
Port to form a "canonical" name. With this setting off, Apache will
use the hostname:port that the client supplied, when possible. This
also affects SERVER_NAME and SERVER_PORT in CGIs.
UseCanonicalName off

```
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\"" combined_virtual
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# The location of the access file (Common Logfile Format).
# If this does not start with /, ServerRoot is prepended to it.
# CustomLog logs/access_log common
CustomLog /var/log/httpd/access_log combined_virtual

# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.

KeepAlive On

# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We reccomend you leave this number high, for maximum performance.

MaxKeepAliveRequests 100

# KeepAliveTimeout: Number of seconds to wait for the next request

KeepAliveTimeout 15

# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# Unix with it as it spirals down...

MaxClients 150

# MaxRequestsPerChild: the number of requests each child process is
#  allowed to process before the child dies.
#  The child will exit so as to avoid problems after prolonged use when
#  Apache (and maybe the libraries it uses) leak.  On most systems, this
#  isn't really needed, but a few (such as Solaris) do have notable leaks
#  in the libraries.
```

```
MaxRequestsPerChild 100

# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e. use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.

ServerName destinationearth.net

# Server-pool size regulation.  Rather than making you guess how many
# server processes you need, Apache dynamically adapts to the load it
# sees --- that is, it tries to maintain enough server processes to
# handle the current load, plus a few spare servers to handle transient
# load spikes (e.g., multiple simultaneous requests from a single
# Netscape browser).

# It does this by periodically checking how many servers are waiting
# for a request.  If there are fewer than MinSpareServers, it creates
# a new spare.  If there are more than MaxSpareServers, some of the
# spares die off.  These values are probably OK for most sites ---

MinSpareServers 8
MaxSpareServers 20

# Number of servers to start --- should be a reasonable ballpark figure.

StartServers 10

# Timeout: The number of seconds before receives and sends time out

Timeout 300

# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.

DocumentRoot /home/httpsd/configuration/html

# AccessFileName: The name of the file to look for in each directory
# for access control information.

AccessFileName .htaccess
```

```
# DefaultType is the default MIME type for documents which the server
# cannot find the type of from filename extensions.

DefaultType text/plain

AddType application/x-x509-ca-cert .crt

# The following directives disable keepalives and HTTP header flushes.
# The first directive disables it for Netscape 2.x and browsers which
# spoof it. There are known problems with these.
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properly
# support keepalive when it is used on 301 or 302 (redirect) responses.

BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

# AddHandler allows you to map certain file extensions to "handlers",
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
# Format: AddHandler action-name ext1

# To use CGI scripts:
AddHandler cgi-script .cgi
AddHandler server-parsed .shtml


#=============================================================
# Aliases for primary secure web site and secure virtual domains
#=============================================================

NameVirtualHost 192.168.6.1:443

<VirtualHost 192.168.6.1:443>

        ServerName secure.destinationearth.net
        ServerAlias destinationearth.net www.destinationearth.net

        # primary content

         DocumentRoot        /home/httpsd/configuration/html
        ScriptAlias /cgi-bin /home/httpsd/configuration/cgi-bin

        # alias for Apache icons

        Alias /icons/ /var/www/icons/
```

```
        # SSL Directives
        SSLEngine on
        SSLCertificateFile /etc/ssl/keys/servercrt.crt
        SSLCertificateKeyFile /etc/ssl/keys/serverrsa.key
        # End SSL Directives

        SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0

</VirtualHost>

#=============================================================
# End of aliases
#=============================================================

# First, we configure the "default" to be a very restrictive set of
# permissions.

<Directory />
   Options None
   AllowOverride None
   order deny,allow
   deny from all
   allow from none
</Directory>

# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.

#-------------------------------------------------------------
# primary directories
#-------------------------------------------------------------


<Directory /home/httpsd/configuration/html>
   SSLRequireSSL
   SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
   Options Includes
   AllowOverride None
   order deny,allow
   deny from all
   allow from all
   AuthName "trial access"
```

```
    AuthType Basic
    AuthExternal pwauth
    require valid-user
</Directory>

<Directory /var/www/icons>
    Options Includes
    AllowOverride None
    order deny,allow
    deny from all
    allow from all
</Directory>

<Directory /home/httpsd/configuration/cgi-bin>
    SSLRequireSSL
    SSLRequire %{SSL_CIPHER_USEKEYSIZE} >= 128
    Options ExecCGI
    AllowOverride None
    order deny,allow
    deny from all
    allow from all
</Directory>

#=============================================================
# End http.conf Configuration
#=============================================================
```

APPENDIX D
FIREWALL CONFIGURATION SCRIPT

IPChains firewall

/etc/firewall.conf

```
# Loopback Device
LOOPBACKIF="lo"
LOOPBACKIP="auto"

# External (Public) Interface
EXTIF="eth0"
EXTIP="auto"
EXTBROAD="auto"

# Default Gateway
DEFAULTGW="auto"

# Internal (Private) Interface
INTIF="eth1"
INTIP="auto"
INTNM="255.255.255.0"
INTLAN="192.168.6.0/24"

# Other reserved and well-known addresses
ANYWHERE="any/0"
UNIVERSE="0.0.0.0/0"
LOOPBACK="127.0.0.0/0"
CLASS_A="10.0.0.0/8"
CLASS_B="172.16.0.0/12"
CLASS_C="192.168.0.0/24"
CLASS_D_MULTICAST="224.0.0.0/4"
CLASS_E_RESERVED="240.0.0.0/5"
BROADCAST_SRC="0.0.0.0"
BROADCAST_DEST="255.255.255.255"

# Ports range from 0:65535
PRIVPORTS="0:1023"
UNPRIVPORTS="1024:65535"

# Ports for specific services
TRACEROUTE_SRC_PORTS="32769:65535"
TRACEROUTE_DEST_PORTS="33434:33523"
SSH_PORTS="1020:1023"

# Enable kernel IP Forwarding
IP_FORWARD="Y"

# IPSec Gateway
## Space separated list of remote gateways
IPSECSG="209.33.129.114 24.12.43.211"

## Space separated list of virtual interfaces for FreeS/Wan IPSEC
## implementation.  Only include those that are actually used.  If
## you want to limit the traffic that flows through the tunnels, you
## you must manually edit the rules that are put in place using this
## variable in /etc/rc.d/rc.firewall
FREESWANVI="ipsec0 ipsec1"


# Settings for internal Interface
```

```
    ## Allow open access on the internal interface.  If you answer "N" and you
    ## want to allow some machines access to the internal interface or access to
    ## some services on the internal interface, you must edit /etc/rc.d/rc.firewall
    ## and enter rules in the INTERNAL_UNSECURE section.  There are examples in the
    ## file
    INTERNAL_UNSECURE="N"

    INTERNAL_SECURE="192.168.6.12"    ## Administrator workstation IP address


    # Settings for External Interface

    ## Does the external interface use DHCP/BOOTP to get its address
    EXTERNAL_DYNAMICIP="N"

    ## This machine is a public SSH server
    EXTERNAL_SERVICES_SSH="Y"
```

