OPERATING SYSTEM`S SECURITY: LINUX

A Thesis
Submitted to the Department of Computer Science and Engineering
of
BRAC University
by

Mehedi Al Mamun
ID: 01101072

In Partial Fulfillment of the
Requirements for the Degree

Of

Bachelor of Science in Computer Science

August 14th 2005

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, previously submitted for any degree.

Signature of Supervisor

Signature of Author

ACKNOWLEDGMENTS

I want to give special thanks to Dr. Mumit Khan. Without his help, It would be impossible for me to complete my thesis. I would also like to thanks to all of my friends specially Md.Zahurul Islam , Zahiduzzaman for supporting me to do this work successfully. Finally I am grateful to S.M Mahbubuzaman, Assistant system Administrator of BRAC University for helping me with necessary equipments.

ABSTRACT

I propose a comprehensive investigation of the security issues in the Linux Operating System. Linux is an open source operating system and frequently used by both individual users and corporate users. The goal of this project is to conduct a thorough understanding of how Linux provides the standard security model known as CIA, or Confidentiality, Integrity, and Availability.

TABLE OF CONTENT

LIST OF FIGURES

Figure PAGE

1.1 Introduction to Information Security

Information security covers a wide area of computing and information processing. Right now every large organization, university's, industries that depends on computer systems and networks to conduct daily transactions and access crucial information regard their data as an important part of their assets. A number of people are using their computers to gain access to the resources that the Internet has to offer. Peoples all around the world are frequently uses credit card, online banking. There is nothing difference between when a thief steal money from houses and a hacker knows our bank account number with password. That is one example why information security is needed but there are many other aspects with Information security .

Computer Security: "A computer is secure if you can depend on it and its software to behave as you expect"[6]. Through out this thesis paper I discusses as a software of operating system how Linux provides information security.

1.2 Security Model CIA

The major technical areas of computer security are usually represented by the initials CIA, confidentiality, integrity, and authentication or availability. Confidentiality means that information cannot be access by unauthorized parties. Confidentiality is also known as secrecy or privacy. Sensitive information must be available only to a set of pre-defined individuals. Unauthorized transmission and usage of information should be restricted.

Integrity means that information is protected against unauthorized changes that are not detectable to authorized users. Unauthorized users should be restricted from the ability to modify or destroy sensitive information .

Authentication means that users are who they claim to be. Availability means that resources are accessible by authorized parties any time that it is needed. Availability is a warranty that information can be obtained with an agreed-upon frequency and timelines[9,11].

2.1 Brief History Of Linux

Linux is quite possibly the most important free software . It has developed into an operating system for business, education, and personal productivity. What makes Linux so different is that it is a free implementation of UNIX. It was and still is developed cooperatively by a group of volunteers, primarily on the Internet, who exchange code, report bugs, and fix problems in an open-ended environment.

Linux is developed primarily by Linus Torvalds at the University of Helsinki in Finland, with the help of many UNIX programmers and wizards across the Internet. Linux was developed by the GNU project of the Free Software Foundation in Cambridge, Massachusetts, U.S.A. However, programmers from all over the world have contributed to the growing pool of Linux software.

Linux was originally developed as a hobby project by Linus Torvalds. It was inspired by Minix, a small UNIX system developed by Andy Tanenbaum. The very early development of Linux mostly dealt with the task-switching features of the 80386 protected-mode interface, all written in assembly code. On October 5, 1991, Linus announced the first ``official'' version of Linux, which was version 0.02.

2.2 Linux Architecture

Linux kernel is monolithic. It is a large, complex do-it-yourself program, composed of several logically different components(or subsystems). The architectural style of the Linux kernel is close to Data Abstraction style at the highest level. The kernel is composed of subsystems that maintain internal representation consistency by using a specific procedural interface. On the other hand, layered styles for Linux as a whole and within the subsystems of the Linux kernel[16].

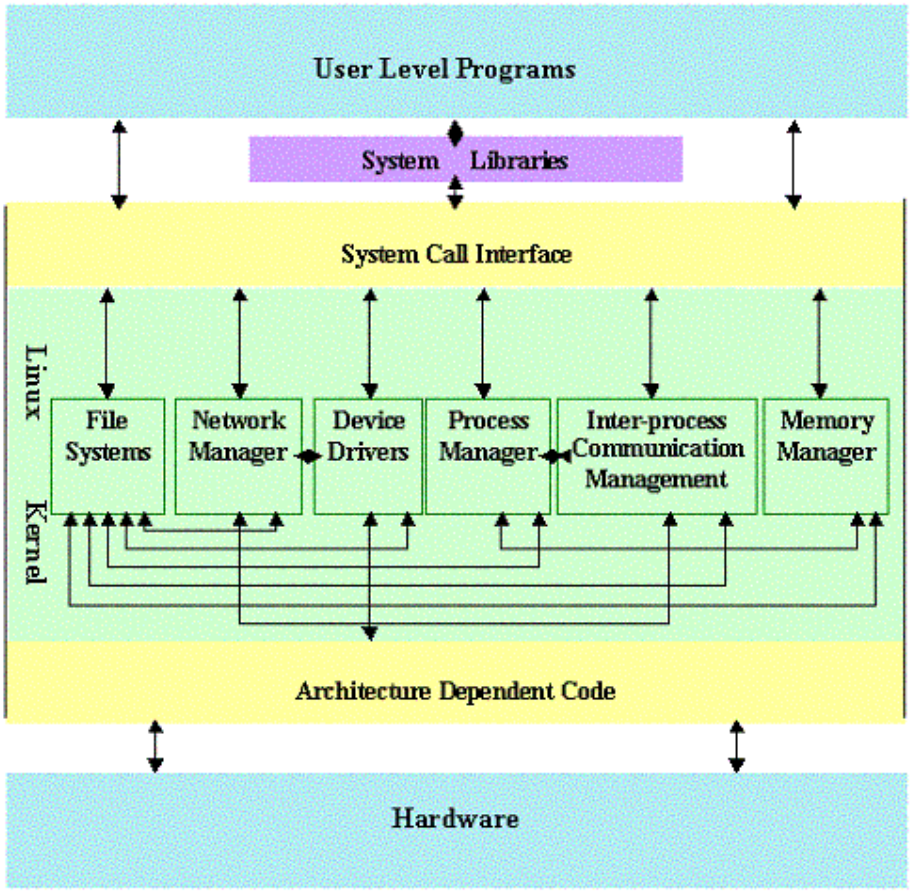


Figure 2.1 Linux Kernel Architecture

In Linux architecture, each layer provides a service to the layer above it and serves to the client below. Benefits of this style of architecture is that this design supports increasing levels of abstraction and enhancement as changes to the functionality of one layer affects at most two others. Linux kernel includes the middle layer kernel modules and two interface layers system call interface and architecture-dependent code, which provide interface between the user applications, kernel modules, and hardware. For instance, the system call interface layer provides an interface between the virtual file system and the user level programs that need to access a file system. Similarly, the architecture-dependent interface provides an interface between the virtual file system and the disk that it must access. The kernel modules layer conceptually composed of six major subsystems: the process

scheduler, the memory management, the virtual file systems, the network management, inter-process communication management and the device drivers. These subsystems interact with each other using procedure calls and shared data structures[16].

