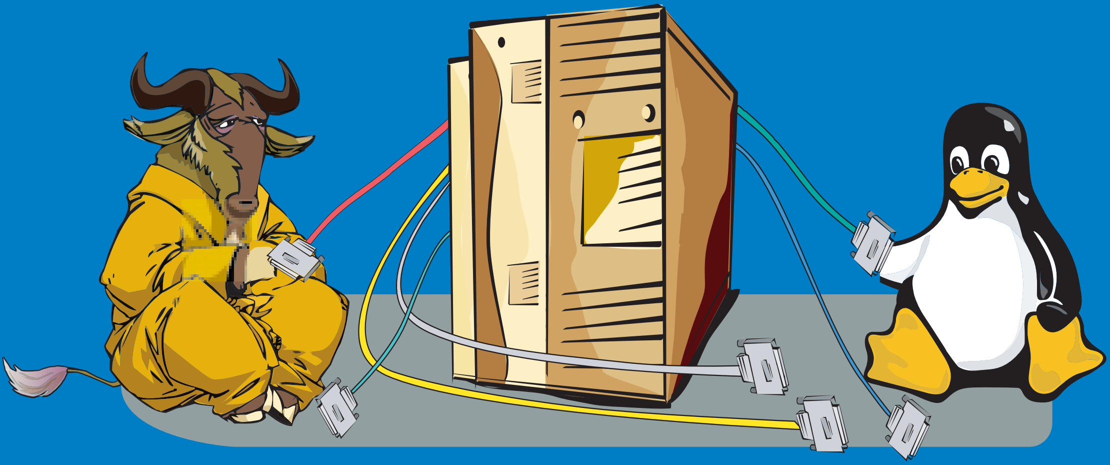
International
Open
Source
Network

Free/Open Source Software

Network Infrastructure and Security

Gaurab Raj Upadhaya



Foreword by
ROBERT SHAW

Asia-Pacific Development Information Programme
e-Primers on Free/Open Source Software

Free/Open Source Software

Network Infrastructure and Security

Gaurab Raj Upadhaya

Asia-Pacific Development Information Programme

e-Primers on Free/Open Source Software

ELSEVIER
A division of
Reed Elsevier India Private Limited
Shri Pratap Udyog, 274 Capt Gaur Marg
Sriniwas Puri, New Delhi 110 065
Tel: +91 11 42890200
Fax: +91 11 42890201
Web: http://www.asiaelsevier.com

ISBN-13 : 978-81-312-0421-4
ISBN-10 : 81-312-0421-9

TABLE OF CONTENT

FOREWORD

Whenever I'm asked why Free/Open Source Software (FOSS) is so important, I recount the story of Xanadu. Mind you, that's not story of the mythical Xanadu, the opulent 13th century summer capital of the Kublai (grandson of Genghis) Khan's empire that spanned all of Asia. I mean the software vision of Ted Nelson, a genius of the 1960s. That Xanadu is widely acknowledged as the inspiration behind the concept of hypertext, Apple's original Hypercard and the World Wide Web.

Although its original concept stemmed from technology rather than a business model, it is the latter that probably resulted in its demise. That's because Xanadu tried to become that era's digital rights management scheme; a toll-bridge solution. Each time someone clicked on a hyper link, a 2-cent charge would credit that person's account with a small percentage going to Xanadu. For nearly 25 years, between four and ten scientists worked on some aspect of the project. In the late 1980s, Dave Walker, the CEO of AutoDesk, bought into the vision and plowed through about US$ 24 million trying to make Xanadu a shippable product.

But others had a simpler and clearer vision. Tim Berners-Lee, a scientist at the European research centre CERN in Geneva, working with a few collaborators, engineered a completely new competing distributed hypertext system that became the World Wide Web. Tim lobbied and fought with board members at CERN who wanted to control and license the WWW technology to the rest of the world. He emerged victorious and an open standard became the standard. Berners-Lee correctly realized that the value of his making an open contribution of his invention was far more important to humanity than any possible value CERN could bring with stewardship. The rest of the story is, as they say, history and represents the ultimate example of how FOSS has the power to radically change our world.

During the negotiations leading to the first and second phases of the World Summit on the Information Society, there was a growing realization of many government policy makers, particularly those from developing economies, of how FOSS was an important tool to promote digital literacy and improve access to public services for their citizens. But thought leaders are now beginning to realize that the FOSS movement is but one manifestation of something more fundamental and new about how collective intellectual innovation can blossom once networked information infrastructures are in place. In Yale academic Yochai Benkler's new book, The Wealth of Networks, he notes:

"The change brought about by the networked information environment is deep. It is structural. It goes to the very foundations of how liberal markets and liberal democracies have co-evolved for almost two centuries…A series of changes in the technologies, economic organization, and social practices of production in this environment has created new opportunities for how we make and exchange information, knowledge, and culture. These changes have increased the role of non-market and non-proprietary production, both by individuals alone and by cooperative efforts in a wide range of loosely or tightly woven collaborations."

We should applaud the efforts of the Asia-Pacific Development Information Programme and the International Open Source Network for their promotion activities of FOSS. This FOSS e-Primer on Network Infrastructure and Security is one of series of simple 'how-to' guidebooks they have produced for the lay person.

A globally interconnected information network makes it clear that cyber security is a responsibility of all countries worldwide. But even in developed countries, trained cyber security professionals are lacking so developing economies are at a huge disadvantage. Lacking in resources and with few incentives, it can be argued that developing countries represent the weakest link in promoting global cyber security. This means we need to make special efforts to assist developing economies in adopting the "technology, processes and people" of cyber security. This book represents an excellent contribution toward that laudable goal.

Robert Shaw
Deputy Head, Strategy and Policy Unit
International Telecommunication Union

LIST OF ABBREVIATIONS

ACL	Access Control List
APDIP	Asia-Pacific Development Information Programme
APNIC	Asia Pacific Network Information Centre
APRICOT	Asia Pacific Regional Internet Conference on Operational Technologies
ARPANET	Advanced Research Program Agency Network
BIND	Berkeley Internet Name Domain
BO	Branch Office
ccTLD	Country Code Top Level Domain
CEO	Chief Executive Officer
CIDR	Classless Inter Domain Routing
DNS	Domain Name System
DoS	Denial of Service
FOSS	Free/Open Source Software
BSD	Berkeley Software Distribution
GPL	General Public License
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
gTLD	Global Top Level Domain
ICANN	Internet Corporation for Assigned Name and Numbers
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IOSN	International Open Source Network
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
LGPL	Lesser General Public License
MTA	Mail Transfer Agent
NAT	Network Address Translation
NetBEUI	NetBIOS Extended User Interface
NIC	Network Interface Card
NOS	Network Operating System
OS	Operating System
PCH	Packet Clearing House
POP	Post Office Protocol
RAM	Random Access Memory
RBL	Real Time Block List
RFC	Request for Comments
RIR	Regional Internet Registry
SANOG	South Asian Network Operators Group
SMTP	Simple Mail Transfer Protocol
SRI-NIC	Stanford Research Institute-Network Information Center
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UCE	Unsolicited Commercial E-mail
UDP	User Datagram Protocol
UNDP	United Nations Development Programme
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network

INTRODUCTION TO FOSS AND GNU/LINUX

Free/Open Source Software

Free/Open Source Software, or FOSS, is software that is liberally licensed to grant users the right to study, change and improve its design since its source code is made available.

The definition of free software, as enunciated by the Free Software Foundation, states that four essential freedoms should be contained within its license:

1. The freedom to run the program for any purpose;
2. The freedom to study and modify the program;
3. The freedom to copy the program so you can help your neighbour, and
4. The freedom to improve the program and release your improvements to the public, so that the whole community benefits.

Freedoms 2 and 4 require access to the source because studying and modifying software without source code[1] is extremely difficult and highly inefficient since humans cannot usually understand machine code or object code.

The Open Source definition published by the Open Source Initiative states that:

1. The license should not prohibit free redistribution;
2. The program must include source code and must allow distribution in source code as well as compiled form;
3. The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software;
4. The integrity of the author's source code and reputation must be maintained by requiring derived works to carry a different name or version number from the original software;
5. The license must not discriminate against any person or group of persons;
6. The license must not restrict anyone from making use of the program in a specific field of endeavour;
7. The rights attached to the program must apply to all to whom the program is redistributed, without the need for execution of an additional license by those parties;
8. The rights attached to the program must not depend on the program being part of a particular software distribution;
9. The license must not place restrictions on other software that is distributed along with the licensed software; and
10. No provision of the license may be predicated on any individual technology or style of interface.

Some examples of FOSS software include: the Linux kernel, GNOME Desktop and Free Berkeley Software Distribution (FreeBSD). The most well known and popular FOSS licenses include: GNU General Public

[1] Code written by a programmer in a high-level language and readable by people but not computers. Source code must be converted to object code or machine language before a computer can read or execute the program.

License (GPL), GNU Lesser General Public License (LGPL), BSD license, Apache License, MIT License and Mozilla Public License. For more information on FOSS, please refer to *FOSS: A General Introduction* by Kenneth Wong and Phet Sayo.[2]

GNU/Linux

GNU/Linux (also known as Linux) is a computer operating system. It is one of the most prominent examples of FOSS development. Initially, GNU/Linux was primarily developed and used by individual enthusiasts. Since then, GNU/Linux has gained the support of major corporations such as IBM, Sun Microsystems, Hewlett-Packard, and Novell for use in servers and is becoming popular in the desktop market. Proponents and analysts attribute this success to its vendor independence, low cost, security, and reliability.

In 1983, Richard Stallman founded the GNU Project. The goal of GNU was to develop a complete Unix-like operating system composed entirely of free software. By the beginning of the 1990s, GNU had produced or collected most of the necessary components – libraries, compilers, text editors, a Unix-like shell – except for the lowest level, the kernel. The GNU project began developing its own kernel, the Hurd, in 1990, based on the Mach micro-kernel. This Mach-based design subsequently proved difficult, however, and development proceeded slowly.

Meanwhile, in 1991, another kernel was begun as a hobby by Finnish university student Linus Torvalds while attending the University of Helsinki. Linux was further developed by various programmers over the Internet. In 1992, it was combined with the GNU system, resulting in a fully functional free operating system. The GNU system is most commonly encountered in this form, usually referred to as a 'GNU/Linux system' or a 'Linux distribution'. This is by far the most popular variant of GNU. There are over 300 distributions based on GNU with the Linux kernel. Some of the most widely used are: Debian, Gentoo, Mandriva, Red Hat Linux, Slackware, SuSE and Ubuntu.