/etc/rc.d/rc.firewall

```
#!/bin/sh
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# This configuration assumes the following:
#
#       1) The external interface is running on "eth0"
#       2) The external IP address is statically assigned
#       3) The internal IP network interface is "eth1"
#       4) The internal network is addressed within the private
#           192.168.6.x TCP/IP addressing scheme per RFC1918
#
#   ****
#   NOTE:  All 2.2.x Linux kernels prior to 2.2.11 have a fragmentation
#   ****   bug that renders all strong IPCHAINS rulesets void.  It
#          is CRITICAL that users upgrade the Linux kernel to 2.2.11+
#          for proper firewall security.
#
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Initializing
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
echo -e "\n\nLoading IPCHAINS Firewall"
echo "------------------------------------------------------------"

# Load the configuration
[ -f /etc/firewall.conf ] || exit 0
. /etc/firewall.conf

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Network Parameters
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# The loopback interface and address
#---------------------------------------------------------------------
  if [ "$LOOPBACKIP" = "auto" ]; then
     echo -e "- Auto detecting loopback interface IP..."
     LOOPBACKIP=`/sbin/ifconfig | grep -A 4 $LOOPBACKIF | awk '/inet/ { print $2 } ' |
sed -e s/addr://`
     echo $LOOPBACKIP
  fi
#---------------------------------------------------------------------


# External interface device.
#
# eg:   EXTIP="100.200.0.212"
#---------------------------------------------------------------------
  if [ "$EXTIP" = "auto" ]; then
     echo -e "- Auto detecting External interface IP..."
```

```
     EXTIP=`/sbin/ifconfig | grep -A 4 $EXTIF | awk '/inet/ { print $2 } ' | sed -e
s/addr://`
     echo $EXTIP
  fi
#--------------------------------------------------------------------


# Broadcast address of the external network
#
# eg:   EXTBROAD="100.200.0.255"
#--------------------------------------------------------------------
  if [ "$EXTBROAD" = "auto" ]; then
     echo -e "- Auto detecting External broadcast address..."
     EXTBROAD=`/sbin/ifconfig | grep -A 1 $EXTIF | awk '/Bcast/ { print $3 }' | sed -e
s/Bcast://`
     echo $EXTBROAD
  fi
#--------------------------------------------------------------------


# Gateway for the external network
#
# eg:   DGW="100.200.0.1"
#--------------------------------------------------------------------
  if [ "$DEFAULTGW" = "auto" ]; then
     echo -e "- Auto detecting default gateway...."
     DEFAULTGW=`/sbin/route  | grep default | awk '{ print $2}'`
     echo $DEFAULTGW
  fi
#--------------------------------------------------------------------


# IP address on the internal interface
#--------------------------------------------------------------------
  if [ "$INTIP" = "auto" ]; then
     echo -e "- Auto detecting Internal interface IP..."
     INTIP=`/sbin/ifconfig | grep -A 4 $INTIF | awk '/inet/ { print $2 } ' | sed -e
s/addr://`
     echo $INTIP
  fi
#--------------------------------------------------------------------


# Netmask of internal interface to calculate internal network
#--------------------------------------------------------------------
  if [ "$INTNM" = "auto" ]; then
     echo -e "- Auto detecting internal netmask..."
     INTNM=`/sbin/ifconfig | grep -A 4 $INTIF | awk '/inet/ { print $4 } ' | sed -e
s/Mask://`
     echo $INTNM
  fi
#--------------------------------------------------------------------


# IP network address of the internal network
#--------------------------------------------------------------------
  if [ "$INTLAN" = "auto" ]; then
     echo -e "- Auto calculating address of internal network..."
     INTLAN=`/bin/ipcalc --network $INTIP $INTNM | sed -e s/NETWORK=//`
     echo $INTLAN
  fi
#--------------------------------------------------------------------


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# IPCHAINS Logging
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# The output of this logging can be found in the /var/log/messages
# file.  If you need to reduce some of the logging, edit the rulesets and
# delete the "$LOGGING" syntax from the ruleset that you aren't
# interested in.
```

```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

LOGGING="-l"


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Configuration Logging
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Log all configuration setting to a file.
# This script will dump all configuration
# settings to a file called
# /tmp/rc.firewall.dump
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "Writing /tmp/rc.firewall.dump."
rm -f /tmp/rc.firewall.dump
echo Local TCP/IP Configuration from rc.firewall > /tmp/rc.firewall.dump
echo ------------------------------------------------- >> /tmp/rc.firewall.dump
echo Loopback interface name: $LOOPBACKIF >> /tmp/rc.firewall.dump
echo Loopback IP: $LOOPBACKIP >> /tmp/rc.firewall.dump
echo ------------------------------------------------- >> /tmp/rc.firewall.dump
echo Internal interface name: $INTIF >> /tmp/rc.firewall.dump
echo Internal interface IP: $INTIP >> /tmp/rc.firewall.dump
echo Internal LAN address: $INTLAN >> /tmp/rc.firewall.dump
echo ------------------------------------------------- >> /tmp/rc.firewall.dump
echo External interface name: $EXTIF >> /tmp/rc.firewall.dump
echo External interface IP: $EXTIP >> /tmp/rc.firewall.dump
echo External interface broadcast IP: $EXTBROAD >> /tmp/rc.firewall.dump
echo _____________________________________________________ >> /tmp/rc.firewall.dump
echo Default gateway/route: $DEFAULTGW >> /tmp/rc.firewall.dump
echo ------------------------------------------------- >> /tmp/rc.firewall.dump


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Output for TCP/IP Configuration
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Output TCP/IP Configuration from rc.firewall
# to standard out
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo Local TCP/IP Configuration from rc.firewall
echo -----------------------------------------------------
echo Loopback interface name: $LOOPBACKIF
echo Loopback IP: $LOOPBACKIP
echo -----------------------------------------------------
echo Internal interface name: $INTIF
echo Internal interface IP: $INTIP
echo Internal LAN address: $INTLAN
echo -----------------------------------------------------
echo External interface name: $EXTIF
echo External interface IP: $EXTIP
echo External interface broadcast IP: $EXTBROAD
echo _____________________________________________________
echo Default gateway/route: $DEFAULTGW
echo -----------------------------------------------------


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# General
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Performs general processing
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#-------------------------------------------------------------------
# Enable IP Forwarding
#
#The net.ipv4.ip_forward parameter should be set to one to enable IP