2.2.1 Network Services Module

The network subsystem allows Linux systems to connect to other systems over a network. There are a number of possible hardware devices that are supported, and a number of network protocols that can be used. The network subsystem abstracts both of these implementation details so that user processes and other kernel subsystems can access the network without necessarily knowing what physical devices or protocol is being used[16].

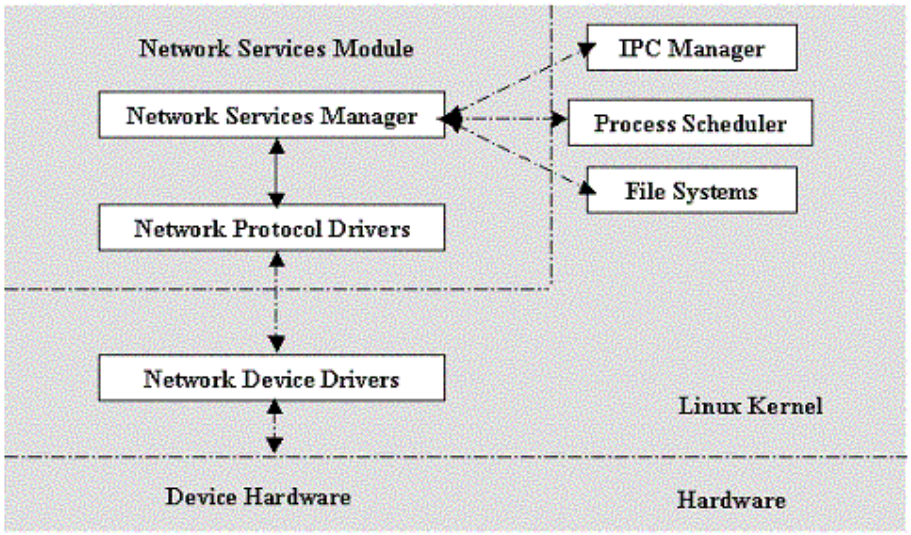


Figure 2.2 Linux kernel's Network service module

The network services module follows a very simple layered architecture style. It provides access to several networking standards and a variety of network hardware. The network services manager interfaces with the TCP/IP protocol drivers which in turn interface with the necessary device drivers required to make use of the attached networking hardware[16].

The network resource manager also communicates with the IPC manager in order to provide support for IPC through sockets. The network

subsystem uses the process scheduler to suspend and resume processes while waiting for hardware requests to complete. In addition, the network subsystem supplies the virtual file system with the implementation of a logical file system leading to the virtual file system depending on the network interface and having data and control flow with it. Thus provides a common interface for user applications[16].

2.2.2 Device Drivers Modules

The device driver layer is responsible for presenting a common interface to all physical devices such as graphics cards, network cards, hard disks etc. The Linux kernel has three types of device driver: character, block, and network. The two types relevant to the file subsystem are character and block devices. Character devices must be accessed sequentially; typical examples are tape drives, modems, and mice. Block devices can be accessed in any order, but can only be read and written to in multiples of the block size[16].

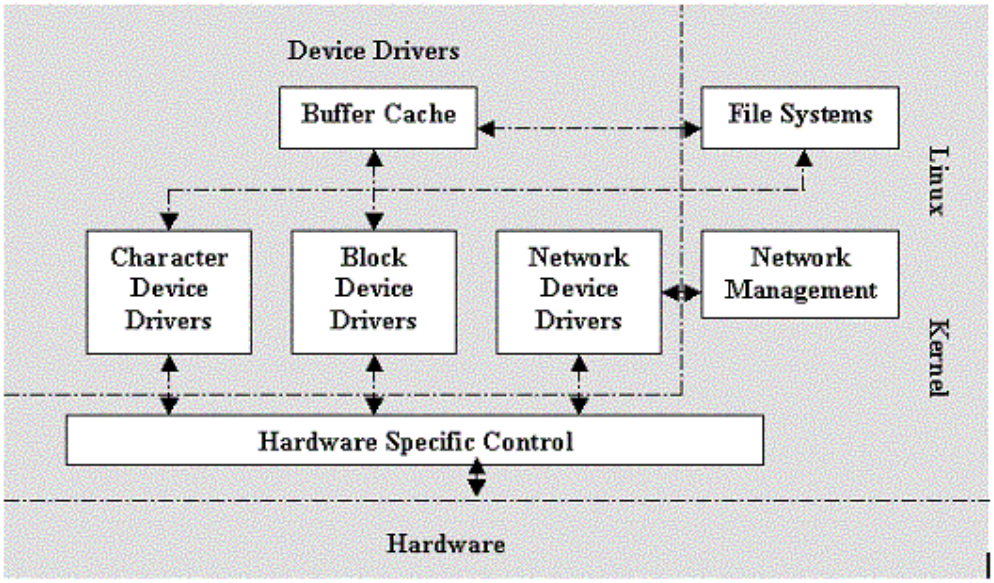


Figure 2.3 Linux kernel's Device drivers module

Each device can be accessed as though it was a file in the file system (this file is referred to as a device special file). Since most of the kernel deals with devices through this file interface, it is relatively easy to add a new device driver by implementing the hardware-specific code to support this abstract file

interface. The Linux kernel uses a buffer cache to improve performance when accessing block devices. All access to block devices occurs through a buffer cache subsystem. The buffer cache greatly increases system performance by minimizing reads and writes to hardware devices[16].

2.3 Differences Between Linux and MS-DOS

It is important to understand the differences between Linux and other operating systems, like MS-DOS, OS/2, and the other implementations of UNIX for personal computers. First of all, Linux coexists happily with other operating systems on the same machine.It can run MS-DOS and OS/2 along with Linux on the same system without problems.

MS-DOS does not fully utilize the functionality of 80386 and 80486 processors. On the other hand, Linux runs completely in the processor's protected mode, and utilizes all of its features. Linux can directly access all the available memory and beyond, with virtual RAM. Linux provides a complete UNIX interface which is not available under MS-DOS. We can easily develop and port UNIX applications to Linux, but under MS-DOS it is limited to a subset of UNIX functionality. MS-DOS is inexpensive compared to other commercial operating systems and has a strong foothold in the personal computer world. Linux however, is free.