The Linux kernel was originally envisioned to be used on Intel 80386-based machines. While not originally intended to be portable to other architectures, Linux is now one of the most widely ported operating systems in the world, running on a diverse range of systems from the iPAQ to IBM System z9. Specialized distributions exist for less mainstream architectures. Linux is increasingly common as an operating system for supercomputers. In November 2005, the Top 500 List of supercomputers stated that the two fastest supercomputers in the world run Linux. Of the 500 systems, 74 percent run some version of Linux, including seven of the top 10.

The Linux kernel and most GNU components are licensed under the GNU GPL. The GPL requires that all source code modifications and derived works also be licensed under the GPL, and is sometimes referred to as a "share and share-alike" (or copyleft) license.

Network Operator's Perspective on FOSS

All network operators need to provide a set of services for which they need reliable systems. Services such as web surfing, e-mail and Domain Name System (DNS) form the backbone of all network providers. Network operators also need scalability as the number of users grows, since they usually operate in a high volume business. This requires a reliable system which can be scaled up quickly and which can run services efficiently.

FOSS provides several advantages over proprietary software in terms of savings with regard to license fees, flexibility in installation and operation, and availability of source code that can act as reference implementations for the basic building blocks of Internet infrastructure. FOSS also reduces virus and security problems frequently associated with other software because of the possibility of public audit of source code. Finally, illegal copying of software is an increasing concern worldwide, and using FOSS ensures that you are always on the side of the law.

GNU/Linux for Networking

While there are other operating systems, like *BSD, and those that also provide access to source code, GNU/Linux is the most popular FOSS system in the world.

[2] http://www.iosn.net/foss/foss-general-primer/foss_primer_current.pdf

From a networking perspective, the UNIX operating system is the pioneer in multi-user systems. UNIX and UNIX clones such as *BSD and GNU/Linux have been developed with network performance and security in mind.

In the past, most network systems relied on expensive hardware and proprietary UNIX platforms to get maximum benefits. With GNU/Linux, the same kind of benefits can be achieved on cheaper hardware. At the same time, the availability of the source code enables operators to create their own extensions and fine-tune the software to their own needs.

Also, most servers and programs that are required for network operations are available on GNU/Linux, and a lot of additional utilities are being developed constantly to provide added advantages.

BSD Alternatives

While GNU/Linux has received wide recognition in the past few years, there are alternative open source operating systems, mainly the different types of BSD: NetBSD, FreeBSD and OpenBSD. The BSD license is one of the most widely used licenses for FOSS. Many software programs are released under this license, including Apache and Berkeley Internet Name Domain (BIND). This license has few restrictions when compared to licenses such as the GNU GPL, or even the default restrictions provided by copyright, putting it relatively closer to the public domain. Therefore, while preserving open access to the source code, the BSD license also allows the software to be used in proprietary software and products.

NETWORK CONCEPTS AND ARCHITECTURES

Computer Networks

A network is a mechanism that enables distributed computers and users to communicate and share resources. Sharing of resources may mean sharing a printer, a scanner or even a large hard disk on a particular computer. A computer network connects two or more computers or peripherals so that all users on the network have access to these resources.

There are two major types of computer networks:

- Server-based networks; and
- Peer-to-peer networks.

In a **server-based network**, there exists one or more computer(s) on which the resources are centralized. The network is also more or less controlled by the server. Servers generally have superior processing power, memory and hard disk space compared to desktop systems.

In a **peer-to-peer network**, all computers on the network are responsible for sharing resources. Each computer shares the resources that it has. All computers are servers as well as clients. There is no central control over the resources. Peer-to-peer networks are based on peer relationships among computers in the network.

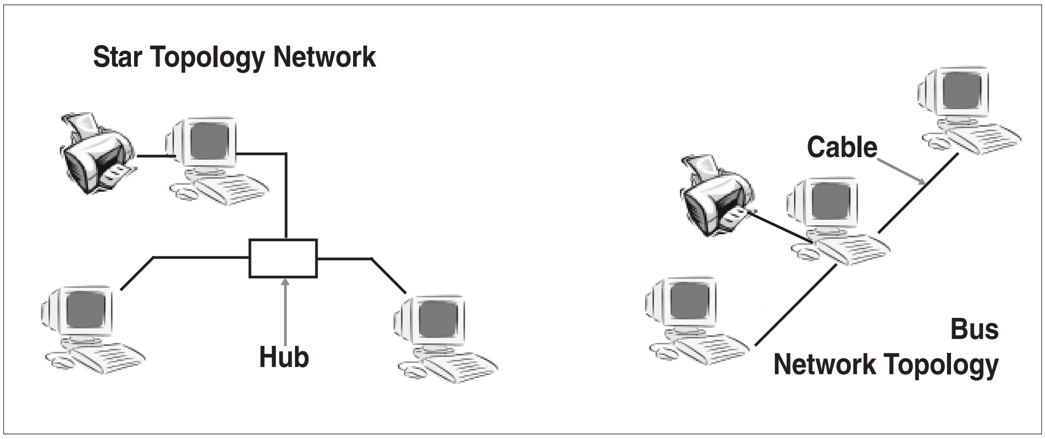
Network Topology

Topology refers to the arrangement of computers in a network. In physical terms, there are two main topologies that are commonly used:

- Bus topology; and
- Star topology

The simplest arrangement is **Bus topology** in which all computers are connected in a series to a single stretch of cable. This means that the entire length of the cable must be intact for the network to work properly; which makes larger networks prone to failure.

A hub or a switch is used as the central exchange point in a **Star topology**. All computers are connected to the hub in a star-like configuration of wires going in all directions from the hub. A Star topology is easier to maintain and to expand.



LAN Architecture

Two computers can communicate in different ways through different Local Area Network (LAN) architectures. Of the two well-known architectures — Ethernet and Token Ring — the former is more popular.

Ethernet

Ethernet is independent of any vendor, which accounts for its great success. Basic Ethernet architecture operates at a speed of 10 Mbps (mega bits per second). Other variations include Fast Ethernet, operating at 100 Mbps, and Gigabit Ethernet operating at 1 Giga bit per second. The Ethernet standards are referred to in the 802.nn notation, an Institute of Electrical and Electronics Engineers (IEEE) standard. WiFi (wireless fidelity) networks also follow the same standards, and are thus characterized as 802.11x.

Token Ring

Token-Ring LAN technology was developed and promoted by IBM in the early 1980s and was later standardized as IEEE 802.5. Initially very successful, its popularity went into steep decline after the introduction of the 10 Mbps Ethernet standard in the early 1990s.

Network Hardware

Network Cabling

Cables play a fundamental role in any network, and exist at the most primary level of the network. Two types of cables are in common use – co-axial cables (similar to cable TV cables) and twisted pair cables (similar to telephone cables). Co-axial cables are generally used with T-connectors in a Bus topology network. Twisted pair cables are used mostly in Hub-based networks.

Network Adapter Cards

Network Interface Cards (NICs) or Network Adapter Cards, as they are commonly known, provide an interface on the computer to connect to the network cable. Today, most computers come with a built-in network card. However, if it is not built into the system, an additional card can be inserted inside the computer, which will then connect to the network cable.

Networks Operation and Layers

Computer networks operate at different layers, each of which is independent of the others. The layers are stacked on top of each other, with each layer providing service to the layer above it. Network layers, as these layers are commonly known, provide general understanding of any computer network. A very simple form of these layers is illustrated below.

Operating Software Layer (with Internet Protocol)	Network Applications	Application Layer
	Transmission Control Protocol / User Datagram Protocol	Transport Layer
	Internet Protocol	Network Layer
Hardware Layer	Network Interface Card Drivers	Software/Hardware Layer
	Network Interface Card Layer	Electronic Layer

NIC Driver

An NIC driver provides a layer between the hardware and the software in any network. All network card manufacturers provide NIC drivers, which sometimes also come with the operating system. However, it is always better to use the driver that comes with the card.

Network Protocols

Network protocols provide a way for diverse computer hardware to communicate with each other in a standard way. These protocols follow set rules unique to themselves. Common network protocols are IP (commonly referred to as the Transmission Control Protocol/Internet Protocol or TCP/IP) and NetBEUI (NetBIOS Extended User Interface). Other protocols had been developed and used in the past, but IP has replaced most of these. NetBEUI is developed by Microsoft and is native to the Windows environment. IP is the Internet Protocol and is the most popular protocol today.

TCP/IP

TCP/IP are the basic protocols of the Internet. There are other protocols in TCP/IP like User Datagram Protocol (UDP). IP is the network layer and provides the basis for the network through the IP addresses. The IP address of each computer must be unique. TCP and UDP are the transport layers and are responsible for transport of network data packets.

IP addresses follow the format **nnn.nnn.nnn.nnn** (where **nnn** is less than 255). A network mask separates the network address and the host address.

For internal network use, the IP range defined in Request For Comments (RFC)[3] 1918[4] should be used. These are 192.168.0.0 to 192.168.255.255, 172.16.0.0 to 172.31.255.255 and 10.0.0.0 to 10.255.255.255

Components of a Network

Network Operating System

A Network Operating System (NOS) is a network capable operating system that handles network resources and provides services like file and printer sharing. It is also capable of being an application server.

Users and Groups

Any NOS provides some form of security to its resources and services, the simplest being the user name-password pair. All users on the network are given a unique user name and they are also given a secret password. Only after verifying that a proper user name and password have been supplied, does the NOS provide access to the designated network services.

A group is a collection of users in any network. A group is created to give similar rights to a group of users. Groups make administration and management easier.

Rights

Not all resources in a network may be available to everyone. It is necessary to shield some parts of the network from unauthorized use. So, all NOS provide rights to its users and groups. Users and groups are given only the required rights, while the more powerful rights are limited to the administrators.

[3] RFC stands for Request for Comments, which is the series of documentation for Internet Standards. RFC 1918 means RFC number 1918, which defines private use IP addresses. RFCs can be downloaded from http://www.faqs.org/rfcs/. We will refer to many RFCs for further reference throughout the e-primer.
[4] RFC 1918. ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt

Super User or Administrator

In most NOS, there is a super user or administrator who has absolute rights in the network system. In the UNIX and GNU/Linux environments, 'root' is the super user. An administrator is the supreme user in a Windows network. In NetWare networks, supervisors are super, meaning such users are capable of doing anything that the NOS permits. These accounts should be used with care.

Monitoring Network Performance

When users start complaining that the network is too slow, we know that there are problems in the network. Network performance must be monitored to find where the fault lies. It may lie in the network hardware or software.

Inferior grade cables and loose cable connections cause 80 percent of network problems and performance issues. High quality and prescribed cables and connectors should always be used. Another cause of problems may be the server hardware. Servers require more Random Access Memory (RAM). Some network performance problems call for the replacement of an older switch.