# masquerading of Internet requests from the internal network.  This
# protects computers on the internal network so all requests appear to
# be coming from one and only one host.  In additions, all computers on
# the internal network have access to Internet resources without
```

```
# becoming vulnerable to attack.
#------------------------------------------------------------------
if [ "$IP_FORWARD" = "Y" ]; then
  echo "  - Enabling IP forwarding."
  echo "1" > /proc/sys/net/ipv4/ip_forward
fi
#------------------------------------------------------------------


#------------------------------------------------------------------
# Disable IP spoofing attacks.
#
# This drops traffic addressed for one network though it is being received on a
# different interface.
#------------------------------------------------------------------
echo "  - Disabling IP Spoofing attacks."
for file in /proc/sys/net/ipv4/conf/*/rp_filter
do
 echo "1" > $file
done
#------------------------------------------------------------------


#------------------------------------------------------------------
# Ignore all ICMP requests
#
# In order to prevent ping flooding, a basic form of a Denial of Service (DoS)
# attack where an IP address is flooded with ICMP requests to effectively
# close down any services provided on that IP address. The
# net.ipv4.icmp_ip_echo_ignore_all parameter should be set to one so the
# server will ignore all ICMP (ping) requests.  Ignoring an ICMP request is
# considerably better than simply denying the request-no response from
# an IP address typically indicates no link at that address, but a denied
# response is an acknowledgement to the sender that a link does exist.
#------------------------------------------------------------------
#echo " - Ignoring all ICMP requests."
#echo "1" > /proc/sys/net/ipv4/icmp_ip_echo_ignore_all
#------------------------------------------------------------------


#------------------------------------------------------------------
# Ignore all broadcast requests
#
# prevent intentional or unintentional ping flooding.  When a packet is
# sent to the network broadcast address, it is sent to all machines on that
# network.  The machines on the network then respond to that request and
# the responses can result in network congestion or a DoS attack.  Most
# routers, as a rule, will not forward packets destined for a broadcast
# address but an exception to that rule most certainly exists.  By ignoring
# broadcast ICMP requests, the Linux server will be less apt to
# suffer from ping attacks.
#------------------------------------------------------------------
echo " - Ignoring all broadcast requests."
echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
#------------------------------------------------------------------


#------------------------------------------------------------------
# Disable ICMP Redirect Acceptance
#
# When a host uses a non-optimal or stale route to a destination an
# ICMP redirect packet is returned by the routers to inform the host
# what the correct route should be.  If an attacker is able to forge ICMP
# redirect packets, he or she may be able to alter the routing table on
# the host.  By altering the host's routing table, the malevolent may
# gain access to sensitive traffic by diverting traffic via another path.
# In order to prevent such an attack, set the
# net.ipv4.conf.all.accept_all_redirects equal to zero.
#------------------------------------------------------------------
echo " - Disabling ICMP Redirect Acceptance."
for file in /proc/sys/net/ipv4/conf/*/accept_redirects
```

```
do
 echo "0" > $file
done

for file in /proc/sys/net/ipv4/conf/*/send_redirects
 do
  echo 0 > $file
 done
#------------------------------------------------------------------


#------------------------------------------------------------------
# Disable IP source routing
#
# Often, attackers will create packets with a source address different
# from their actual IP address.  This is known as IP spoofing and it is
# common to protect the attacker's identity in DoS attacks.  The attacker
# does not require a reply since the real intention is to deny access to
# any services the destination computer offers.  In other cases, the
# attacker sends a packet with a source address of a trusted internal IP
# to the external IP of a gateway host. The gateway host, if not prepared,
# will read the packet as an internal secure communication and may allow
# the attacker into the system.  In order to combat IP spoofing attacks,
# setting the Linux kernel parameter net.ipv4.conf.all.rp_filter to one
# enables spoofing protection thereby peventing the server from being
# the source of spoofed communications.
#------------------------------------------------------------------
echo " - Disabling IP source routing."
for file in /proc/sys/net/ipv4/conf/*/accept_source_route
do
 echo "0" > $file
done
#------------------------------------------------------------------


#------------------------------------------------------------------
# Enable automatic defragmentation
#
# Since not all information sent to the host can always be contained in a
# single packet, often data is fragmented into multiple packets.  The problem
# with this is that only the first packet contains the port numbers of the
# communication pipe.  It is possible to insert potentially harmful information
# into the remaining packets.  In a normal configuration, fragments are
# passed through the interface and reassembled later-sometimes with an
# attacker's code.  By enabling IP defragmentation, the packets are
# reassembled and put into the correct order, discarding any extraneous
# packets that may have been added.
#------------------------------------------------------------------
echo " - Enabling automatic defragmentation."
echo "1" > /proc/sys/net/ipv4/ip_always_defrag
#------------------------------------------------------------------


#------------------------------------------------------------------
# Enable TCP SYN cookie protection
#
# A more serious DoS attack than ping flooding is the SYN attack.
# Traditionally, systems would employ a SYN-ACK handshake to confirm
# a TCP connection and move it from the waiting queue.  If the SYN-ACK
# handshake does not complete, an attacker makes a request but
# ignores the returned packet with the SYN bit set, the server would wait
# for a long period for the ACK packet that never comes.  While waiting
# for the packet with the ACK bit set, the server would not move the
# request from the wait queue until a timeout occurs and would allow
# no one else to connect.  An attacker could take advantage of this by
# continuing to make requests and thereby denying anyone else access
# to the server.  To protect the server, the ip.ipv4.tcp_syncookies
# parameter should be set to one so a SYN attack cannot take place.
#------------------------------------------------------------------
echo " - Enabling TCP SYN cookie protection."
echo "1" > /proc/sys/net/ipv4/tcp_syncookies
```