3.1 BIOS Security

An Operating System is needed security from its booting up to shutting down. If an attacker has access to the BIOS, nothing will be remain safe. Whenever an operating system is boot up it is very important whether an unauthorized user has full physical access to computer. The BIOS is the lowest level of software that configures or manipulates of x86 based hardware. LILO and other Linux boot methods access the BIOS to determine how to boot up Linux machine.

We can set BIOS password to prevent from unauthorized physical access. To prevent changes to BIOS Settings BIOS password is needed .Some examples are, disallow booting from floppy drives and passwords to access some BIOS features. Many x86 BIOS's also allow to specify various other good security settings.

3.2 Boot Loader Security

At this moment we prevents an illegal user to change the BIOS settings by setting BIOS password. But In Linux Operating System there is an option to change the root password during booting. The two common boot loaders for Linux are LILO (Linux Loader) and GRUB (Grand Unified Boot loader).

3.2.1 Linux booting process

The boot process of Linux on a Intel i386 architecture has following steps:

- A Linux Loader is placed at the first sector read by the bios.
- The kernel is loaded.
- Init is started and executes various scripts.

3.2.2 Securing boot process

3.2.2.1 Reasons for securing boot loader

The following are the primary reasons for protecting a LINUX boot loader.

1. Prevent Access to Single User Mode

If an attacker can boot into single user mode , he becomes the root user and a root user can change anything to his own way which is directly threat to a secure operating system.

2. Prevent Access to the GRUB Console

If an illegal user get access to GRUB editor, he can change its configuration or he can get information by using cat command.

3. Prevent Access to Non-Secure Operating Systems-

If the system is using dual booting, any body can select an operating system which is less secure than Linux which ignores access controls and file permissions[11].

3.2.2.2 LILO security

LILO is the Linux boot loader, it handles all the tasks of getting the kernel into memory and bootstrapping them machine into something that resembles a useful computing device. To secure LILO we have to set a password, and use the restricted keyword. We can also set security on a per image basis, or adding a password directive in to the global section of its confiuration file. Add a passwords directive to /etc/lilo.conf file.

password= <my password >

If we want to allow booting a kernel without password verification, but do not want to allow users to add arguments without password add restricted keyword below the password line[11].

```
image=/boot/vmlinuz-<version>
password=<thisisapassword>
restricted
```

3.2.2.3 Grub security

GRUB adds an extra level of security by supporting MD5 encryption for the password in the configuration file. To generate an encrypted password, run the command

$ /sbin/grub-md5-crypt

When prompted for password type a password and it will return an MD5 hash of the password. Next edit the /boot/grub/grub.conf and set the MD5 hash value of password[11].

password –md5<password-hash>

3.3 Users and Groups

User accounts are important to verify the identity of the person using a computer system. By verifying the accounts of user the system is able to determine if the user is permitted to log into the system and, if so, which resources the user is allowed to access.

There are three types of users:

i) Root
ii) Normal users
iii) System users

The superuser, normally named root ,has complete control over the entire system. The root user can access all files on the system and the root user is generally the only user who can execute certain programs .The root has a user ID 0.Any account with a user ID of 0 is a root user, even if the username is not root.

Normal users are users who can log in. Normal users usually have a home directory and can create and manipulate files in their home directory and in other directory. Normal users typically have restricted access to files and directories on the machine and as a result they cannot perform many system-level functions.

System users don't login. They are accounts that are used for specific purposes and are not allowed by a specific person. For example user nobody and lp. The user nobody is the user who typically handle the http request and lp handles print request.

Groups are logical constructs that can be used to cluster user accounts together for a specific purpose. Careful group creation and assignment of privileges, access to restricted resources can be maintained for those who need them and denied to others. The user who creates a file is assigned as the owner and group owner. The file is also assigned separate read, write,

and execute permissions for the owner, the group, and everyone else. The owner of a file can be changed only by the root user. The group to which a file belongs can be changed by root or by the owner of the file if the owner is part of the group being added to the file.

For security purposes we should be aware at least the following:

- Login activity
- Authorization information
- Authentication information
- Commands users have run
- Restarts and shutdowns of the system
- Network transactions records

3.3.1 Managing user accounts

Linux provides a large number of tools including account permissions, passwords, account aging, adding and deleting of users, etc. Following commands are useful to manage user and group.

chage - change user password expiry information

- groups - print the groups a user is in
- newusers - update and create new users in batch
- passwd - update a user's authentication tokens(s)
- nologin - prevent non-root users from log into the system
- su - run a shell with substitute user and group IDs
- useradd - Create a new user or update default new user information
- userdel - Delete a user account and related files

- usermod - Modify a user account
- chgrp - change the group ownership of files
- chown - change the user and group ownership of files
- gpasswd - administer the /etc/group file
- groupadd - Create a new group
- groupdel - Delete a group
- groupmod - Modify a group
- groups - print the groups a user is in
- grpck - verify integrity of group files
- pwconv - convert to and from shadow passwords
- pwunconv - convert to and from shadow passwords
- grpconv - convert to and from shadow passwords
- grpunconv- convert to and from shadow passwords
- vipw - edit the password or group files
- vigr - edit the password or group files

3.3.2 Passwords

Password is one of the most important security features and basic means of authentication. It is important to set secure, unguessable passwords. Password security is the most critical means to protect system from compromise. An effective well-chosen password is always desirable not be compromised the system.

Linux have several characteristics of password storing mechanism.

i) In a file that is readable only by root.

ii) In a one way hash format.

Following guideline should be maintain for strong password

- Use of shadow passwords.
- Do not use only words or numbers.
- Do not use recognizable words.
- Do not use hacker terminology.
- Do not use personal Information.
- Do not invert recognizable words.
- Make the password at least eight characters long.
- Mix upper and lower case letters.
- Mix letters and numbers.
- Include non-alphanumeric characters.

3.3.3 Guest accounts privilege

Guest accounts on servers don't have to be set up with the same privilege as those of regular users. If these guests only need to run a few programs or access a collection of local files, then enable them to do just this and nothing more.

3.4 File Systems Security

File systems security is important to keep a system safe. By changing important file like server configuration, network configuration and system configuration a machine can be compromised. A file system is the methods and data structures that an operating system uses to keep track of files on a disk or partition that is, the way the files are organized on the disk. The central concepts of Linux file systems are super block, inode, data block, directory block, and indirection block. The super block contains information about the file system as a whole, such as its size. An inode contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file.

There is space only for a few data block numbers in the inode, however, and if more are needed, more space for pointers to the data blocks is allocated dynamically. These dynamically allocated blocks are indirect blocks; the name indicates that in order to find the data block, one has to find its number in the indirect block first.

Linux chooses to have a single hierarchal directory structure. Everything starts from the root directory, represented by /, and then expands into sub-directories instead of having so-called 'drives'. On the other hand, Linux sorts directories descending from the root directory according to their importance to the boot process. Another reason for this unified file system is that Linux caches a lot of disk accesses using system memory while it is proper commands. This will shut down the system in a decent way which will thus, guarantee the integrity of files.