Unnecessary services running on the server should be stopped to get the maximum performance from the server. Every NOS provides some utility to monitor network performance, as well as server performance.

Troubleshooting the Network

The first step is to check the cables. After that, check the power connections to the network switch or resources on the network. If the problem is not solved, check the different services running on the server.

Role of FOSS in Networking

The Internet is the biggest example of the successful implementation of free and open source tools and software. It is the largest network in the world and its basis is the IP protocol. While it is true that the Internet has helped in the growth of FOSS in general, FOSS has also made the Internet a much more reliable network. Detailed information about the Internet Engineering Task Force (IETF), the protocol standards body of the Internet, and its implications as a FOSS reference implementation can be found in RFC 1718.

Limitations to the Use of FOSS

Yes, there are limitations. Most FOSS is produced in conformance to open standards and processes. There are proprietary software and processes that may not have a FOSS equivalent. Sometimes, it may also be necessary to use proprietary software to access certain types of hardware. Another limitation may be the rapid pace of development of FOSS. To an average user, this may be confusing but for service providers this is definitely an advantage.

MAJOR NETWORKING FUNCTIONS WITH FOSS

Domain Name System

DNS is the glue that keeps the technical side of the Internet connected to the users of the network. Simply stated, the DNS provides names to IP address translation and vice versa. The Internet is based on the IP protocol, which means that computers on the Internet know the path to each other only through the IP addresses. However, remembering the numerical IP address of each computer on the Internet is not possible for most people. Here is where DNS comes in handy.

The main purpose of DNS is to map names to objects on the Internet. The object may be an IP address or the identification of mail servers, name servers, and even personal telephone numbers. Names, after all, are easier to remember than numbers.

The earlier version of DNS was a file named hosts.txt, which was manually maintained by the Stanford Research Institute-Network Information Center (SRI-NIC) in the early days of the ARPANET[5] in the 1970s. This hosts.txt file was updated on a single computer and pulled by computers all over the world. While this method worked for some time, name collision became imminent when more hosts were added to the network. The flat single file mapping was simply not scalable. Thus DNS was created. It is defined in RFC 1034 and 1035.

DNS consists of three components:

- DNS name space – the domain names;
- DNS server – the server that hosts the name space; and
- Resolver – the client which uses the server.

The three components work in tandem to create a viable solution to name resolution. DNS is designed in such a way that all the data are maintained locally but retrievable globally. The data is distributed among different name servers and no single computer has all of the data. The data is internally always consistent, thereby providing a stable system. Moreover, DNS is designed in such a way that any device can send DNS queries to a server.

DNS Name Space

DNS name space is a concept. Names are references to addresses, objects and physical presences that constitute a human, understandable reference identifying an endpoint. DNS name space is the name space defined for DNS. On the Internet, domain names provide a hierarchy for DNS name space, and thus, an order to how Internet addresses are identified.

Let’s take a comparative example. If someone were to send me a letter, it would be addressed to

Gaurab Raj Upadhaya

205/8 Sahayogi Marg, Kathmandu, Nepal

In this example, the address provides a way in which the letter can be sent to me from anywhere in the world. Similarly, if someone were to send me an e-mail, they could use either of the following e-mail addresses:

gaurab@wlink.com.np
gaurab@lahai.com

[5] Advanced Research Program Agency Network is considered the precursor to the current Internet.

In both cases, while the e-mail ends up in the same mailbox, it travels differently to that address. What needs to be understood is the domain hierarchy that makes the e-mail work. It is part of the DNS name space. Domain names are the implementation of DNS name space. The domain name system uses an inverted tree-shaped structure. The topmost level of the DNS tree is called the 'root'.[6] The 'root' is referenced with a ' . ' (dot). Immediately below the 'root' are the Country Code Top Level Domain (ccTLD) and the Global Top Level Domain (gTLD). These two top levels are predefined and fixed on a global scale. The ccTLDs are assigned as per the ISO 3166 standard. The gTLDs are decided by the Internet Corporation for Assigned Name and Numbers (ICANN). Examples of ccTLDs are .np, .in, .my, .uk, .se, etc. ccTLDs are always two letter codes. Examples of gTLDs are .com, .org, .net, .gov, .edu, .mil, .info, .name, and .aero.

Domains and Sub Domains

Below the TLDs are the user level spaces. These are commonly referred to as the domain names. For example, anything under .net is in the net domain, and anything under .uk is under the UK domain. And by extension, a sub domain under lahai.com, such as evo.lahai.com, is under the lahai.com domain.

Every domain created under an upper level domain is referred to as a sub domain. Thus, in the example above, evo.lahai.com is a sub domain under lahai.com.

Zones and Delegation

In computer terms, each DNS name space is reflected by its zone file. It is also referred to as the 'administrative name space'. Each domain or sub domain on a name server has its own zone file, which is the main file that provides the mapping.

What makes DNS so scalable is its ability to define a delegation for sub domains to other servers and other zone files. Thus, the root zone file delegates ccTLD and gTLD functions to their respective servers, and each ccTLD or gTLD server further delegates specific domain information to their registered owners. Thus, the actual names to object mapping will be provided only by the authoritative zone for that domain. This procedure can be compared to a parent delegating authority to his/her child.

Name Servers

Name servers host the DNS zone files. They answer queries directed to them. Name servers are of two types.

1. Authoritative name servers
   - master
   - slave
2. Non-authoritative name servers
   - caching name servers
   - caching forwarders

Most implementations are a combination of two or more types.

Authoritative Name Servers

Authoritative name servers host the main zone files for the designated domain. The authority of the name server is based on delegation from the upper level domain. Thus for any server to be authoritative for evo.lahai.com domain, it has to be delegated in the lahai.com zone file.

The master file is where the main file is hosted. The slave mirrors the file from the master. There can be multiple slave servers to one master server. A single master server can support more than 20 million names, but it might not be a good idea to actually do this. Different DNS server software are capable of

[6] Not to be confused with the 'root' user on GNU/Linux and Unix systems.

handling large numbers of DNS queries. A commonly cited example is 300,000 queries per second. Changes in the master copy of the database are replicated to the slaves immediately or according to timing set by the administrator.

Recursive Name Server

A recursive name server is not authoritative for all domains for which it is serving data. It acts on behalf of other clients, and caches the result in its memory. If the same query is sent within a predefined time period, then instead of searching the entire DNS structure, it serves the data from the cache. In case of caching forwarders, the server uses another DNS server to get the result. When the data are forwarded to the client, they are marked as non-authoritative.

Mixed Implementation

In smaller organizations, a single name server can be used for multiple purposes. A server can be authoritative for a select few domains but it may also serve as a non-authoritative caching server for other domains. With recent cases of DNS cache poisoning, it is strongly recommended that the same server not be used for both authoritative and caching functions.

Resolver

The resolver is the client that asks the server for the DNS data. The resolver is normally implemented at the operating system level in the form of a library, so that multiple applications can use it.

DNS Security

A big strength of the Internet is in the user's ability to use names to reach the right servers and systems. A mis-configured DNS or a malformed DNS query can deter users; hence, the need for a secure DNS system. It is very important to follow these simple points:

- Allow only authorized systems to do zone transfers from the master server.
- Have a minimum of two DNS servers, and remember not to put them in the same location.
- Make sure that your forward and reverse DNS information is consistent.
- Follow current best practices for DNS implementation.

Using BIND for DNS

BIND is an implementation of the DNS protocols. It provides an openly re-distributable reference implementation of the major components of the domain name system, including:

- A DNS server (named);
- A DNS resolver library; and
- Tools for verifying the proper operation of the DNS server.

The BIND DNS server is used on the vast majority of name serving machines on the Internet, as it provides a robust and stable architecture on top of which an organization's naming architecture can be built. The resolver library included in the BIND distribution provides the standard interface for translation between domain names and Internet addresses and is intended for linking with applications requiring name service.

Getting and Installing BIND

BIND is normally installed by default by most GNU/Linux distributions. Otherwise, you can always get a copy from the BIND home page at http://www.isc.org/products/BIND/. The installation process of BIND in different distributions of GNU/Linux may differ. It is best to follow the distribution guide for installation.

Configuration of BIND

The BIND configuration has to be undertaken in three stages.

First, the client side or the resolver library needs to be configured, followed by the server itself and finally, the tools.

Resolver configuration

The resolver is the client side of the DNS system. Even if you do not run a DNS sever on the computer, you will need to have the resolver installed. Naturally, in order to configure BIND, you first need to configure the resolver library. This is done by configuration of the following files:

```
/etc/host.conf
```

This file specifies how the host name resolution is performed. This has been made obsolete, but older installations may still use it.

```
/etc/nsswitch.conf
```

This file has replaced host.conf. It specifies the order in which name resolution takes place. It tells the computer the order in which it should try to convert names into IP addresses.

```
# /etc/nsswitch.conf
```

Any line starting with a '#' sign is a comment.

# In this example, hosts are resolved through DNS, and then from files.

```
hosts: dns files
```

# only files are used for network name resolution

```
networks: files
```

The default file created during installation is usually sufficient.

```
/etc/resolv.conf
```

resolv.conf is the basic DNS configuration file, which specifies the DNS server and the domain name. The three key words here are 'domain', 'search', and 'nameserver'.

```
# /etc/resolv.conf
# Our domain
domain gaurab.org.np
# Default search domains in order of priority
search gaurab.org.np lahai.com.np
#
# We use the local server as the first name server.
nameserver 127.0.0.1
# we have second name server at up stream provider.
nameserver 206.220.231.1
```

This file is also created during the installation, and if your network configurations have not changed, you can leave it unchanged.

Server configuration

'**named**' is the best known FOSS DNS daemon. A daemon is a software program that runs continuously on the server as a service. The DNS server is thus commonly referred to as the 'name daemon' or just 'named'. The file name of the DNS server is also 'named'.

For BIND versions 4.x.x, named used the configuration file/etc/named.boot. But in the later versions of BIND (8.x.x), the configuration file is /etc/named.conf.

For our purposes, we use /etc/named.conf. In the following example, a master DNS for the domains lahai.com and gaurab.org.np is specified. A slave DNS for domain wlink.com.np is also shown.

```
//
// /etc/named.conf file for ns.lahai.com
// in this file '//' is the comment.

// you specify the default data directory for DNS. Now all DNS
// related files should go into /var/named or any other
// directory as specified.

options {
directory "/var/named";
};

// First you need to add the DNS root zone file name. It's there
// by default.

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

// Now we are specifying a master domain called lahai.com
// whose information is stored in the file 'named.lahai.com'

zone "lahai.com" {
type master;
file «named.lahai.com»;
};

// the whole thing can also be done in a single line.

zone "gaurab.org.np" { type master; file "named.gaurab.org.np";
};

// Now this server is also a slave for another domain
"wlink.com.np'

zone "wlink.com.np" { type slave; masters { 202.79.32.33; };
file "slave/named.wlink.com.np"; };

zone "0.0.127.in-addr.arpa" { type master; file "named.local"; }
```

This file sets the stage for adding the real data about host name and IP addresses. The /etc/named.conf file can take a lot of additional configuration directives, but these will not be discussed here.