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


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Default Policies
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Change default policies to DENY input and reject all other traffic REJECT.
# We want to only EXPLICTLY allow what traffic is allowed IN and OUT of the
# firewall.  All other traffic will be implicitly blocked.
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
echo "  - Flushing all old rules and setting all default policies to DENY and REJECT "

/sbin/ipchains -P input DENY
/sbin/ipchains -P output REJECT
/sbin/ipchains -P forward REJECT

/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Loopback interface policy
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Enable all traffic on the loopback interface
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "-----------------------------------------------------------------------"
echo "    - Allow all traffic from loopback interface $LOOPBACKIP"
/sbin/ipchains -A input -j ACCEPT -i $LOOPBACKIF -s $UNIVERSE -d $UNIVERSE
/sbin/ipchains -A output -j ACCEPT -i $LOOPBACKIF -s $UNIVERSE -d $UNIVERSE


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# IPSec VPN
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# If you are using an IPSec VPN product, you will need to fill in the
# addresses of the gateways in the IPSECSG
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# Disable IP spoofing protection to allow IPSEC to work properly
# ------------------------------------------------------------------------------
echo "0" > /proc/sys/net/ipv4/conf/ipsec0/rp_filter
echo "0" > /proc/sys/net/ipv4/conf/eth0/rp_filter

for CURGW in $IPSECSG; do
  echo "    - Allow ISAKMP from $IPSECSG to external interface $EXTIF"
  ipchains -A input -j ACCEPT -i $EXTIF -p udp -s $CURGW isakmp
  ipchains -A output -j ACCEPT-i $EXTIF -p udp -d $CURGW isakmp
  echo "    - Allow IPSEC protocol from $IPSECSG on external interface $EXTIF"
  ipchains -A input -j ACCEPT -i $EXTIF -p 50  -s $CURGW
  ipchains -A output -j ACCEPT-i $EXTIF -p 50 -d $CURGW
  ipchains -A input -j ACCEPT-i $EXTIF -p 51 -s $CURGW
  ipchains -A output -j ACCEPT-i $EXTIF -p 51 -d $CURGW
done

for CURIF in $FREESWANVI; do
  echo "    - Allow all traffic to FreeS/WAN Virtual Interface $CURIF"
  ipchains -A input -j ACCEPT -i $CURIF -s $UNIVERSE -d $UNIVERSE
done

for CURIF in $FREESWANVI; do
  echo "    - Allow all trafic from FreeS/WAN virtual interface $CURIF"
  ipchains -A output -j ACCEPT -i $CURIF -s $UNIVERSE -d $UNIVERSE
done

for CURIF in $FREESWANVI; do
   echo "  - Forward anything from FreeS/WAN virtual interface $CURIF"
  /sbin/ipchains -A forward -j ACCEPT -i $CURIF -s $UNIVERSE -d $UNIVERSE
done
```

```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# OpenSSH
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# SSH policies between IPSEC Gateways
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "  - Setting OpenSSH input filters for specific internal hosts."
/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $INTERNAL_SECURE -d $INTIP ssh

echo "  - Setting OpenSSH output filters for specific internal hosts."
/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $INTERNAL_SECURE -d $INTIP ssh

# SSH server: Allow specific IPSEC computers to connect to the Linux server ITSELF
#  for SSH access.
# ------------------------------------------------------------------------------
if [ "$EXTERNAL_SERVICES_SSH" = "Y" ]; then
  echo "    - Allow SSH to external interface $EXTIF"
  /sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $IPSECSG $UNPRIVPORTS -d $EXTIP
ssh
  /sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp ! -y -s $EXTIP ssh -d $IPSECG
$UNPRIVPORTS
  /sbin/ipchains -A input -j ACCEPT -i $EXTIF -p tcp -s $IPSECSG $SSH_PORTS -d $EXTIP ssh
  /sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp ! -y -s $EXTIP ssh -d $IPSECG
$SSH_PORTS
fi


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Secure HTTP (https)
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# HTTPS policy for access from internal LAN
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "  - Setting HTTPS input filters for specific internal hosts."
/sbin/ipchains -A input -j ACCEPT -i $INTIF -p tcp -s $INTERNAL_SECURE -d $INTIP https

echo "  - Setting HTTPS output filters for specific internal hosts."
/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $INTERNAL_SECURE -d $INTIP https


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# DHCP Server
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#  If you have configured a DHCP server on this Linux machine, you
#  will need to enable the following ruleset.
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "  - Allowing DHCP requests on internal network."
/sbin/ipchains -A output -j ACCEPT -i $INTIF -p udp -s $INTIP/32 bootps -d $BROADCAST/0
bootpc
/sbin/ipchains -A output -j ACCEPT -i $INTIF -p tcp -s $INTIP/32 bootps -d $BROADCAST/0
bootpc


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Incoming Traffic from the External Interface
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# This ruleset will control specific traffic that is allowed in from
# the external interface.
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "------------------------------------------------------------------------"
echo "  - Setting input filters for traffic from the external interface."

# ------------------------------------------------------------------------------
# SPOOFING & BAD ADDRESSES
# Refuse spoofed packets.
# Ignore blatantly illegal source addresses.
# Protect yourself from sending to bad addresses.
# ------------------------------------------------------------------------------
```

```
# Remote interface, claiming to be external IP, IP spoofing
# Remote interface, claiming to be local machines, IP spoofing, get lost & log
# ------------------------------------------------------------------------
echo "    - Reject and log spoofing."
/sbin/ipchains -A input -j DENY -i $EXTIF -s $EXTIP -d $UNIVERSE $LOGGING
/sbin/ipchains -A input -j DENY -i $EXTIF -s $INTLAN -d $UNIVERSE $LOGGING

# Refuse incoming packets claiming to be from a Class A, B or C private network
# ------------------------------------------------------------------------
/sbin/ipchains -A input -j DENY -i $EXTIF -s $CLASS_A -d $UNIVERSE $LOGGING
/sbin/ipchains -A input -j DENY -i $EXTIF -s $CLASS_B -d $UNIVERSE $LOGGING
/sbin/ipchains -A input -j DENY -i $EXTIF -s $CLASS_C -d $UNIVERSE $LOGGING
/sbin/ipchains -A input -j DENY -i $EXTIF -s $UNIVERSE -d $CLASS_A $LOGGING
/sbin/ipchains -A input -j DENY -i $EXTIF -s $UNIVERSE -d $CLASS_B $LOGGING
/sbin/ipchains -A input -j DENY -i $EXTIF -s $UNIVERSE -d $CLASS_C $LOGGING

# Refuse outgoing packets claiming to be from a Class A, B, or C private network
# ------------------------------------------------------------------------
/sbin/ipchains -A output -j DENY -i $EXTIF -s $CLASS_A -d $UNIVERSE $LOGGING
/sbin/ipchains -A output -j DENY -i $EXTIF -s $CLASS_B -d $UNIVERSE $LOGGING
/sbin/ipchains -A output -j DENY -i $EXTIF -s $CLASS_C -d $UNIVERSE $LOGGING
/sbin/ipchains -A output -j DENY -i $EXTIF -s $UNIVERSE -d $CLASS_A $LOGGING
/sbin/ipchains -A output -j DENY -i $EXTIF -s $UNIVERSE -d $CLASS_B $LOGGING
/sbin/ipchains -A output -j DENY -i $EXTIF -s $UNIVERSE -d $CLASS_C $LOGGING

# Refuse packets claiming to be from the loopback interface
# ------------------------------------------------------------------------
/sbin/ipchains -A input -j DENY -i $EXTIF -s $LOOPBACKIP -d $UNIVERSE $LOGGING
/sbin/ipchains -A ouput -j DENY -i $EXTIF -s $LOOPBACKIP -d $UNIVERSE $LOGGING

# Refuse broadcast address SOURCE packets
# ------------------------------------------------------------------------
/sbin/ipchains -A input -j DENY -i $EXTIF -s $BROADCAST_DEST -d $UNIVERSE $LOGGING
/sbin/ipchains -A input -j DENY -i $EXTIF -s $BROADCAST_SRC -d $UNIVERSE $LOGGING