/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

For each object in the file system, Linux stores administrative information in a structure known as an inode. Instead, they have indices (numbers) indicating their positions in the array of inodes.

Each inode generally contains:

- The location of the item's contents on the disk, if any
- The item's type (e.g., file, directory, symbolic link)

- The item's size, in bytes, if applicable
- The time the file's inode was last modified
- The time the file's contents were last modified
- The time the file was last accessed (the atime) for read ( ), exec ( ), etc
- A reference count: the number of names the file has
- The file's owner (a UID)
- The file's group (a GID)
- The file's mode bits (also called file permissions or permissionbits)

Linux separates access control on files and directories according to three characteristics: owner, group, and other. There is always exactly one owner, any number of members of the group, and everyone else. Any user will be able to view contents of a file and edit by setting read ,write permission of a file.

Following table is the list file protection command with chmod [6].

Table– 3.1 Linux File protection command

Command	Meaning
chmod 400 file	To protect a file against accidental overwriting.
chmod 500 directory	To protect from accidentally removing, renaming or moving files from this directory.
chmod 600 file	A private file only changeable by the user who entered this command.
chmod 644 file	A publicly readable file that can only be changed by the issuing user.
chmod 660 file	Users belonging to group can change this files, others don't have any access to it at all.
chmod 700 file	Protects a file against any access from other users, while the issuing user still has full access.
chmod 755 directory	For files that should be readable and executable by others, but only changeable by the issuing user.
chmod 775 file	Standard file sharing mode for a group.
chmod 777 file	Everybody can do everything to this file.

4.1 Network File System

NFS is the standard way Linux machines can share files over the network. A client can mount directories of a server and thereafter the files are accessible just as if they were local disk storage. It works for a large organization, University or other institutions where information need to keep a common palace so that everybody can read, write as privileged. Administrative data can kept in one host and easy to maintain. The virtual file system (VFS) interface is the mechanism used by NFS to transparently and automatically redirect all access to NFS-mounted files to the remote server. NFS clients use the remote procedure call (RPC) suite of network application helper programs to mount remote file systems. If the mount cannot occur during the default RPC timeout period, then the client retries the mount process until the NFS number of retires has been exceeded.

To configure NFS server we have to edit /etc/exports file. A sample exports file is shown below-

```
#/etc/exports
/data/files *(ro,sync)
/home    192.168.1.0/24(rw,sync)
/data/test *.my-site.com(rw,sync)
```

4.1.1 Important daemon of NFS

NFS isn't a single program, but a suite of interrelated programs that work together to get the job done.

Portmap: is a server that converts RPC program numbers into DARPA protocol port numbers. It must be running in order to make RPC calls. When an RPC server is started, it will tell portmap what port number it is listening to, and what RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it will first contact portmap on the server machine to determine the port number where RPC packets should be sent. Portmap must be started before any RPC servers are invoked. Normally portmap forks and dissociates itself from the terminal like any other daemon. By default, portmap listens to TCP port 111 on which an initial connection is made. This is then used to negotiate a range of TCP

ports, usually above port 1024, to be used for subsequent data transfers. We need to run portmap on both the NFS server and client[12].

nfs: It Starts the RPC processes needed to serve shared NFS file systems. The nfs daemon needs to be run on the NFS server only[12].

nfslock : It Used to allow NFS clients to lock files on the server via RPC processes. The nfslock daemon needs to be run on both the NFS server and client[12].

4.1.2 Securing NFS

Before implementing an NFS server first we have to secure the PORTMAP services. The PORTMAP service is a dynamic port assignment daemon for RPC services. It has weak authentication mechanisms and has the ability to assign a wide range of ports for the services it controls. For these reasons, it is difficult to secure.

Linux provides a number of way to secure the PORTMAP, for this we have to do following things

- Protect portmap With TCP Wrappers.
- Protect portmap With iptables.

Remote root user of client can act like a local root user. To prevent this never use no_root_squash option so that the power of the remote root user become to the lowest local user[11].

Specify the client list in the /etc/exports file which are allowed to export file from server.

4.2 Network Information System(NIS)

NIS stands for Network Information Service. It is an RPC service called ypserv which is used in conjunction with portmap and other related services to distribute maps of usernames, passwords, and other sensitive information to any computer within its domain.

4.2.1 Important daemon of NIS

An NIS server is comprised of several applications. They include the following:

ypserv: The ypserv daemon is typically activated at system startup. ypserv runs only on NIS server machines with a complete NIS database. On other machines using the NIS services, have to run ypbind as client. ypbind must run on every machine which has NIS client processes; ypserv may or may not be running on the same node, but must be running somewhere on the net- work. On startup or when receiving the signal SIGHUP, ypserv parses the /etc/ypserv.conf file[12].

Ypbind: It is the main daemon at client side and finds the server for NIS domains and maintains the NIS binding information. The client could get the information over RPC from ypbind. The binding files resides in the directory /var/yp/bind. After a binding has been established, ypbind will send YPPROC_DOMAIN requests to the current NIS server at 20 seconds intervals. If it doesn't get an response or the NIS server tells that he doesn't has this domain any longer, ypbind will search a new NIS server. All 15 minutes ypbind will check, if the current NIS server is the fastest. At startup or when receiving signal SIGHUP, ypbind parses the file /etc/yp.conf and tries to use the entries for its initial binding[12].

yppasswdd: It Also called the yppasswdd service, this daemon allows users to change their NIS passwords[12].

ypxfrd: It Also called the ypxfrd service, this daemon is responsible for NIS map transfers over the network[12].

4.2.2 Securing NIS

1. Use LDAP(Light weight Directory Access Protocol)

NIS is rather insecure by today's standards. It has no host authentication mechanisms and passes all of its information in clear text, including password hashes. As a result, extreme care must be taken to set up a network that uses NIS. So instead of NIS use LDAP which overcomes the problems of NIS.

2. Use a Password-Like NIS Domain Name and Hostname

Any machine within an NIS domain can use commands to extract information from the server without authentication, as long as the user knows the NIS server's DNS hostname and NIS domain name. To make access to

NIS maps harder for an attacker, create a random string for the DNS hostname. Similarly, create a different randomized NIS domain name. This will make it much more difficult for an attacker to access the NIS server[11].

3. Assign Static Ports and Use iptables Rules

All of the servers related to NIS can be assigned specific ports except for rpc.yppasswdd the daemon that allows users to change their login passwords. Assigning ports to the other two NIS server daemons, rpc.ypxfrd and ypserv, allows to create firewall rules to further protect the NIS server daemons from intruders[11].