After relevant entries have been made in the named.conf file, it is necessary to create the host name records for the corresponding domains. All of the files should be placed in the directory specified by the directory directive in the named.conf file.

The **named.local** file provides reverse zone lookup for the loopback interface or the 127.0.0.0 network used by the loopback addresses. The default file should be left unchanged. The **named.ca** provides the root server information to the DNS server. The default should never be edited. Now let's look at a sample DNS file for the domain lahai.com (**named.lahai.com**)

```
; file /var/named/named.lahai.com

@ IN SOA ns.lahai.com. gaurab.lahai.com. (
2004050801 ; serial number
86400 ; refresh: once per day (1D)
3600 ; retry: one hour (1H)
3600000 ; expire: 42 days (6W)
604800 ; minimum: 1 week (1W)
)
```

```
; we are specifying three Name servers.

IN NS ns.lahai.com.
IN NS a.ns.hopcount.ca.
IN NS ns1.lahai.com.

; local mail is distributed on another server

IN MX 10 mail.lahai.com.
IN MX 20 ns.lahai.com.
; loopback address
localhost. IN A 127.0.0.1

; The glue records so that the NS records can resolve.

ns IN A 204.61.208.110
ns1 IN A 202.79.55.14

; main DNS entry
www IN A 207.189.222.2
mail IN A 202.51.76.8

; Aliases for the www machine.

tftp IN CNAME www
```

The above file is the main file for the domain 'lahai.com'. If you want to add additional names for the domain 'lahai.com', like pop3.lahai.com and smtp.lahai.com, then you should add them in the above file.

The order in which the name servers are listed in this file makes them master and slave. The first NS is always the master server and the other two are slave servers. Each time the master server is updated, it can automatically send a notification to the other NS servers listed in the file. This parameter can be configured in the named.conf file.

A note about Reverse DNS

The majority of DNS-related problems are usually due to mis-configured reverse DNS. Reverse DNS is the mapping of numbers into names, or the opposite of the forward name resolution. Many applications use this facility to verify that the network source IP address is valid. A common example is SPAM or unsolicited commercial e-mail (junk e-mail) prevention software, which may refuse to accept mails from any domain that does not have a reverse DNS configured.

The reverse DNS works through delegation of the particular group of IP addresses from one of the Regional Internet Registries (RIRs), which is Asia Pacific Network Information Centre (APNIC) (http://www.apnic.net) in the Asia-Pacific region. Since Internet Service Providers (ISPs) are normally APNIC members, they are responsible for configuring the appropriate reverse DNS for the IP addresses being used by them and their clients.

Since each computer has its own loopback interface and the IP address associated with it, BIND comes with the default installation of the **named.local** file, which is the reverse DNS for 127.0.0.0 network. This file looks like the following:

```
; /var/named/named.local

$TTL 86400
@ IN SOA localhost. root.localhost. (
1997022700 ; Serial
28800 ; Refresh
14400 ; Retry
3600000 ; Expire
86400 ) ; Minimum
IN NS localhost.

1 IN PTR localhost.
```

Administrating BIND DNS

BIND includes a utility called **rndc** that allows you to administer the named daemon, locally or remotely, with command line statements. The rndc program uses the '**/etc/rndc.conf'** file for its configuration options, which can be overridden with command line options.

Before you can use the rndc, you need to add the following to your named.conf file:

```
/etc/named.conf

controls {
inet 127.0.0.1 allow { localhost; } keys { <key-name>; };
};

key "<key-name>" {
algorithm hmac-md5;
secret "<key-value>";
};
```

In this case, the *<key-value>* is an HMAC-MD5[7] key. You can generate your own HMAC-MD5 keys with the following command:

```
dnssec-keygen -a hmac-md5 -b <bit-length> -n HOST <key-file-name>
```

A key with at least a 256-bit length is a good idea. The actual key that should be placed in the *<key-value>* area can be found in the *<key-file-name>*.

Configuration file /etc/rndc.conf

```
options {
default-server localhost;
default-key "<key-name>";
};

server localhost {
key "<key-name>";
};

key "<key-name>" {
algorithm hmac-md5;
secret "<key-value>";
};
```

The <key-name> and <key-value> should be exactly the same as their settings in /etc/named.conf.

To test all of the settings, try the rndc reload command. You should see a response similar to this:

```
rndc: reload command successful
```

You can also use the rndc reload to reload any changes made to your DNS files.

DNS Tools

There are two common tools to test DNS: **nslookup** and **dig**. Nslookup is the older of the two, and is less preferred. You can use the **dig** utility to test the DNS service. Use the command '**man dig'** on most Unix and Unix-like systems to access the relevant manual pages.

[7]A popular way to encrypt. It uses a one way hash algorithm for encryption.

The Mail Server

Internet and e-mail were considered synonymous in the early days of the Internet. Even today, more than a quarter of the total Internet traffic is still e-mail, and it is not surprising that FOSS rules the world of e-mail. On the Internet, e-mail messages work on the basis of Simple Mail Transfer Protocol (SMTP) which is defined in RFC 2821. SMTP is a really simple protocol designed to make the transfer of e-mail messages between mail servers as easy as possible.

SMTP works in plain text, and communicates between the mail servers, which are also referred to as Mail Transfer Agents or MTAs. The most popular mail server software is 'sendmail'. Other examples are exim, qmail, and postfix. The closed source alternatives are Lotus Notes and Microsoft Exchange.

The beauty of FOSS is the level of software complexity available. While exim and postfix have a smaller footprint and consume a small amount of memory on the servers, sendmail is a complex beast that runs the busiest mail servers of the world.

Another important benefit of FOSS mail servers is the modularity of the software. Sendmail itself provides for a large number of extensions and provision for including modules. This makes it easier for developers to extend the software for their in-house needs. If you need to develop an extension to your e-mail server to automatically handle different types of e-mail, FOSS is a better option.

Other Mail-related Protocols

The two main mail-related protocols are Post Office Protocol (POP) and Internet Mail Access Protocol (IMAP). These provide the end-user functionality for users. POP and IMAP are used by e-mail software for accessing e-mails stored on a server. So if you use an e-mail client like Eudora or Thunderbird, then it will use either POP or IMAP to pull e-mail from your mail server to the local machine.

Handling Spam

Unsolicited Commercial E-mail (UCE) or spam is increasingly a big problem for all service providers. Most mail server software now have at least minimum anti-spam features.

Incoming spam

Spam Assassin is a popular software used to filter incoming spam. It can be invoked for either a single user or the whole system, and provides the ability to configure a complex set of rules to detect and delete incoming spam.

Stopping outgoing spam

It is also the duty of the provider not to let spammers use their network for sending mail. Mis-configured mail servers that allow for open-relay are some of the biggest sources of spam. Increasingly, mail servers are configured not to be open-relay by default. Virus-infected computers are also another source of spam.

Anti-spam Features

One of the biggest advantages of FOSS is its extensibility. Nothing highlights this more than the anti-spam features available in mail servers. Today, almost 80 percent of all e-mail messages are thought to be UCE, commonly referred to as junk e-mail or simply spam. UCE not only consumes a lot of bandwidth and network resources, it is also a nuisance to users and decreases productivity for organizations.

The best anti-spam tools available today are all FOSS. In order to stop junk e-mail, it is necessary to identify their origin, and what better way of doing this than thousands of users collectively identifying spammers. The FOSS concept makes sure that not a single junk e-mail goes unreported so that the origin can be identified easily.

A common anti-spam technique is the use of Real Time Block Lists or RBLs. Different RBLs list the IP addresses of networks that are known to be the origin of huge amounts of spam. Again, the open nature of these lists, as well as software like Spam Assassin, makes it easier to tune the software to a user's own needs.

For example, in a corporate environment, the users needed to send e-mail in capital letters due to the nature of their work. Now, if the entire e-mail were in capital letters, most anti-spam tools would identify it as junk e-mail. However, if we use FOSS solution, we can modify the code and remove this criterion for mail originating from within the network.

Using Sendmail for SMTP

Sendmail is one of the SMTP servers available under Linux. It is also one of the oldest open source software that is widely used. Many people consider sendmail to be too complicated and difficult to use. Sendmail has its advantages and disadvantages. Because it has many features, it is a complex piece of software. But, at the same time, the basic operations of sendmail can be managed easily.

Sendmail configuration is handled either by directly editing the sendmail configuration file (not recommended) or through the use of the M4 macro language in creating a new configuration file from a set of variables.

Now we will deal with sendmail. Given below are the minimum necessary changes to the default sendmail installation.

Enabling Network-ability in sendmail

Default installation of software in many distributions is limited to the mail server listening only on the loopback address,[8] i.e, the server is not reachable over the network. For sendmail to be reachable from the network, you will need to edit the appropriate line in the **/etc/mail/sendmail.mc** file. You should edit to remove 127.0.0.1 from the following line

```
DAEMON_OPTIONS (‘Port=smtp, Name=MTA’)
```

After that you will have to run the m4 macro to create new sendmail configuration files

```
[root@mail /etc/mail]# m4 sendmail.mc > sendmail.cf
[root@mail /etc/mail]# service sendmail restart
```

This should enable network reachability for the sendmail daemon. There are also a lot of other options on the sendmail.mc file that you can play around with.

Local Domain Names

Edit /etc/mail/local-host-names and add all domain and domain aliases that your site uses.

```
# local-hosts-names -
# include all aliases for your machine here.
lahai.com
gaurab.org.np
ns.lahai.com

# some examples
mail.you.com
yoursite1.com
mail.yoursite1.com
yoursite2.com
mail.yoursite2.com
yoursite3.com
mail.yoursite3.com
```

These are necessary so that sendmail accepts mail for these domains.