# Refuse Class D multicast addresses
# Multicast is illegal as a source address.
# Multicast uses UDP.
# ------------------------------------------------------------------------
/sbin/ipchains -A input -j DENY -i $EXTIF -s $CLASS_D_MULTICAST -d $UNIVERSE $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -s $CLASS_D_MULTICAST -d $UNIVERSE $LOGGING
/sbin/ipchains -A output -j REJECT -i $EXTIF -d $CLASS_D_MULTICAST $LOGGING

# Refuse Class E reserved IP  addresses
# ------------------------------------------------------------------------
ipchains -A input -j DENY -s $CLASS_E_RESERVED_NET-d $UNIVERSE $LOGGING

# Refuse special addresses defined as reserved by the IANA.
# Note:  The remaining reserved addresses are not included.
# Filtering them causes problems as reserved blocks are
# being allocated more often now.
# ------------------------------------------------------------------------
ipchains -A input   -j DENY   -s 127.0.0.0/8        $LOGGING
ipchains -A input   -j DENY   -s 169.254.0.0/16     $LOGGING
ipchains -A input   -j DENY   -s 192.0.2.0/24       $LOGGING
ipchains -A input   -j DENY   -s 224.0.0.0/3        $LOGGING


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# ICMP
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#    To prevent denial of service attacks based on ICMP bombs, filter
#    incoming Redirect (5) and outgoing Destination Unreachable (3).
#    Note, however, disabling Destination Unreachable (3) is not
#    advisable, as it is used to negotiate packet fragment size.
#
# For bi-directional ping.
#     Message Types:  Echo_Reply (0),  Echo_Request (8)
#     To prevent attacks, limit the src addresses to your ISP range.
#
# For outgoing traceroute.
```

```
#     Message Types:  INCOMING Dest_Unreachable (3), Time_Exceeded (11)
#     default UDP base: 33434 to base+nhops-1
#
# For incoming traceroute.
#     Message Types:  OUTGOING Dest_Unreachable (3), Time_Exceeded (11)
#     To block this, deny OUTGOING 3 and 11
#
#  0: echo-reply (pong)
#  3: destination-unreachable, port-unreachable, fragmentation-needed, etc.
#  4: source-quench
#  5: redirect
#  8: echo-request (ping)
# 11: time-exceeded
# 12: parameter-problem
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "  -Setting ICMP filtering policies."

ipchains -A input -j ACCEPT -i $EXTIF -p icmp --icmp-type echo-reply -d $EXTIP
ipchains -A input -j ACCEPT-i $EXTIF -p icmp --icmp-type destination-unreachable -d
$EXTIP
ipchains -A input -j ACCEPT -i $EXTIF -p icmp --icmp-type source-quench -d $EXTIP
ipchains -A input -j ACCEPT -i $EXTIF -p icmp --icmp-type time-exceeded -d $EXTIP
ipchains -A input -j ACCEPT -i $EXTIF -p icmp --icmp-type parameter-problem -d $EXTIP
ipchains -A output -j ACCEPT -i $EXTIF -p icmp -s $EXTIP fragmentation-needed
ipchains -A output -j ACCEPT -i $EXTIF -p icmp -s $EXTIP source-quench
ipchains -A output -j ACCEPT -i $EXTIF -p icmp -s $EXTIP echo-request
ipchains -A output -j ACCEPT -i $EXTIF -p icmp -s $EXTIP parameter-problem


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Incoming Traffic on all Interfaces
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# This will control input traffic for all interfaces.  This is
# usually used for what could be considered as public services.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "------------------------------------------------------------------------"
echo "  - Setting input filters for public services (all interfaces)."

# AUTH: Allow the authentication protocol, ident, to function on all
#       interfaces but disable it in /etc/inetd.conf.  The reason to
#       allow this traffic in but block it via Inetd is because some
#       legacy TCP/IP stacks don't deal with REJECTed "auth" requests
#       properly.
#
echo "    - Allow AUTH on all interfaces"
/sbin/ipchains -A input -j ACCEPT -p tcp -s $UNIVERSE -d $UNIVERSE auth


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Outgoing Traffic on the External Interface
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# This ruleset will control what traffic can go out on the external interface.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

echo "------------------------------------------------------------------------"
echo "  - Setting output filters for traffic from the external interface."

# Reject outgoing traffic to the local net from the remote interface,
# stuffed routing; deny & log
# ------------------------------------------------------------------------
echo "    - Reject outgoing traffic from external interface $EXTIF to internal network
$INTLAN - Stuffed Routing"
/sbin/ipchains -A output -j REJECT -i $EXTIF -s $UNIVERSE -d $INTLAN $LOGGING

# Reject outgoing traffic from the local net from the external interface,
# stuffed masquerading, deny and log
# ------------------------------------------------------------------------
echo "    - Reject outgoing traffic from internal network $INTLAN to external interface
$EXTIF - Stuffed Masquerading"
```

```
/sbin/ipchains -A output -j REJECT -i $EXTIF -s $INTLAN -d $UNIVERSE $LOGGING

# AUTH: Allow authentication tap indent on all interfaces (but disable it
#       in /etc/inetd.conf).
#+ ---------------------------------------------------------------------------
echo "    - Allow ident traffic from external interface $EXTIF"
/sbin/ipchains -A output -j ACCEPT -i $EXTIF -p tcp -s $UNIVERSE auth -d $UNIVERSE


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Enable logging for selected denied packets
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    ipchains -A input  -j DENY -i $EXTIF -p tcp $LOGGING
    ipchains -A input -j DENY -i $EXTIF -p udp --destination-port $PRIVPORTS $LOGGING
    ipchains -A input -j DENY -i $EXTIF -p udp --destination-port $UNPRIVPORTS $LOGGING
    ipchains -A input -j DENY -i $EXTIF -p icmp --icmp-type 5 $LOGGING
    ipchains -A input -j DENY -i $EXTIF -p icmp --icmp-type 13:255 $LOGGING
    ipchains -A output -j REJECT-i $EXTIF $LOGGING


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Catch All INPUT Rule
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
echo "---------------------------------------------------------------------"
echo "  - Deny and log all input not specifically allowed."