4.3 Domain Name Systems(DNS)

4.3.1 How DNS works

Domain Name System (DNS) is the invention of a hierarchical domain-based naming scheme and a distributed database system for implementing this naming scheme. It is primarily used for mapping hostnames and e-mail destinations to ip addresses but can be used for other purposes. Linux uses Berkley Internet Name Domain (BIND) for name resolution.

DNS is distributed database is indexed by domain names. Each domain name is essentially just a path in a large inverted tree, called the domain name space. The tree's hierarchical structure shown in figure 4.1. The tree has a single root at the top. DNS simply calls it "the root"[5].

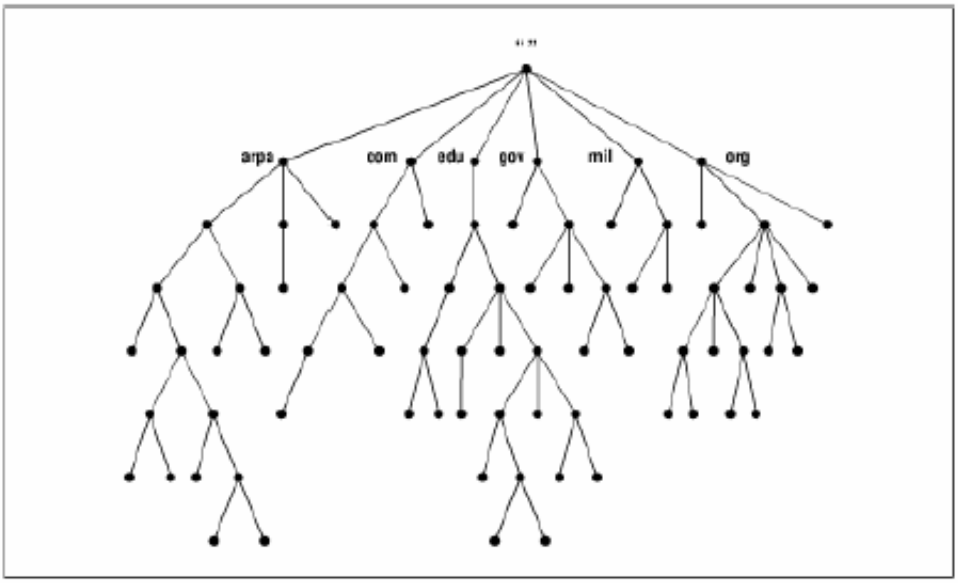


Figure- 4.1 The Structure of DNS name space

A domain is simply a subtree of the domain name space. The domain name of a domain is the same as the domain name of the node at the very top of the domain. Every domain whether it is a single host or a top-level domain, can have a set of resource records associated with it. For a single host, the most common resource record is just its IP address, but many other kinds of resource records also exist. When a resolver gives a domain name to DNS, what it get back are the resource records associated with that name. Thus the primary function of DNS is to map domain names onto resource records[5].

To avoid the problems associated with having only a single source of information, the DNS name space is divided into non overlapping zones. Each zone contains some part of the tree and also contains name servers holding the information about that zone. Normally a zone will have one primary name server, which gets its information from a file on its disk and one or more secondary name servers, which get their information from the primary name server. A sample zone files for my-site.com is shown follow[13].

```
zone” my-site.com” IN{
        type master;
        file ”my-site.com.db”;
        allow-query{any};
};
```

4.3.2 Securing DNS

DNS can reveal a lot about the nature of the domain. For this we should take some precautions to hide some of the information for the shake of security. The host command does one DNS query at a time but the dig command is much more powerful. When given the right parameters it can download the entire contents of domain's zone file. This may not seem like an important security threat at first glance, but it is. Anyone can use this command to determine all server's ip address and from the names determine what type of server it is and then launch an appropriate cyber attack. Without master and slave servers zone transfer should be disabled. We can do this by applying the allow-transfer directive to the global options section of named.conf file[8].

```
options{
allow-transfer{none;}; }
```

In order to prevent unauthorized access to the named daemon, BIND uses a shared secret key method which is used to grant privileges to hosts. An identical key must be present in both /etc/named.conf and rndc configuration file, /etc/rndc.conf[13].

BIND support advanced features DNSSEC, which stands for DNS Security Extensions, is a method by which DNS servers can verify that DNS data is coming from the correct place, and that the response is unmodified. It is a public/private key system. This means that the owner of a DNS zone has a private key and a public key. Using the private key to digitally sign a zone will allow anyone with the zone's public key to verify that the data is authentic[13].

4.4 HTTP Server

4.4.1 Apache HTTP server

A web server provides services through HTTP protocol. Usually the server receive a request from client for specific resource and returns the resource as a response. Linux uses Apache HTTP server as a web server.

Apache is a modular and process based server. This implies that only the most basic functionality is included in the core server and the server forks itself a number of times to answer simultaneous requests. The children are isolated from each other. This is reliable if a module misbehaves, the parent process kills that child and it only affects the request being served, not the server as a whole[15].

4.4.2 Apache HTTP server's security

The number of individuals and companies with internet access is expanding rapidly. As a result businesses are interested about setting up facilities on the web for electronic commerce. But the reality is that the Internet and the Web are extremely vulnerable to compromises of various sorts. As businesses wake up to this reality the demand for secure Web services grows. Using regular HTTP communications between a browser and a web server are sent a plaintext, which could be intercepted and read by someone along the route between the browser and the server.

Apache HTTP server has strong security policy which provides by the mod_ssl and openssl packages. It supports Transport Layer Security and Secure Socket Layer. The Secure Sockets Layer/Transport Layer Security protocols allow data between the Web server and client to be encrypted[14].

SSL provides following three basic security functions.

1. Authentication: SSL supports server-only, client/server, and anonymous authentication.

2. Confidentiality: SSL uses public-key cryptography for secure key exchange, and symmetric-key encryption for bulk-cipher.

3. Message integrity: SSL uses cryptographic hash-based message authentication codes (MAC).

Secure Sockets Layer (SSL) is transport layer approach to Web security; SSL uses TCP/IP to provide a reliable, general purpose security service to upper-layer protocols. SSL is not a single protocol but rater two layers of protocol as illustrated in following figure.

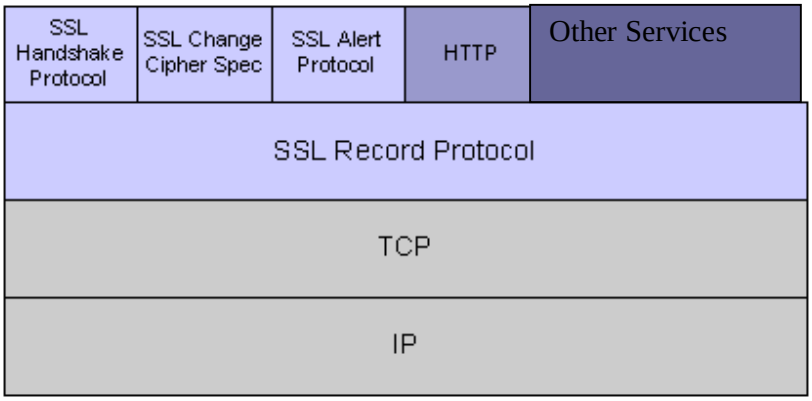


Figure 4.2 SSL Protocol Stack

The SSL Record Protocol provides basic security services to various higher-layer protocols. Three higher-layer protocols are defined as part of SSL the handshake Protocol, the Change Cipher Spec Protocol and the Alert protocol.