[8] The IP address of the loopback interface is 127.0.0.1

Virtual Domain Users

However, the above configuration does not entirely solve the problem of virtual domain users. For that, use the virtusertable feature. Go to /etc/mail/virtusertable

```
# /etc/mail/virtusertable

#virtual e-mail address real username

user1@yoursite1.com user1_yoursite1

# for domain pop, i.e, all e-mail in a domain into a single
account

@yoursite2.com yoursite2
```

Be sure to restart the sendmail daemon after making the changes.

```
[root@mail /etc/mail]# service sendmail restart
```

Access Control

Sendmail provides an access control feature through the /etc/mail/access file.

```
# /etc/mail/access

spam@cybermail.com REJECT
aol.com REJECT
207.46.131.30 REJECT
postmaster@aol.com OK
linux.org.np RELAY
192.0.2. OK
```

OK	Accept the mail message.
RELAY	Accept messages from this host or user even if they are not destined for our host; that is, accept messages for relaying to other hosts from this host.
REJECT	Reject the mail with a generic message.

It is required that you allow RELAY from your own network. Otherwise, computers on the network using the server as their outgoing SMTP server will not be able to send e-mail.

Running sendmail as System Daemon

The script is located at /etc/rc.d/init.d/sendmail and is started automatically when the computer is started. You can also start it using other commands

```
[root@mail /etc/mail]# /etc/init.d/sendmail start
[root@mail /etc/mail]# service sendmail restart
```

Running sendmail from xineted

It is a good idea (from a security standpoint) to have sendmail run from xinetd.conf and not as a standalone daemon. For that we need to add it to /etc/xinetd.d directory and remove it from /etc/rc.d/init.d, and then add the sendmail queue processing to cron. Here is what you have to do:

1. When using xinetd, create a file sendmail in /etc/xinetd.d/ similar to

```
default: on
service sendmail
{

socket_type = stream
wait = no
user = root
server = /usr/bin/sendmail -bs
}
```

2. Edit /etc/rc.d/init.d/sendmail to have exit 0 somewhere in the very beginning (this might not be the best way, so be sure to document the changes you do to these files) so that this file does nothing other than start sendmail.

3. By editing your (root's) crontab[9] (to edit use crontab -e), add a line like this

```
*/20 * * * * /usr/sbin/sendmail -q
```

That would process the sendmail queue every 20 minutes (if it exists).

Other Mail Servers

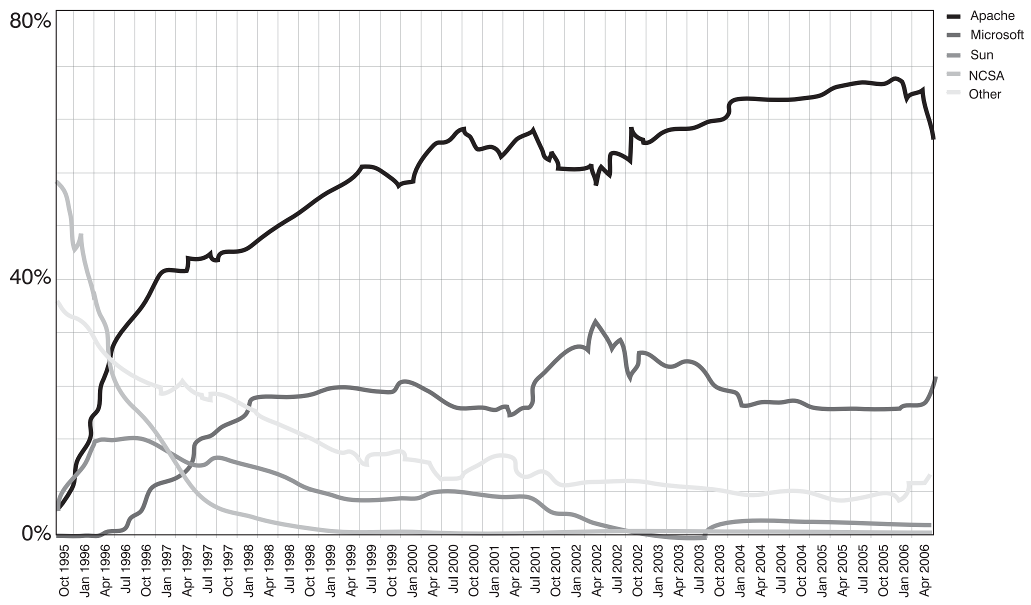
The other popular mail servers are postfix, exim, and qmail. While postfix is shipped as a default on a few Linux distributions, many small service providers have adopted exim because of its simplicity and robustness. Exim also has strong anti-spam features built into it.

The Web Server – Apache

The dominance of FOSS in the web server market is well known. The Apache web server is the undisputed leader in web server surveys conducted on the Internet. It also has many strengths: it is the leader in introducing name-based virtual hosts, and the first truly modular web server that would work seamlessly with database servers. It also has fully built authentication, authorization and access control functions, as well as scripting support.

April 2006 Web Server Survey

The latest survey statistics are available at http://news.netcraft.com/archives/web_server_survey.html



[9] Cron is a software used to schedule different other software on the computer.

Apache also fully integrates with OpenSSL, which provides a secure sockets layer,[10] thereby enabling use of the Apache web server for e-commerce and secure transaction. And, best of all, it can be fully configured through a text-based configuration file – httpd.conf.

Configuring Apache

Configuring Apache is also fairly easy, unless you want to run complex software on the web server. The configuration files are usually located by default in '/etc/httpd/conf'. Additional configuration files are located in '/etc/httpd'. It is also common for Apache to be installed in '/usr/local/apache'.

The main configuration file for Apache is the httpd.conf. Apache usually works out of the box.

The httpd.conf File

Directives are the settings that define how Apache should actually run, where files are located on your server, how much of the machine's resources Apache may use, which content visitors are allowed to see, how many concurrent visitors the server can handle, and other parameters.

Let's look at the main directives:

```
/etc/httpd/conf/httpd.conf - the main configuration file
```

Server identification

ServerName: construct self-referential URLs
ServerAdmin: e-mail of server administrator displayed on error messages

File locations

DocumentRoot - the location where the static content of your website lives
ServerRoot – for relative location of files that do not begin with a slash "/"
ErrorLog – to log server-wide error messages
PidFile – contains process ID of the httpd process
Alias – to serve files outside the DocumentRoot
ScriptAlias – the location for CGI scripts, dynamic content generators
DirectoryIndex - the file specified is displayed by default
Userdir – to serve files from public_html dir in user's home dir as http://www.site.com/~user

Process creation

MaxClients – the number of simultaneous connections allowed from clients
Server-Pool Regulation - Apache under UNIX is multi-process; it balances the overhead required to spawn child processes with system resources. You can change setting such as MinSpareServers, MaxSpareServers, StartServers, MaxClients to fine tune the performance of the server
User,Group – set the privileges of the Apache child processes

Network configuration

BindAddress - restricts the server to listening to a single IP address
Listen - specifies multiple IP addresses and/or Ports
KeepAlive - an extension to HTTP, which provides a persistent connection
Port – the TCP port number the web server runs on; can be changed to an unused port

URL redirection:
To redirect requests to another URL Redirect permanent /foo/ http://www.example.com/bar/

[10] Secure Sockets Layer (SSL) encrypts data that is travelling over the public network.

Virtual Hosts

Virtual hosts is the practice of maintaining more than one web server name on one physical machine. For example, the same physical machine can host both the http://www.apdip.net and http://www.iosn.net.

Parameters are specific to a virtual host, which overrides some of the main server configuration defaults. There can be two types of virtual hosts – IP-based and name-based.

In an IP-based virtual host, the IP address of the connection is used to determine the correct virtual host to serve. The approach requires a separate IP address for each virtual host.

In name-based virtual hosts, the host names are sent as part of the HTTP headers, which means that many different hosts can share the same IP address. However, you will need to map each host to the IP address in DNS. This eases the demand for scarce IP addresses. Name-based virtual hosts cannot be used with SSL secure servers and older software may not be compatible.

Virtual host directives

- NameVirtualHost – designate the IP address and port number to listen (optional)
- <VirtualHost> – same argument as NameVirtualHost
- ServerName – designate which host is served
- DocumentRoot – where in the file system the content for that host lives
- ServerAlias – make the host accessible by more than one name

Here is an example.

NameVirtualHost *

```
<VirtualHost *>
ServerName www.domain.tld
DocumentRoot /www/domain
</VirtualHost>

<VirtualHost *>
ServerName www.otherdomain.tld
DocumentRoot /www/otherdomain
serverAlias otherdomain.tld *.otherdomain.tld
</VirtualHost>
```

If no matching virtual host is found, then **the first listed virtual host** that matches the IP address will be used. All other standard Apache directives can be used inside the virtual host directive.

Access Control per Directory using .htaccess file

.htaccess file is a text file containing Apache directives

AccessFileName .htaccess …in httpd.conf

.htaccess file contents

```
AuthName “restricted stuff”
AuthType Basic
AuthUserFile /usr/local/etc/httpd/htusers
AuthGroupFile /usr/local/httpd/htgroup
require valid-user
require group staff
require user lahai gaurab
```

```
        AuthName “restrict posting”
        AuthType Basic
        AuthUserFile /usr/local/etc/httpd/htusers
        AuthGroupFile /usr/local/httpd/htgroup
        <Limit POST>
        require group admin
        </Limit>
```

htpasswd – to manage users for access control

htpasswd -c /usr/local/etc/httpd/users martin

htpasswd /usr/local/etc/httpd/users ritesh

```
        /usr/local/etc/httpd/htusers contents:
        martin:WrU808BHQai36
        jane:iABCQFQs40E8M
        art:FAdHN3W753sSU

        /usr/local/httpd/htgroup contents:
        staff:martin jane
        admin:lahai gaurab
```

References

httpd.apache.org/docs/vhosts/index.html
httpd.apache.org/docs/vhosts/examples.html

Proxy and Web Caching with Squid

As with mail servers and web servers, FOSS also set the standard in the area of proxy and cache servers. Squid is synonymous with proxy services in the networking world. It is a very modular, high performance proxy and web caching server. The Squid website is http://www.squid-cache.org. Squid proxy caches can be clustered to provide much better speed and access. Squid cache was also one of the first cache systems to implement a hierarchical cache system.