# All other incoming is denied and logged.
/sbin/ipchains -A input -j DENY -s $UNIVERSE -d $UNIVERSE $LOGGING


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Catch All OUTPUT Rule
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
echo "---------------------------------------------------------------------"
echo "  - Reject and log all output not specifically alllowed"

# All other outgoing is denied and logged.  This ruleset should catch
# everything including samba that hasn't already been blocked.
#
/sbin/ipchains -A output -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Catch All FORWARDING Rule
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
echo "---------------------------------------------------------------------"
echo "Forwarding Rules:"
# Catch all rule, all other forwarding is denied.
#
echo "  - Deny and log anything not specifically allowed to be forwarded"
/sbin/ipchains -A forward -j REJECT -s $UNIVERSE -d $UNIVERSE $LOGGING


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Finish
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
echo "---------------------------------------------------------------------"
echo -e "Firewall implemented. \n\n"
```

LIST OF REFERENCES

Adams, R., & Erickson, C. (1999 August). Users and groups. CS 437 Distributed Computing, Grand Valley State University. Retrieved June 20, 2001 from http://www.csis.gvsu.edu/ class/cs437/Notes/OS/09users.html

Allen, M. (2000, June 1). How Linux works CTDP guide version 0.6.0. The Computer Technology Documentation Project. Retrieved June 17, 2001 from http://ctdp.tripod.com/os/linux/howlinuxworks/

Almesberger, W. (1998, December 4). LILO: generic boot loader for Linux. Version 21. Generic Books. Retrieved June 17, 2001 from http://genericbooks.com/Literature/ Documents/pdf/Books/lilo_tech.pdf

Ball, W., & Pitts, D. (1996). Red Hat Linux unleashed. SunSite: Sun Software, Information, and Technology Exchange. Retrieved July 27, 2001 from http://sunsite.net.edu.cn/ tutorials/linux/httoc.htm

Bandel, D. A. (2000, May). Linux security toolkit. Foster City, Califonria. M&T Books

Bauer, B. & Bowden, T. (2001). SuSE Linux - guide for geeks. Retrieved August 21, 2001 from www.bb-zone.com Web site: http://www.bb-zone.com/zope/bbzone/docs/slgfg

Beekmans, G. (2001). Linux from scratch version 3.0-rc2. Linux From Scratch Organization Web site. http://www.linuxfromscratch.org/view/3.0-rc2/

Boldt, A. (2000). Linux configuration help texts: help texts for kernels 2.2.x. Personal Web site. Retrieved from July 29, 2001 http://math-www.uni-paderborn.de/~axel/Configure.help-2.2

Boran, S. (1999, December 13). An overview of corporate information security: combining organisational, physical, and IT security. Network Security Library. GFI Software Ltd. Retrieved September 5, 2001 from http://secinf.net/info/policy/ coverstory19991213.html

Bowen, R., Coar, K. A. L., Grip-Jansson, P., Marlowe, M., & Chinnappan, M. (2000, March 9). Apache server unleashed. Indianapolis, Indiana. Sams

Brandt, K., Green, S., & Zúñiga, E. (2001 March). Cracker exploits: battle plans, 15 cracker exploits every security professional should know about-and how to defend against [Electronic version]. Information Security. Retrieved from Information Security Magazine: http://www.infosecuritymag.com/articles/march01/ features4_battle_plans.shtml

Carella, F. (1999, March). Understanding the Linux boot sequence. Sault College of Applied Arts and Technology, Sault Ste Marie, Ontario Canada. Retrieved June 17, 2001 from http://apollo.saultc.on.ca/~fcarella/PersonalHowtos/boot.slave/bootsequence.html

Cavalli, A. (1996, September 4). Dynamic virtual private networks: a security solution for enabling business intranets. TradeWave Corporation White Paper. Retrieved May 12, 2001 from: http://www.adimpleo.com/library/trwave/securvpn.pdf

Chae, L. (1998, October 1). Virtual private networks. Network Magazine. Retrieved May 12, 2001 from: http://www.networkmagazine.com/article/NMG20000727S0029

DeClario, N. (2001, September 19). Building a secure web server using Apache and OpenSSL. Linux Security.com. Retrieved June 24, 2001 from http://www.linuxsecurity.com/feature_stories/feature_story-67.html

Denker, J. S., Bellovin, S. M., Daniel, H., Mintz, N. L., Killian, T., & Plotnick, M. A. (1999, November 12). Moat: a virtual private network appliance and services platform. Paper presented at the 13th Systems Administration Conference. Retrieved May 25, 2001, from: http://www.usenix.org/publications/library/proceedings/lisa99/full_papers/denker/denker.pdf

Drake, J. (2000). Linux networking HOWTO - docbook rev .02. Linux Documentation Project. Commandprompt, Inc. Retrieved August, 16, 2000 from http://www.linuxdoc.org/HOWTO/Net-HOWTO/index.html

Drakos, N. (1997) Luvisetto, M. (2001, May 3 trans.).Linux administrator guide. Retrieved from the World Wide Web: http://www.bo.infn.it/alice/alice-doc/mll-doc/linux/admin/

Engelschall, R. S. & Ralf S. (2001). User manual: the Apache interface to OpenSSL. Mod_ssl homepage. Retrieved August 26, 2001 from http://www.modssl.org/docs/2.8/

Fenzi, K., & Wreski, D. (2000, March). Linux security HOWTO. Linux Documentation Project. Retrieved August 11, 2000 from http://www.linuxdoc.org/HOWTO/Security-HOWTO

Ferguson, N. & Schneier, B. (2000, January). A cryptographic evaluation of IPSec. http://citeseer.nj.nec.com/ferguson00cryptographic.html

Ferguson, P., & Huston, G. (1998, April). What is a VPN? The Internet Protocol Journal, Cisco, vol. 1, no. 1, June 1998

Fraser, B. (Ed). (1997, September). Site security handbook. Request For Comments 2196. Internet Network Internet Engineering Task Force (IETF) Working Group. Retrieved August 13, 2001 from http://www.faqs.org/rfcs/rfc2196.html

FreeS/WAN documentation. Retrieved May 26, 2001 from http://www.freeswan.org/freeswan_trees/freeswan-1.9/doc/HowTo.html

Global network provider innovates with VPN service. Packet Magazine Archives, Second Quarter 1998. Retrieved May 12, 2001, from Cisco's Packet Magazine Web site: http://www.cisco.com/warp/public/784/packet/april98/12.html

Gortmaker, P. (1999, May 5). Linux ethernet-HOWTO. Linux Documentation Project. Retrieved August 17, 2000 from http://www.linuxdoc.org/HOWTO/Ethernet-HOWTO.html

Gortmaker, P. (1999, May). The Linux bootprompt-HOWTO. Linux Documentation Project. Retrieved June 16, 2001 from: http://www.linuxdoc.org/HOWTO/BootPrompt-HOWTO.html