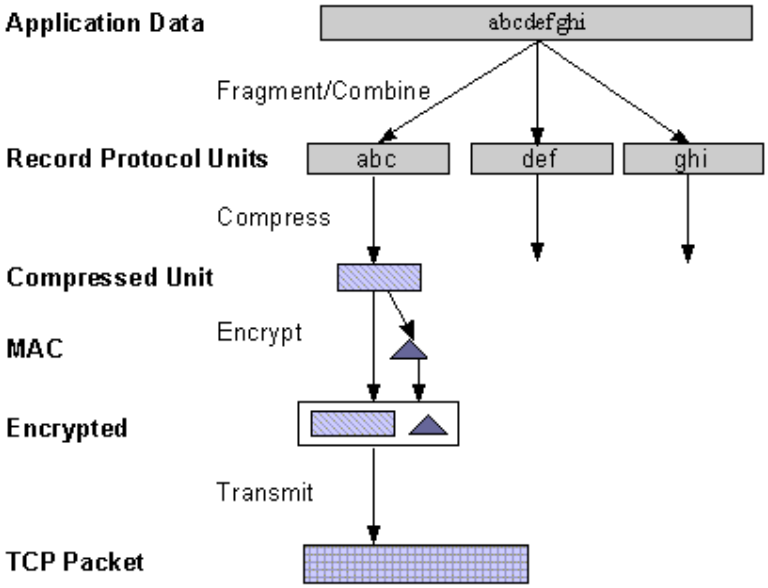


Figure 4.3 SSL Record Protocol Operation

The SSL Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks,opt6ionally compresses the data, applies a MAC, encrypts, adds a header and transmits the resulting in a TCP segment. Received data are decrypted, verified,

decompressed and reassembled and then delivers to higher level users. Figure 4.3 illustrates the overall operation of the SSL Record Protocols[2].

Change Cipher Spec Protocol is one of the three SSL-specific protocols that use SSL Record Protocol . It consists of a single message which consists of single byte with the value 1.The sole purpose of this message is to cause the pending state to be copied into the current state which updates the cipher suite to be used on this connection[2].

SSL Alert Protocol is used to alerts the peer entity. This protocols alerts about unexpected_message, bad_recordmac, decompression_failure, illegal_parameter[2].

Handshake Protocol is the most complex part of SSL. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of message exchanged by client and server, shown in Figure -4.4[2].

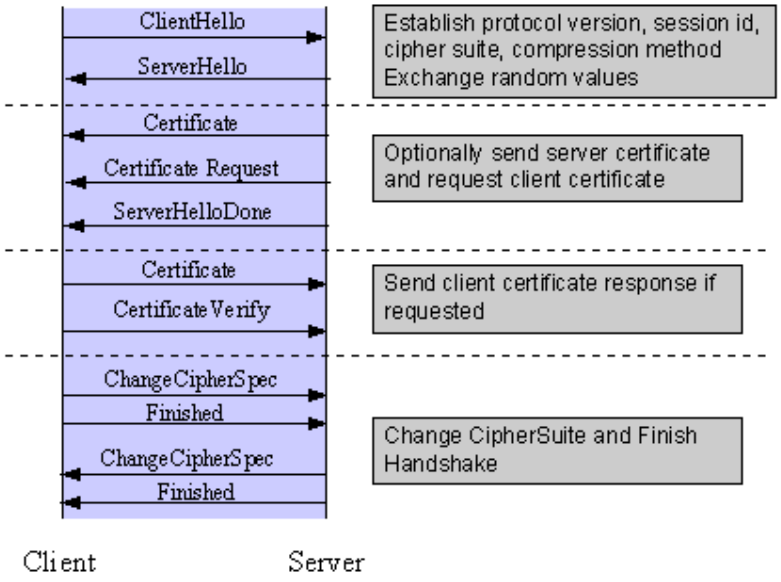


Figure 4.4 SSL Handshake Protocol

The initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.

1. Establish Security Capabilities : This phase is used to initiate a logical connection and to establish the security capabilities. The exchanged is initiated by the client which sends a client_hello message with Version, Random, Session ID, CipherSuite, Compression Method parameters. Client sends a list of cryptographic algorithm in decreasing order of preference. After sending the client_hello message the client waits for the server_helo message, which contains the same parameters as the client_hello message. In the server_hello message Version field contains the lower version suggested by the client and highest supported by the server. The Random field is generated by the server and is independent of client's Random field. The CipherSuite field contains the single cipher suite selected by the server from those proposed by the client[2].

2.Server Authentication and Key Exchange : The server begins this phase by sending its certificate, if it needs to be authenticated.The message contains one or a chain of X.509 certificates. A server_key_exchange message may be sent if it is required.The certificate_request message includes two parameters, certificate_type and certificate_authorities.The final message server_done message which is sent by server to indicate the end of the server hello and associated messages. After sending this message, the server will wait for a client response[2].

3. Client Authentication and Key Exchange : After receipt of the server_done message the client begins this phase by sending a certificate message. Next client_key_exchange message is sent. Finally send certificate_verify message to provide explicit verification of a client certificate[2].

4. Finish : This phase completes the setting up of a secure connection[2].

For key exchanges SSL Protocols used following protocols[2].

- RSA key exchange when certificates are used.
- Diffie-Hellman key exchange for exchanging keys without certificates.

For data transfer following algorithm are uses[2].

- RC4 with 40-bit keys
- RC4 with 128-bit keys
- RC2 with 40 bit key
- DES with 40 bit key
- DES with 56 bit key
- Triple-DES with 168 bit key
- Idea (128 bit key)
- Fortezza (96 bit key)

4.5 Email Security

4.5.1 Sendmail

Electronic mail(email) is the most heavily used network-based application. An e-mail message, just like a letter sent through regular mail begins with a sender and ends with a receiver. In between these two people are many postal workers who ensure that the letter is properly handled. Email works similar fashion and although there are not many people between the sender and receiver programs perform the same function. These programs use network protocols to do the job of ensuring that the message goes from sender to receiver. An email applications fall into at least one of three classifications are Mail Transfer Agent (MTA), Mail Delivery Agent (MDA) and Mail User Agent(MUA).The purposes of a MTA is to transfer mail between two MTA's and MDA used for transfer mail from mail server to client inbox .MUA allows a user to read and compose email messages.