Some advantages of Squid:

- High-performance proxy caching server for web clients
- A full-feature web proxy cache
- Designed to run on UNIX systems
- Free, open source software
- Handles all requests in a single, non-blocking I/O-driven process
- Keeps meta data and especially hot objects cached in RAM
- Caches DNS lookups
- Implements negative caching of failed requests
- Supports SSL, extensive access controls, and full request logging
- Using ICP, caches can be arranged in a hierarchy or mesh for additional bandwidth savings

Squid consists of:

- Main server program *Squid*
- DNS lookup program *dnsserver for faster DNS lookups*
- Optional programs for rewriting requests and performing authentication

- Some management and client tools

squid.conf – the Main Configuration File

- Default configuration file denies all client requests
- Configure to allow access only to trusted hosts and/or users
- Carefully design your access control scheme
- Checks it from time to time to make sure that it works as you expect
- People will abuse it if the proxy allows access from untrusted hosts or users
- Makes their browsing anonymous
- Intentionally uses your proxy for transactions that may be illegal
- Websites exist with a list of open-access HTTP proxies

Here is an example of the Squid configuration: to run basic Squid, the only thing configurable is the proxy port. The default Squid proxy port is 3128 but you can always change it.

```
Network options
http_port port Hostname: port
```

Squid Access Control

Squid is better known for its complex access control system. You can allow and restrict access based not only on IP addresses but also on domain name. The use of regular expression lets you create complex rules for access through the proxy server. For access control in Squid, a sophisticated access control system, similar to the one used in routers, is used. It is basically a two-step process:

1. Defining the access listed through use of the acl command; and
2. Allowing or denying access based on the access list created earlier.

i. acl used for defining an Access List. ‘
 ‘acl’ literally stands for access control list. The default ACLs are:

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 80 21 443 563 70 210 1025-65535
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT
```

ii. http_access – to control http access to clients
 If there are no “access” lines present, the default is to allow the request.

```
Default
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access deny all
```

The “deny all” line is very important.

Examples

Restrict access to work hours (9 am-5 pm, Monday to Friday) from IP 192.168.2/24

```
acl ip_acl src 192.168.2/24
acl time_acl time M T W H F 9:00-17:00
http_access allow ip_acl time_acl
http_access deny all
```

Rules are read from top to bottom

```
acl xyz src 172.161.163.86
acl morning time 06:00-11:00
acl lunch time 14:00-14:30

http_access allow xyz morning
http_access deny xyz
http_access allow xyz lunch
```

Be careful with the order of allowing subnets

```
acl mynetwork src 10.0.0.0/255.0.0.0
acl servernet src 10.0.1.0/255.255.255.0

http_access deny servernet
http_access allow mynet
```

Always_direct and never_direct tags

```
# always go direct to local machines
always_direct allow my-iplist-1
always_direct allow my-iplist-2
# never go direct to other hosts
never_direct allow all
```

After all of the http_access rules, if access is not denied, then it is allowed.

If none of the "http_access" lines cause a match, then the default is the opposite of the last line in the list.

It is a good idea to have a "deny all" or "allow all" entry at the end of your access lists.

iii. cache_dir: dir to store cached data

```
cache_dir /usr/local/squid/cache/ 100 16 256
```

Can support more than one disk with multiple mount points

```
cache_dir /usr/local/squid/cache1/ 10000 16 256
cache_dir /usr/local/squid/cache2/ 20000 16 256
```

iv. cache_mgr: e-mail of the Cache Admin

Appended to the end of error pages returned to users

```
cache_effective_user squid
cache_effective_group squid
Changes user and group ID's once it has bound to the incoming
network port
ftp_user: set the e-mail address that is used for FTP proxy
```

Client: Connects to a cache and requests a page, and prints out useful timing information

v. Squid logs

```
/usr/local/squid/logs/cache.log
/usr/local/squid/logs/access.log
```

Transparent Caching/Transparent Proxy

This picks up the appropriate packets, caches requests and solves the biggest problem with caching, which is getting users to use the cache server in a transparent way. Four factors need to be considered:

- *Correct network layout* - all network traffic needs to pass through a *filter device*
- *Filtering*: filtering out the appropriate packets
- *Kernel transparency*: redirecting port 80 connections to Squid
- *Squid settings*: Squid needs to know that it is supposed to act in transparent mode

A detailed explanation of how to achieve transparent proxy is available at http://www.linuxdoc.org/HOWTO/mini/TransparentProxy.html.

SECURITY FUNCTIONS WITH FOSS

Security Paradigms, Security Policies and Infrastructure Issues

Network and information security is one of the most important considerations when designing and deploying a network. Before we can get into the details of network security implementation with FOSS, we need to look at the different security paradigms that have been in existence for some time now. The older paradigm supported the notion that perimeter security was the ultimate form of security for any network. That myth has now been debunked, and the new paradigm advocates that every device must be made secure in itself in addition to the perimeter security imposed by firewalls and packet filters.

Security as a technical solution is also no longer a valid assumption. Security now involves not only stopping malicious packets from entering your network, but also preventing these malicious packets from coming from your network; deploying tools to detect intrusion attempts and generate logs; intelligent network sniffers and use of an individualized security policy; centralized logs to keep on top of information overload (or log overload), centrally managing the configuration firewalls and routers to detect changes (configuration management and security policy management); and host security tools to ensure that the last frontier of the protection – the host itself – is protecting data and is not compromised. All of these form a web of security that will effectively protect the organization, provided you know what you want to protect and how you want to protect it. This is to be articulated as a security policy.

The security policy is now one of the most important security tools for any organization as it helps in maintaining consistency in security implementation. Without the use of a security policy, it becomes difficult for an administrator to decide what to focus on. It also helps an organization to take action against people who defy the policy. In the absence of any such policy, work is always done on an *ad hoc* manner, which can create more possibilities for security breaches.

There is no one solution that can solve your security problems. What is more useful are best practices that document ways in which common security problems can be avoided. There are also security best practices for when a security breach occurs. Let's look at common reasons for security breaches and some of the best practice remedies.

Network Security Risks

Open Architecture of TCP/IP

While TPC/IP is a highly efficient, cost-effective and flexible communications protocol for local and global communications, and is widely adopted on the global Internet and in the internal networks of large corporations, it was designed 20 years ago when the Internet consisted of a few hundred closely controlled hosts with limited security. The Internet now connects millions of computers that are controlled by millions of individuals and organizations, and where the core network is administered by thousands of competing operators. This complex network spans the whole globe, connected by fibres, leased lines, dial-up modems, and mobile phones. Thus, while it is tolerant of random errors, TCP/IP is vulnerable to a number of malicious attacks.

The basic premise of 'end-to-end' connectivity on the Internet means that you are as likely to receive illegitimate traffic as legitimate traffic. Thus the scope of security is not to stop all traffic, but to detect and deter the most malicious of attacks.

Common Types of Threats and Attacks

- Unauthorized access – insecure hosts, cracking, intrusion
- Eavesdropping on a transmission – gaining access to the medium to look for passwords, credit card numbers, or business secrets

- Hijacking, or taking over a communication
- Inspecting and modifying any data being transmitted
- IP spoofing, or falsifying network addresses
- Impersonating to fool access control mechanisms
- Redirecting connections to a fake server
- Denial of Service (DoS) attacks, when a huge amount of useless data coming to the target makes it unable to function properly
- Interruption of service due to system destruction or using up all available system resources for the service CPU, memory, bandwidth
- Default passwords and default configuration

Mistakes People Make that Lead to Security Breaches

Technological holes account for a great number of successful break-ins, but people do their share as well.[11]

Five Worst Security Mistakes End-users Make

- Failing to install an anti-virus, keep its signatures up-to-date, and perform full system scans regularly.
- Opening unsolicited e-mail attachments without verifying their sources and checking their content first, or anything from untrusted sources.
- Failing to install security patches, especially for frequently used software like those for word processing, web browser, e-mail client and operating systems.
- Not making and testing backups.
- Using a modem while connected through a local area network.

Seven Worst Security Mistakes Senior Executives Make

- Assigning untrained people to maintain security and providing neither the training nor the time to make it possible to learn and do the job.
- Failing to understand the relationship between information security and business problems. They understand physical security but do not see the consequences of poor information security.
- Failing to deal with the operational aspects of security: making a few fixes and then not allowing the follow-through necessary to ensure that the problems stay fixed.
- Relying primarily on a firewall.
- Failing to realize how much money their information and organizational reputations are worth.
- Authorizing reactive, short-term fixes so that problems re-emerge rapidly.
- Pretending the problem will go away if they ignore it.

[11] Most of the points under these topics trace their origin back to the SANS institute. SANS Institute is located at http://www.sans.org. Text has been modified to suit changing times.

Ten Worst Security Mistakes IT People Make

- Connecting systems to the Internet before hardening them.
- Connecting test systems to the Internet with default accounts/passwords.
- Failing to update systems when security holes are found.
- Using telnet and other unencrypted protocols for managing systems, routers, firewalls and PKI.
- Giving users passwords over the phone or changing user passwords in response to telephone or personal requests when the requester is not authenticated.
- Failing to maintain and test backups.
- Running unnecessary services: ftpd, telnetd, finger, rpc, mail, rservices.
- Implementing firewalls with rules that do not stop malicious or dangerous traffic, whether incoming or outgoing.
- Failing to implement or update virus detection software.
- Failing to educate users on what to look for and what to do when they see a potential security problem.

Security Best Practices and To-dos

Some set a goal to fully and completely secure a system, which is impractical as it is usually impossible to make a system full-proof. A realistic goal is to set up a regular routine where you identify/correct as many vulnerabilities as possible.

Security Best Practices

A good security system makes it so difficult for an attacker to gain access that he gives up or gives an administrator enough time before he gets in.

- The idea is not that you should protect a system to the point that it cannot be compromised, but to secure it at least enough so that most intruders will not be able to break in, and will choose to direct their efforts elsewhere. This is just like putting iron bars and locks on our windows and doors. We do it not to 'keep the robbers out', but to persuade them to turn their attention to our neighbours.
- Rather than directing our efforts at protecting against the thousands of specific threats (this exploit, that Trojan virus, these mis-configurations), we focus our energies on tasks that provide the most comprehensive protection against the majority of threats.
- Best security practices are very dynamic, constantly changing and evolving.
- Administrators should include their own best security practices and modify those mentioned here to best fit their environment.

Points to Ponder

- Take into consideration your needs, risks, and resources.
- Information systems are unavoidably complex and fluid, so the most effective way to apply security is in layers.
- You should place security measures at different points in your network, allowing each to do what it does best.

- From an attacker's perspective, you will have constructed a series of obstacles of varying difficulty between the attacker and your systems.
- Secure each component in your system (firewalls, routers, servers, hosts and appliances) so that even if an attacker works his/her way through your obstacle course, at the end he/she will find systems that are resistant to attack.

Backup

- Maintain full and reliable backups of all data and log files.
- Archive all software (purchased or freeware), upgrades, and patches off-line so that they can be reloaded when necessary.
- Backup configurations, such as the Windows registry and text/binary configuration files, used by the operating systems or applications.
- Consider the media, retention requirements, storage, rotation, methods (incremental, differential, full) and scheduling.
- Keep a copy of a full backup in a secure off-site location for disaster recovery.

Security To-dos

- Enforce Deny by Default. Everything that is not allowed specifically is denied. This is a good way of tightening security.
- Reduce access to things on a 'need to know' basis. That means that anybody who does not need access to a computer, a network, or a piece of information should not have access to it. This is also known as the Least Privilege Rule. Give the least privilege, the least access capabilities you can for someone to do his/her job.
- Try to have identified choke points, which are points where you can inspect everything going on the network – i.e., a network where all communications pass through the Internet, a VPN concentrator where all the external accesses converge, a Modem pool where all external dial-in access come to, a dedicated VLAN where all the Wireless (WLAN) Access Points are connected to see who is coming from the wireless entry points.
- Try to identify and focus on your weakest link, or the weakest part of your security that would compromise all the rest. Remember that the hacker just needs to succeed once to hack you, whereas you have to succeed all the time in your security job to protect your organization. So, the weakest link is always where you should focus your attention, trying to have no security 'fence' notably 'lower' than all the others in your network.
- You should also try to have the capability to 'fall back' into a secure position in case of trouble. Security Policy Management systems can ensure the 'lock down' of the network in case of a worm, for example. Such a configuration would allow the DNS and maybe some vital traffic for the company (payroll, electronic fund transfer, ERP traffic) and block all the potential worm infection vectors (mail, web).
- You should try to have a security device 'deep down' in your network, not only at the boundaries with the external world. If the life of your company depends on one computer holding sensitive data or running a factory, add a firewall in front of it. This is 'Defense in Depth'.
- If you diversify your defense, such as running two different brands of antivirus (one kind for your mail gateway, and another kind for the desktops and user machines), that may help you catch some attacks that were not blocked by one brand but will be by the other brand. You can also have a diversity of kinds of defense, like having antivirus and personal firewalls on the desktops.
- KISS – Keep It Short and Simple. If your network and your security are simple, you will be less likely to have forgotten something that attackers will exploit. Clarity is important as you HAVE TO understand the whole network and the whole security organization. Do not feel that it is

'dumb' to ask 'simple questions'; these are often the ones that will make everybody realize that there is a problem.

- Security through obscurity is the opposite of the following principle. If someone tells you 'do not worry, this is secure' and does not want to explain the security mechanism, then YOU SHOULD WORRY. Hiding how things work and having your security depending on this to be secure is the worst thing you can have. For example, you should still secure computers in your network even if attackers cannot download the DNS zone of your company which lists the IP addresses. Sooner or later, an attacker will find a way to know about this 'hidden' computer and will then have full access. The only things you should keep secret and hidden are your password and your cryptographic keys (PGP private keys). Assume that all other knowledge may be known by attackers.[12]

Securing Your Network and Hosts Properly

Firewalls

Many people think that a firewall is a single device that is configured to protect your internal network from the external world. In reality, a firewall is a system (or a group of systems) that enforces an access control policy between two networks. It disallows unauthorized and/or malicious traffic from traveling into and out of your network. Users must realize that firewalls cannot protect you from attacks that do not go through it. If there is another entry point to your network that is not protected by a firewall, then your network is not secure. Also, contrary to the common misconception, firewalls do not verify the content of the traffic going through it.

Types of Firewalls

Packet filtering firewalls

- Examines the source and destination address of the data packet and either allows or denies the packet from traveling through the network.
- Blocks access through the firewall to any packets that try to access ports which have been declared 'off-limits'.

Application layer firewalls

- Attempts to hide the configuration of the network behind the firewall by acting on behalf of the network/servers.
- All requests for access are translated at the firewall so that all packets are sent to and from the firewall, rather than from the hosts behind the firewall.

Stateful inspection firewalls

- Examines the state and the context of the packets.
- Remembers what outgoing requests have been sent and allows only the responses to those requests back through the firewall.
- Attempts to access the internal network that have not been requested by the internal network will be denied.

Firewall Best Practices

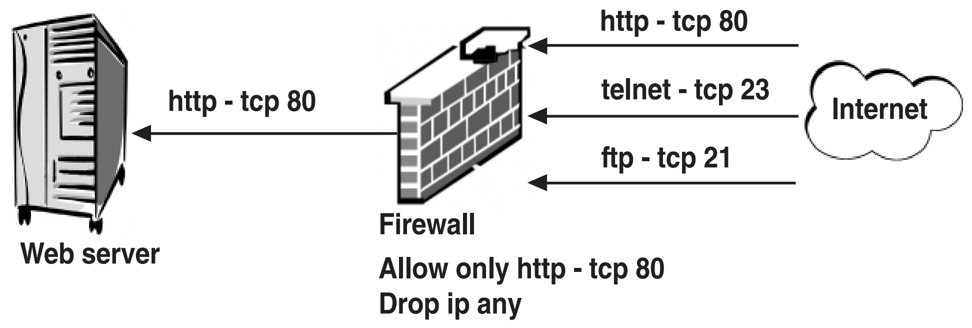
Regardless of which type of firewall, someone has to configure it to make it work properly. The rules for access must be defined and entered into the firewall for enforcement. A security manager is usually responsible for the firewall configuration. But again, if the policy is not made properly, then there is not

[12] Contributed by Philippe Langlois <philippelanglois@free.fr>

much that the security manager can do. Some rules of thumb are:

- Explicitly deny all traffic except for what you want.
- The default policy should be that if the firewall does not know what to do with the packet, it should deny/drop the data packet.
- Do not rely only on your firewall for the protection of your network. Remember that it is only a device, and devices do fail.
- Make sure you implement what is called 'defense in depth', or multiple layers of network protection.
- Make sure all of the network traffic passes through the firewall.
- If the firewall becomes disabled, then disable all communication.
- If there is another way into the network (like a modem pool or a maintenance network connection), then this connection could be used to enter the network, completely bypassing firewall protection.

A typical firewall setup



- Disable or uninstall any unnecessary services and software on the firewall.
- Limit the number of applications that run on the firewall.
- Consider running antivirus, content filtering, Virtual Private Network, etc. on different hardware.
- Do not rely on packet filtering alone. Use stateful inspection and application proxies if possible.
- Ensure that you are filtering packets for illegal/incorrect addresses, to avoid IP spoofing.
- Ensure that physical access to the firewall is controlled.
- Use firewalls internally to segment networks between different departments and permit access control based on business needs.
- Remember that firewalls will not prevent attacks that originate from inside your network.

Netfilter/iptables

iptables[13] is the most popular FOSS firewall tool available on GNU/Linux. It works in combination with netfilter, which is included as an integral part of the Linux kernel. It has been in the kernel as the firewalling subsystem since release 2.4.x.

[13] http://www.iptables.org

iptables is a much improved version of previously available IP Chains and ipfwadm software.

It does packet filtering (stateless or stateful) as well as Network Address Translation (NAT) and IP Masquerading. It can also perform packet mangling (manipulation).

iptables consists of the following:

- Userspace command for runtime configuration of the firewalling subsystem;
- Kernel modules and the user interface application. Kernel modules are components that handle various tasks and are compiled as loadable modules in the GNU/Linux kernel;
- Generic table structure for the definition of rulesets; and
- A number of classifiers (matches) and one connected action (target).

iptables – Tables , Chains and Rules

iptables normally has multiple tables defined in the kernel. The default is the 'filter' table. The Linux kernel starts with three lists of rules in the 'filter' table. A chain is a checklist of rules. There are three built-in chains: the INPUT, OUTPUT and FORWARD chains. The Linux kernel examines each chain to decide the fate of the packet. In turn, each rule says 'if the packet header looks like this, then here is what to do with the packet'.

iptables Syntax