Grennan, M. (2000, February 26). Firewall and proxy server-HOWTO. Linux Documentation Project. Retrieved August 2, 2000 from http://www.linuxdoc.org/HOWTO/Firewall-HOWTO.html

Guttman, B. & Bagwill, R. (1997 July, 21). Internet security policy: a technical guide [DRAFT]. Information Technology Laboratory Computer Security Division. National Institute of Standards and Technology. Gaithersburg, MD 20899-0001. Retrieved August 23, 2001 from http://csrc.nist.gov/isptg/

Guttman, E., Leong, L., & Malkin, G. (1999, February). Users' security handbook. Request For Comments 2504. Internet Network Internet Engineering Task Force (IETF) Working Group. Retrieved August 13, 2001 from http://www.faqs.org/rfcs/ rfc2504.html

Hardin, J. D. (2000, October 22). Linux VPN masquerade howto. Linux Documentation Project. Retrieved August 11, 2000 from http://www.linuxdoc.org/HOWTO/VPN-Masquerade-HOWTO.html

Harris, T. & Koehntopp, K. (2000). Linux partition HOWTO. Linux Documentation Project. Retrieved August 11, 2000 from http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/ other-formats/html_single/Partition.html

InternetWeek Online. (n.d.). Exclusive vpn research. InternetWeek Online. Retrieved May 12, 2001 from: http://www.internetweek.com/VPN/812VPNcharts.htm

Ioannidis, S., Keromytis, A., Bellovin, S., & Smith, J. (2000). Implementing a distributed firewall. In Proceedings of Computer and Communications Security (CCS) 2000, November 2000. http://citeseer.nj.nec.com/ioannidis00implementing.html

Jackson, M. H. (1996, April 3). Linux shadow password HOWTO. Linux Documentation Project [no pagination]. Retrieved June 16, 2001 from http://linuxdocs.org/HOWTOs/Shadow-Password-HOWTO.html

Jackson, S., & Birdwell, J.D (1997, August). Definitive user access on a secure web server. Laboratory for Information Technologies. University of Tennessee. Retreived July 27, 2001 from Laboratory for Information Technologies Web site: http://www.lit.net/lit/ PDF/whosinit.pdf

Jordan, M. (n.d.). What is Linux? Linux Online. Retrieved June 23, 2001from http://www.linux.org/info/.

Kent, S. & Atkinson, R. (1998, November). Security architecture for the internet protocol. Technical Report Request for Comments 2401 IETF. Internet RFC/STD/FYI/BCP Archives. http://www.faqs.org/rfcs/rfc2401.html

Kerberos: the network authentication protocol (n.d.). Retrieved June 23, 2001 from the Massachusetts Institute of Technology Web site: http://web.mit.edu/kerberos/www/

Klein, D. V. (1990). Foiling the cracker: a survey of, and improvements to, password security. Technical report. Software Engineering Institute, Carnegie Mellon University. Retrieved May 12, 2000 from http://www.ja.net/CERT/Klein/D.Klein.Foiling.the.Cracker.txt

Knight, E. (2000, March 20). Computer vulnerabilities. Retrieved July 27, 2001 from Security Paradigm Web site: http://www.securityparadigm.com/compvuln_draft.pdf

Kozierok, C. M. (2001, April 17). The PC guide. Version: 2.2.0. The PC Guide home page. Retrieved May 19, 2001 from http://www.PCGuide.com.

Langfeldt, N. & Norrish, J. (2001, January 18). version 3.1. DNS HOWTO. Linux Documentation Project. Retrieved August 22, 2001 from http://www.ibiblio.org/pub/ Linux/docs/HOWTO/DNS-HOWTO.html

Lowes, M. (2001, April 21). v0.8.1. Professional FTP daemon FSQ. www.proftpd.org web site. Retrieved July 13, 2001 from http://ppd.sourceforge.net/faq/proftpdfaq-full.html

MacKenzie, D. (2001, May). Manpage of CHMOD. Free Software Foundation, Inc. Linux on-line manual pages

Magosanyi, A. (1997, August 7). The VPN HOWTO. Linux Documentation Project. Retrieved July 28, 2000 from http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/VPN.html

Mann, S. (n.d.). Limiting su to root with PAM. 101 Security Solutions. Retrieved July 21, 2001 from http://www.101.com/solutions/security/article.asp?ArticleID=528

Miller, D. (2001, February 4). OpenBSD reference manual SFTP: secure file transfer program. OpenBSD Manual Pages. Retrieved August 24, 2001 from http://www.openbsd.org/cgi-bin/man.cgi?query=sftp

Mourani, G & Madhusudan, M. (2000, June 7). Securing and optimizing Linux RedHat edition--a hands on guide. Open Network Architecture. Retrieved May 24, 2001 from http://www.openna.com

Netscape Communications Corporation. (1999). How SSL works. Security Developer Central. Retrieved September 3, 2001 from http://developer.netscape.com/tech/security/ssl/

Ng, Pheng Siong. (2001, March 31). HOWTO: Creating your own CA with OpenSSL. Revision 1.1. Personal Web site. Retrieved September 3, 2001 from http://www.pobox.org.sg/home/ngps/m2/howto.ca.html

O'Keefe, G. (2000, November). From power up to bash prompt. Personal Web site. http://www.netspace.net.au/~gok/power2bash/power2bash/power2bash.html.

OpenSSL core team. (n.d.). OpenSSL command line tool. OpenSSL project. Retrieved August 26, 2001 from http://www.openssl.org/docs/apps/openssl.html

Openwall readme. Retrieved September 2, 2001 from Openwall Web site: http://www.openwall.com/linux/README

Parker, T. (1999, December 20). Linux system administrator's survival guide, second edition. Sams

Peek, J. (2000, January 27). Protecting files with the sticky bit. O'Reilly network. Retrieved July 2, 2001 from O'Reilly Network: http://linux.oreillynet.com/pub/a/linux/lpt/ 22_06.html

The ProFTPD project: directive list. (2001, February 12). The ProFTPD Project. Retrieved July 13, 2001 from http://www.proftpd.net/docs/configuration.html

Rader, M., & Birdwell, J. D. (1997, August). Public/private/wireless information security: a blueprint for safeguarding sensitive information. Laboratory for Information Technologies. University of Tennessee. Retreived July 27, 2001 from Laboratory for Information Technologies Web site: http://www.lit.net/ondcp/wireless.html

Ramsey, P. (2000, June 22). Red Hat Linux 6.x as an internet gateway for a home network. Linux Documentation Project. Retrieved August 30, 2000 from http://www.ibiblio.org/ pub/Linux/docs/HOWTO/ mini/other-formats/html_single/Home-Network-mini-HOWTO.html

Ranch, D. A. (2001, June 10). TrinityOS: a guide to configuring your Linux server for performance, security, and manageability. Dranch's HomePage. Retrieved June 24, 2001 from http://www.ecst.csuchico.edu/~dranch/LINUX/TrinityOS/cHTML/TrinityOS-c.html