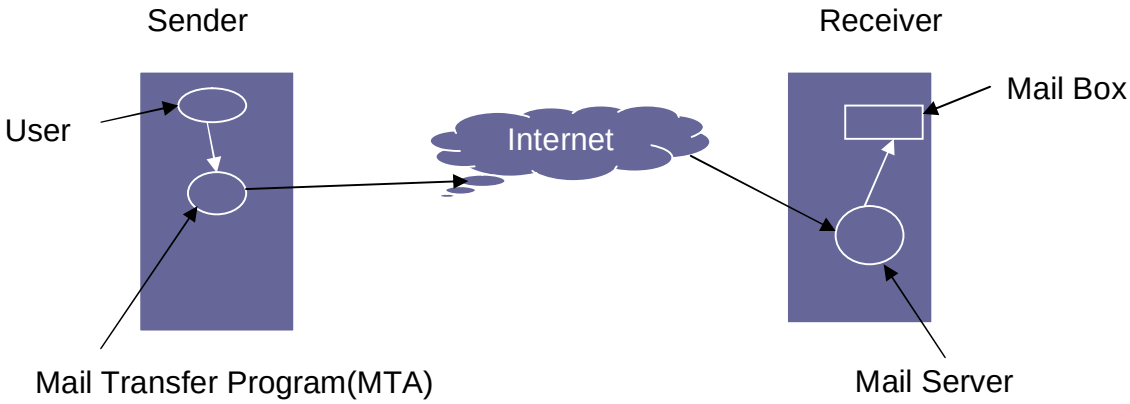


Figure- 4.5 Email Process

Linux uses three kinds of MTA's, Sendmail, Postfix and Fetchmail. In my thesis I uses Sendmail as a MTA.

Sendmail uses SMTP(Simple Mail Transfer Protocol) to transfer email. which case it will process any queued mail and then quit; or it can be run as a persistent background dæmon. If Sendamil running as a dæmon, it listens for incoming SMTP connections on TCP port 25 and periodically tries to send any outbound messages in its queue directory /var/spool/mqueue. If it's being invoked on the fly, it attempts to deliver the outbound message it's been invoked to send and/or checks /var/spool/mqueue for other pending outbound messages.

4.5.1 Securing Email

4.5.1.1 Why Email security needed

When a email message send over the internet it may passes several points. This gives chance to an attacker to read the message or even alter it. Some times message may be lost or other people send message as a duplicate person. But we never want to this vulnerability. So we need to secure our email.

4.5.1.2 Securing Email with GnuPG

We can secure email by GnuPG(GNU Privacy Guard).GnuPG is tool for secure communication and data storage. It can be used to encrypt data and to create digital signatures. It includes an advanced key management facility. It provides data integrity services for messages and data files by using these core technologies:

- digital signatures
- encryption
- compression
- radix-64 conversion

Encryption: GnuPG uses public key encryption to provide confidentiality. With public-key encryption, the object is encrypted using a symmetric encryption algorithm. Each symmetric key is used only once. A new "session key" is generated as a random number for each message. Since it is used only once, the session key is bound to the message and transmitted with it. To protect the key, it is encrypted with the receiver's public key. The sequence is as follows:

1.The sender creates a message.

2.The sender generates a random number to be used as a session key for this message only.

3.The session key is encrypted using each recipient's public key. These "encrypted session keys" start the message.

4.The sender encrypts the message using the session key, which forms the remainder of the message.

5.The receiver decrypts the session key using the recipient's private key.

6.The receiver decrypts the message using the session key. If the message was compressed, it will be decompressed.

Digital signature: The digital signature uses a hash code or message digest algorithm, and a public-key signature algorithm.

Compression: It compress the message after applying the signature but before encryption.

Radix-64 conversion: It provides the service of converting the raw 8-bit binary octet stream to a stream of printable ASCII characters, called Radix-64 encoding or ASCII Armor.

5.1 Conclusion

Linux operating systems security is extremely good. It has a number of option to provides information security. A number of tools is available for Linux to secure system. Sometimes default configuration of Linux is not secure .So we have to careful when we install it and turnoff all the unused port and unnecessary services. By using Linux it is possible to establish standard security policy CIA.

LIST OF REFERENCES

[1] Brian Hatch, James Lee and George Kurtz, “HACKING LINUX EXPOSED: LINUX SECUITY SECRETS & SOLUTION”.

[2]William Stallings, “Cryptography and Network Security”.

[3] William R.Cheswick, Steven M.Bellovin,Aviel D.Rubin,”Firewalls and Internet Security”

[4] Andrew S.Tanenbaum,”Computer Networks”.

[5] Cricket Liu and Paul Albitz, ”DNS And BIND”

[6] http://www.anotherleveldesigns.com/asp/networking/puis/ch01_01.htm

[7] http://www.faqs.org/docs/linux_intro/

[8] http://www.linuxhomenetworking.com/

[9] http://web.interhack.com/publications/whatis-security.pdf.

[10] http://www.linuxsecurity.com/docs/securityadminguide/

[11] www.redhat.com/docs/manuals/ linux/RHL-9-Manual/security-guide/

[12] linux.com.hk/PenguinWeb/manpages.jsp

[13] www.redhat.com/docs/manuals/ linux/RHL-9-Manual/ref-guide/

[14] www.redhat.com/docs/manuals/ linux/RHL-9-Manual/custom-guide/

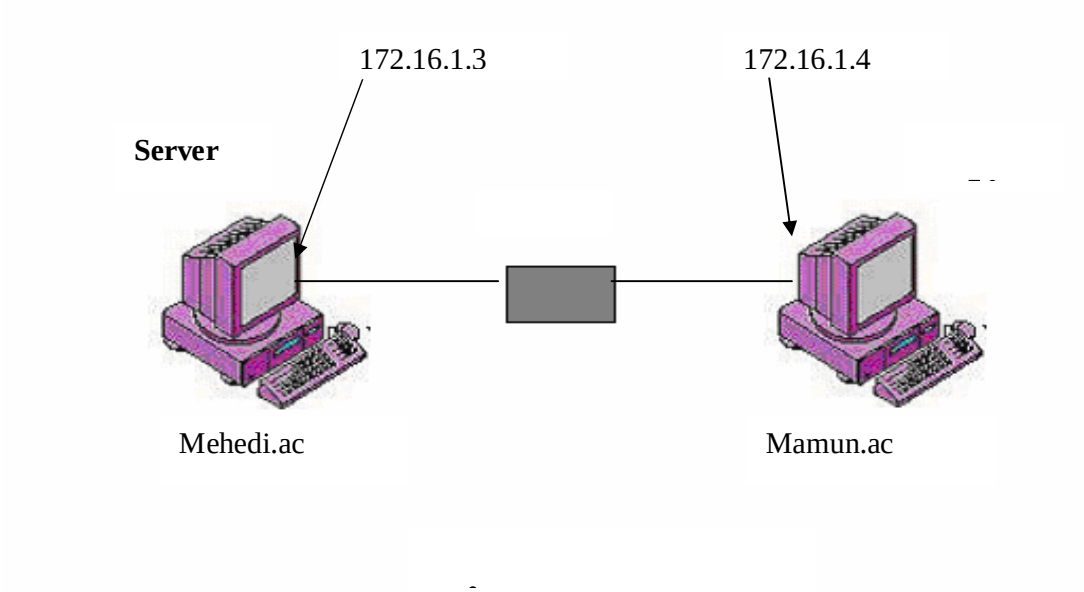
[15] http://apache.hpi.uni-potsdam.de/document/ Multitasking_architecture.html