```
[root@mail /]# iptables [options] [table] rules TARGET
```

Main iptables options

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

You can use 'iptables –h' to see a list of command options

To manipulate rules in a chain

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

The default policies

The default policy is to accept all packets to, from, and through it.

INPUT ACCEPT any any
FORWARD ACCEPT any any
OUTPUT ACCEPT any any

To set the default policies to DROP all packets:

```
# iptables –P INPUT DROP
# iptables –P FORWARD DROP
# iptables –P OUTPUT DROP
```

Targets – what to do if a packet matches a rule

ACCEPT
DROP
REJECT
LOG

Example rules

```
# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
# iptables -D INPUT 1
# iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP

# iptables -A INPUT -s 0/0 -j DROP

# iptables –A INPUT –p tcp –dport 25 –j ACCEPT
# iptables –A FORWARD –p tcp dport 25 –j ACCEPT
# iptables –A FORWARD –I eth0 –o eth1 –p tcp –dport 25 –j ACCEPT
# iptables –A FORWARD –s 192.0.2.0/24 –d 202.52.255.1 –p tcp –
dport 25 –j ACCEPT
# iptables –A FORWARD –s 192.0.2.0/24 –d 202.52.255.5 –p udp –
dport 53 –j ACCEPT
# iptables –A FORWARD –d 202.52.255.5 –p tcp –j LOG –log-prefix
"TCP log "
```

To get help:

```
# iptables -p tcp -h
# iptables -m state -h
# iptables –j ACCEPT -h
```

Connection tracking

Stateful connection tracking of traversing packets with –m state –state options

NEW – packets that create a NEW connection
ESTABLISHED – packets belonging to an existing connection – reply packets
RELATED – packets related to an existing connection – ICMP error messages/ FTP data connections
INVALID – packets not corresponding to any existing connection

```
# iptables –A FORWARD –d 192.0.2.0/24 –p tcp –m –state –state
ESTABLISHED –j REJECT
# iptables –A FORWARD –m –state –state INVALID –j DROP
```

Sample iptables setup

```
iptables –F flush all the chains
iptables –X delete all the empty chains

# Changing Default policy
iptables –P INPUT DROP
iptables –P FORWARD DROP
iptables –P OUTPUT DROP
# INPUT chain
iptables –A INPUT –p icmp –j ACCEPT
iptables –A INPUT –I eth1 –p tcp –dport 22 –j ACCEPT
iptables –A INPUT –I eth1 –m state –state ESTABLISHED,RELATED –j
ACCEPT
```