Ranch, D. A. (2001, June 10). v2.00.0610. Linux IP masquerade HOWTO. Linux Documentation Project. Retrieved July 29, 2001 from http://www.ibiblio.org/ pub/Linux/docs/HOWTO/IP-Masquerade-HOWTO

Red Hat Documentation Team. (2000). Red Hat Linux 6.2: The official Red Hat Linux reference guide. Red Hat, Inc. 2600 Meridian Parkway, Durham, NC, 27709. http://www.redhat.com/support/manuals/RHL-6.2-Manual/ref-guide/

Retallack, R. (2001, June 15). Securing Linux installations. SANS Institute Information Security Reading Room. Retrieved August 22, 2001 from http://www.sans.org/ infosecFAQ/linux/sec_install.htm

Rivest, R. (1992, April). The MD5 message-digest algorithm. MIT Laboratory for Computer Science, and RSA Data Security, Inc.

Russell, R. (2000, July 4). Linux ipchains HOWTO. Linux Documentation Project. Retrieved August 14, 2000 from http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html

Russell, R., & Quinlan, D. (Eds.). (2001). Filesystem hierarchy standard. Filesystem Hierarchy Standard Group. Retrieved May 28, 2001 from the Filesystem Hierarchy Standard Group Web site: http://www.pathname.com/fhs/pub/fhs-2.2.pdf

Sampo, K. (2001, May 16). Net::SSLeay.pm installation. Net::SSLeay.pm Home Page. Retrieved May 26, 2001 from http://www.bacus.pt/Net_SSLeay/

Sax, D. (2000, November 12). DNS spoofing (malicious cache poisoning). Institute Information Security Reading Room. Retrieved August 22, 2001 from http://www.sans.org/infosecFAQ/firewall/DNS_spoof.htm

Schenk, T. (n.d.). Linux: Its history and current distributions. PartnerWorld for Development Library. IBM Web site. Retrieved June 23, 2001 from http://www.developer.ibm.com/library/articles/schenk1.html.

Seifried, K. (2001, June). Linux administrator's security guide. Security Portal. Retrieved July 14, 2001 from http://www.securityportal.com/lasg/

Semeria, C. (n.d.). Securing your network against source IP spoofing attacks. Retrieved August 21, 2001 from http://www.indy.net/~sabronet/secure/895inet.html

Sigle, R. (2001, February 6). Building a secure RedHat Apache server HOWTO. Linux Documentation Project. Retrieved September 3, 2001 from http://www.linuxdoc.org/ HOWTO/SSL-RedHat-HOWTO.html

Skoric, M. (2001, May 13). LILO mini-HOWTO. Linux Documentation Project. Retrieved June 16, 2001 from http://www.linuxdoc.org/HOWTO/mini/LILO.html

Spafford, E. H. (1991, June). OPUS: preventing weak password choices. Purdue Technical Report CSD-TR 92-028

Spitzner, L. (2000, September 19). Armoring Linux. Personal Web site. Retrieved June 8, 2001 from http://www.enteract.com/~lspitz/linux.html

Spitzner, L. (2001, January). What is MD5 and why do I care? Personal Web site. Retrieved January 15, 2001 from http://www.enteract.com/~lspitz/md5.html

Stocksdale, G. (1998, April). NSA glossary of terms used in security and intrusion detection. SANS Institute Resources Retrieved August, 23, 2001 from SANS Institution Resources: http://www.sans.org/newlook/resources/glossary.htm

Tevesk, S. (2001, February 24). OpenSSH security. OpenBSD: security Web site. Retrieved August 24, 2001 from http://www.openssh.com/security.html

Thorpe, J. (2001). BYO Linux 1.0 text version. Retrieved May 30, 2001 from BYOLinux Web site: http://www.byolinux.org/

Toomey, W. (2001, February 28). Users and user administration. School of Computer Science, University of New South Wales, Australian Defence Force Academy. Retrieved June 20, 2001 from http://www.cs.adfa.oz.au/teaching/studinfo/sa3/ Lectures/user_admin.html

Veselosky, V. (1999). Configuring LILO, the Linux loader. Retrieved June 16, 2001 from Control-Escape: Alternative Software Web site: http://www.control-escape.com/lilo-cfg.html

Vuksan, V. (2000, July 3). DHCP mini-HOWTO. Linux Documentation Project. Retrieved August 16, 2000 from http://www.linuxdoc.org/HOWTO/mini/DHCP/index.html

Ward, B. (2001, July 15). version 3.0 . The Linux kernel HOWTO. Linux Documentation Project. Retrieved July 29, 2001 from http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html

Welte, H. (2001, October 14). v1.2. How to set up a chroot environment with Red Hat Linux 6.2. Personal Web site. Retrieved June 16, 2001 from http://www.gnumonks.org/ ftp/pub/doc/chroot-howto.html

Wreski, D. (1998, August 22). Linux security administrator's guide. Network Information Center. Retrieved July 14, 2001 from http://www.nic.com/~dave/SecurityAdminGuide/ SecurityAdminGuide-all.html

Writing buffer overflow exploits--a tutorial for beginners (n.d.). Retrieved July 2, 2001 from Mixter Security: http://members.tripod.com/mixtersecurity/exploit.txt

Wunsch, S. (2001, February 24). v1.3. Chroot-BIND HOWTO. Linux Documentation Project. Retrieved March 28, 2001 from http://www.linuxdoc.org/HOWTO/Chroot-BIND-HOWTO.html

Ylonen, T. (2001, August 1). v 1.14 OpenBSD reference manual ssh: OpenSSH ssh client. OpenSSH Manual Pages. http://www.openssh.com/manual.html Retrieved August 24, 2001 from http://www.openbsd.org/cgi-bin/man.cgi?query=ssh

Ylonen, T. (2001, August 1). v 3.0 OpenBSD system manager's manual SSHD: OpenSSH ssh client. OpenSSH Manual Pages. http://www.openssh.com/manual.html Retrieved August 24, 2001 from http://www.openbsd.org/cgi-bin/man.cgi?query=sshd

Ziegler, R. L. (1999, November 3). Linux firewalls. Indianapolis, Indiana. New Riders Publishing.

BIOGRAPHICAL SKETCH

William Valella was born June 28, 1971, in Belle Glade, Florida. As a native Floridian, he received a bachelor's degree in English from the University of Florida in June 1993 and a master's degree in English education through the University of Florida's PROTEACH program in June 1994. He began teaching English at the middle school level in a small southwestern Florida city in August 1994. He then moved to the high school level and taught TV productions, debate, speech, drama, and produced the the daily news program.

In 1998, he decided to return to the University of Florida to pursue a master's degree in computer science. After a year fulfilling the prerequisites, he was admitted to the graduate program in computer science in 1999. During the summer of 2000, he traveled to San Jose, California, for a summer internship with a leading ATE company. At the end of the summer he and his wife decided to relocate to San Jose. He completed the remainder of his graduate requirements through the University of Florida's FEEDs program under the direction of Dr. Manuel Bermudez.