[16] http://plg.uwaterloo.ca/~itbowman/CS746G/a1/

[17] howtos.linux.com/guides/ Linux-Filesystem-Hierarchy/foreward.shtml

[18] www.tldp.org/LDP/sag/html/sag.html

6.1 Linux Server Configuration



 # /etc/sysconfig/network-scripts/ifconfig-eth0

```
DEVICE=eth0
IPADDR=172.16.1.3
#172.16.255.255
NETMASK=255.255.0.0
BOOTPROTO=none
ONBOOT=yes
#optional
USERCTL=no
PEERDNS=no
TYPE=Ethernet

NETWORK=172.16.0.0
BROADCAST=172.16.255.255
```

#etc/sysconfig/network

```
NETWORKING=yes
HOSTNAME=mehedi.ac

NISDOMAIN=bunix

YPSERV_ARGS="-p 834"
```

```
YPXFRD_ARGS="-p 835"
YPBIND_ARGS="-p 840"
```

#/etc/hosts

```
127.0.0.1       localhost.localdomain         localhost
172.16.1.3      mehedi.ac           mehedi
```

#/etc/exports

```
/home                   *(rw,sync)
```

#/etc/named.conf

```
options {
        directory "/var/named";
        listen-on { 127.0.0.1/32; 172.16.1.0/24; };
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};

zone "." IN {
        type hint;
        file "named.ca";
};


zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};


zone "0.0.127.in-addr.arpa" IN{
        type master;
        file "named.local";
        allow-update { none; };
};



zone "my-site.com" IN {
        type master;
        allow-query { any;};
        file "db.my-site.com";
};
```

```
zone "1.16.172.in-addr.arpa" IN {
        type master;
        file "db.172.16.1";
        allow-query{any;};
};

include "/etc/rndc.key";
```

#/var/named/db.my-site.com

```
$TTL  86400
$ORIGIN my-site.com.
@       1D IN SOA    @ admin (
                                        42        ; serial (d. adams)
                                          3H              ; refresh
                                          15M             ; retry
                                          1W              ; expiry
                                          1D )            ; minimum

                           1D IN NS   @

                    IN    MX    10    my-site.com.
                    1D    IN     A    172.16.1.3
```

#/var/named/db.172.16.1

```
$TTL  86400
@       IN      SOA     my-site.com. admin.my-site.com.  (
                             1997022700 ; Serial
                             28800      ; Refresh
                             14400      ; Retry
                             3600000    ; Expire
                             86400 )    ; Minimum
           IN      NS    my-site.com.

3       IN      PTR     my-site.com.
```

#etc/httpd/conf/httpd.conf

```
ServerTokens OS
ServerRoot "/etc/httpd"
PidFile run/httpd.pid
Timeout 300
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
```

```
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
MaxClients       150
MaxRequestsPerChild  1000
</IfModule>

<IfModule worker.c>
StartServers         2
MaxClients         150
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>

<IfModule perchild.c>
NumServers           5
StartThreads         5
MinSpareThreads      5
MaxSpareThreads     10
MaxThreadsPerChild  20
MaxRequestsPerChild  0
</IfModule>

Listen 80
Include conf.d/*.conf

LoadModule access_module modules/mod_access.so
LoadModule auth_module modules/mod_auth.so
LoadModule auth_anon_module modules/mod_auth_anon.so
LoadModule auth_dbm_module modules/mod_auth_dbm.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule include_module modules/mod_include.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule env_module modules/mod_env.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule cern_meta_module modules/mod_cern_meta.so
LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
LoadModule usertrack_module modules/mod_usertrack.so
LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
LoadModule dav_module modules/mod_dav.so
LoadModule status_module modules/mod_status.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule asis_module modules/mod_asis.so
```

```
LoadModule info_module modules/mod_info.so
LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule dir_module modules/mod_dir.so
LoadModule imap_module modules/mod_imap.so
LoadModule actions_module modules/mod_actions.so
LoadModule speling_module modules/mod_speling.so
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
<IfModule prefork.c>
LoadModule cgi_module modules/mod_cgi.so
</IfModule>
<IfModule worker.c>
LoadModule cgid_module modules/mod_cgid.so
</IfModule>
ExtendedStatus On

### Section 2: 'Main' server configuration

User apache
Group apache
ServerAdmin admin@my-site.com
ServerName www.my-site.com:80

UseCanonicalName Off
DocumentRoot "/var/www/html"
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

<Directory "/var/www/html">
AllowOverride None
Order allow,deny
Allow from all
</Directory>
<LocationMatch "^/$>
    Options -Indexes
    ErrorDocument 403 /error/noindex.html
</LocationMatch>

<IfModule mod_userdir.c>
    UserDir disable
</IfModule>
```

```
DirectoryIndex index.html index.html.var
AccessFileName .htaccess
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
TypesConfig /etc/mime.types
DefaultType text/plain
<IfModule mod_mime_magic.c>
 MIMEMagicFile conf/magic
</IfModule>

HostnameLookups ON
ErrorLog logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog logs/access_log combined
ServerSignature On
Alias /icons/ "/var/www/icons/"
<Directory "/var/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
Alias /manual "/var/www/manual"
<Directory "/var/www/manual">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

<IfModule mod_dav_fs.c>
    # Location of the WebDAV lock database.
    DAVLockDB /var/lib/dav/lockdb

</IfModule>

<IfModule mod_cgid.c>

Scriptsock          run/httpd.cgid

  </IfModule>
```

```
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

<IfModule mod_negotiation.c>
<IfModule mod_include.c>
    <Directory "/var/www/error">
        AllowOverride None
        Options IncludesNoExec
        AddOutputFilter Includes html
        AddHandler type-map var
        Order allow,deny
        Allow from all
        LanguagePriority en es de fr
        ForceLanguagePriority Prefer Fallback
    </Directory>

</IfModule>
</IfModule>
```