```
# FORWARD chain
iptables –A FORWARD –p icmp –j ACCEPT
iptables –A FORWARD –I eth0 –o eth1 –d 192.0.2.1 –p tcp –dport 25
–j ACCEPT
iptables –A FORWARD –I eth0 –o eth1 –d 192.0.2.2 –p tcp –dport
110 –j ACCEPT
iptables –A FORWARD –I eth0 –o eth1 –d 192.0.2.3 –p tcp –dport 80
–j ACCEPT
iptables –A FORWARD –I eth0 –o eth1 –d 192.0.2.4 –p tcp –dport 53
–j ACCEPT
iptables –A FORWARD –I eth0 –o eth1 –d 192.0.2.4 –p udp –dport 53
–j ACCEPT
iptables –A FORWARD –I eth0 –o eth1 –d 192.0.2.0/24 –m state –
state ESTABLISHED,RELATED –j ACCEPT
iptables –A FORWARD –I eth1 –o eth0 –s 192.0.2.1 –p tcp –dport 25
–j ACCEPT
iptables –A FORWARD –I eth1 –o eth0 –s 192.0.2.4 –p tcp –dport 53
–j ACCEPT
iptables –A FORWARD –I eth1 –o eth0 –s 192.0.2.4 –p udp –dport 53
–j ACCEPT
iptables –A FORWARD –I eth1 –o eth0 –s 192.0.2.0/24 –p tcp –dport
80 –j ACCEPT
iptables –A FORWARD –I eth1 –o eth0 –s 192.0.2.0/24 –m state –
state ESTABLISHED,RELATED –j ACCEPT
# OUTPUT chain
iptables –A OUTPUT –p icmp –j ACCEPT
iptables –A OUTPUT –o eth1 –p tcp –dport 22 –j ACCEPT
iptables –A INPUT –o eth1 –m state –state ESTABLISHED,RELATED –j
ACCEPT
# Enabling NAT
iptables -t nat -A POSTROUTING -s 192.0.2.0/24 -j MASQUERADE
```

References

http://www.unixreview.com/documents/s=1237/urm0103c/0103c.htm
http://www.knowplace.org/netfilter/index.html
http://www.GNU/Linuxguruz.org/iptables/
http://www.boingworld.com/workshops/GNU/Linux/iptables-tutorial/iptables-tutorial/iptables-tutorial.html

Server Security, Host/Network Scanning and Intrusion Detection System

It is often advisable that a detection and prevention measure be run in addition to the firewall. This is commonly referred to as Network Scanning and Intrusion Detection System (IDS). There are also other proactive measures that can make your system and servers secure. Let's look at some of these.

Server Security Dos and Don'ts

- Run the server on a hardened and routinely updated operating system.
- Keep current on software/application updates.
- Ensure that you test these updates in a controlled, non-production environment whenever possible.
- One server patch may undo a correction applied by a previous patch so scan the server after the patching-up.
- Disable file sharing on all critical machines as it makes them vulnerable to information theft and certain types of quick-moving viruses.

- Improper sharing configurations can expose critical system files or give full file system access to any hostile party.

Host/Network Scanning

Network scanning refers to the activity of scanning your own networks for vulnerabilities. There are tools available for this purpose. Incidentally, these are most often the same tools used by crackers and external parties to gauge your network security. Thus, using them yourself can lead you to quite a few of those security holes, which would go unnoticed otherwise. Below are some tips on what to scan for.

Scanning systems tips

- Scans will help determine that only the required ports are open.
- Check that services running on the open ports are not vulnerable to known security bugs/holes.
- If new open ports are found, check if your systems have been compromised.
- Perform full port scans using a tool like nmap/ndiff, nessus, or fscan on a regular basis.
- Port scans should cover all ports (1-65,535), both UDP and TCP, on all systems: both client and server devices such as routers, switches, printers, and anything else connected (physically through wire or wireless) to your network.

Nmap

Nmap[14] is a Network MAPper. It is a powerful utility for network exploration or security auditing, which can rapidly scan large networks or single hosts to determine the hosts available on the network, the services they (ports) are offering, the operating system (and OS version) they are running, and the type of packet filters/firewalls in use. It runs on most types of computers, and both console and graphical versions are available. It is also a GNU GPL software, available with full source code.

Here are examples of Nmap usage:

1. This option scans all reserved TCP ports on the machine. The -v means turn on verbose mode.

```
nmap -v target.example.com
```

2. This launches a stealth SYN[15] scan against each machine that is up out of the 255 machines on class ´C´ where target.example.com resides. It also tries to determine what operating system is running on each host that is up and running. This requires root privileges because of the SYN scan and the OS detection.

```
nmap -sS -O target.example.com/24
```

3. This sends a Xmas tree scan to the first half of each of the 255 possible 8-bit subnets in the 198.116 class ´B´ address space. We are testing whether the systems run sshd, DNS, pop3d, mapd, or port 4564. Note t hat Xmas scan does not work on Microsoft boxes due to their reduced TCP stack. The same goes with CISCO, IRIX, HP/UX, and BSD boxes.

```
nmap -sX -p 22,53,110,143,4564 198.116.*.1-127
```

4. Rather than focus on a specific IP range, it is sometimes interesting to slice up the entire Internet and scan a small sample from each slice. This command finds all web servers on machines with IP addresses ending in .2.3, .2.4, or .2.5.

```
nmap -v —randomize_hosts -p 80 ´*.*.2.3-5´
```

[14] http://www.insecure.org/nmap

[15] According to Wikipedia, SYN scan is the most popular form of TCP scanning. Rather than use the operating system's network functions, the port scanner generates raw IP packets itself, and monitors for responses. This scan type is also known as 'half-open scanning', because it never actually opens a full TCP connection.

5. Launch a stealth scan with OS detection on all privileged ports against 255 hosts in the network, and then output the results into the file /root/nmap.scan.

```
    nmap –sS –O 192.0.2.0/24 –oN /root/nmap.scan
```

6. Launch a stealth scan with OS detection on specified ports against 255 hosts in the network, in verbose mode.

```
    nmap –sS –O –v 192.0.2.0/24 –p '1-1024,1080,3128'
```

Nessus

The mantra for network monitoring and scanning could be: 'Prevention is ideal, but detection is a must'. We must realize that the no prevention technique is fool-proof and new vulnerabilities are discovered every week that you may not be aware of. Thus, constant vigilance is required to detect new unknown attacks. Once you are attacked, without logs, you have little chance of finding what the attackers did. It is also common sense that you cannot detect an attack if you do not know what is occurring on your network. Logs provide the details of what is occurring, what systems are being attacked, and what systems have been compromised. If any log entry does not look right, you should investigate them immediately. But thousands of lines of logs are produced every day. It is necessary to use the appropriate tools to diagnose these. It is also necessary for these logs and scans to be compared against vulnerability databases, so that the security gap in the network can be known immediately.

Nessus

Nessus[16] is a security scanner that can remotely audit a given network or server. It can determine whether 'crackers' may break into it, or misuse your network or servers in some way.

Unlike others, Nessus does not take anything for granted and will not assume that services run only on fixed ports. For example, if you run your web server on port 1234, Nessus will detect it and test its security. It will not make its security tests by the version number, but will really attempt to exploit the vulnerability. It is very fast and reliable, and has a modular architecture that allows you to fit it to your needs. Nessus also has front ends for both Windows and Macintosh.

Nessus Installation

Nessus is made up of two parts: a client and a server. You need a UNIX -like system to use the server. In this test, the standard client Nessus is used, mainly because it supports the cipher layer. The Nessus Security Scanner relies on the following items:

- GTK - The Gimp Tool Kit, version 1.2

    GTK is a set of Widgets (like Motif) which is used by many FOSS applications such as The Gimp. GTK is used by the POSIX client Nessus.

    Download it at ftp://ftp.gimp.org/pub/gtk/v1.2

    Note #1: If your system comes with GTK, make sure that you have the gtk-config program installed. If you do not, install the gtk-devel package that should come on your distribution CD-ROM.

    Note #2: If you do not want to install GTK and/or if your system lacks X11, then you can compile a command-line client by doing .

    ```
    /configure –disable-gtk
    ```

    in nessus-core

- Nmap, which we already covered earlier, is an excellent port scanner. It is available at http://www.insecure.org/nmap/.

[16] http://www.nessus.org

- OpenSSL (optional but heavily recommended) is used for the client-server communication as well as in the testing of SSL-enabled services. Get it at http://www.openssl.org.

Nessus also comes as a standalone package that auto-installs itself. To use it, download the script nessus-installer.sh at http://www.nessus.org/download.html.

Using Nessus

Create a nessusd account

The nessusd server has its own user database, with each user having a set of restrictions. This allows you to share a single nessusd server for a whole network and different administrators who will only test their part of the network.

The utility nessus-adduser takes care of the creation of a new account:

```
# nessus-adduser
Using /var/tmp as a temporary file holder

Add a new nessusd user
----------------------

Login: sunil
Authentication (pass/cert) [pass]:
Login password: nessus
User rules
----------

nessusd has a rules system which allows you to restrict the hosts that Sunil has the right to test. For instance, you may want him to be able to scan his own host only.

Please see the nessus-adduser man page for the rules syntax. Enter the rules for this user, and hit ctrl-D once you are done: (the user can have an empty rules set)

Login: sunil
Password: nessus
DN:
Rules:

Is that ok? (y/n) [y]
User added.
```

Configure your Nessus daemon

In the file /usr/local/etc/nessus/nessusd.conf, you can set several options for nessusd. Typically, this is where you can define that you want nessusd to use your favourite language. The standard configuration file has been kept for this illustration.

Start nessusd

Once all of this is done, you can safely start nessusd as root:

```
nessusd -D
```

Fire up Nessus in X windows graphical environment

Click on Login, since this setup is correct. Since this is the first time you are connecting to this server, it will ask for the password. The next time you connect to it, the public key will be enough.

Once connected, the Log-in button changes to Log-out, and a Connected label appears at its left.

The security checks configuration

Let all the security checks be performed, except the DoS attacks, because you do not want hosts to crash.

Clicking on a plugin name will pop up a window explaining what the plugin does.

The plugins preferences

Some security checks will require extra arguments. For instance, the pop2 overflow security test needs a valid pop account. The plugin, which tests whether an FTP directory is writeable or not, asks if it should just trust the permissions or really attempt to store a file.

The scan options

Here you choose which port scanner you want to use. Opt for the Nmap tcp connect scanner, since it is the fastest.

Define the targets

Uncheck the 'Perform a DNS transfer zone' option, since it would make DNS transfer on fr.nessus.org and nessus.org, and it would be useless, since it would not gain any new hosts.

Use the following options to define the targets:

```
192.0.2.1 A single IP address.
192.0.2.1-7 A range of IP addresses.
192.0.2.1-192.0.3.50 Another range of IP addresses.
192.0.2.1/29 Again a range of IP addresses in CIDR[17]
prof.fr.nessus.org A hostname in FQDN[18] notation.
prof A hostname (as long as it is resolvable on the server).

prof, 192.0.2.1/29, ... Any combination of the above mentioned
forms separated by a comma.
```

The rules section

The rules allow a user to restrict his test. For instance, if you want to test 192.0.2.0/29, except 192.0.2.2. The rule set entered allows you to do that. Once all of this is done, start the scan.

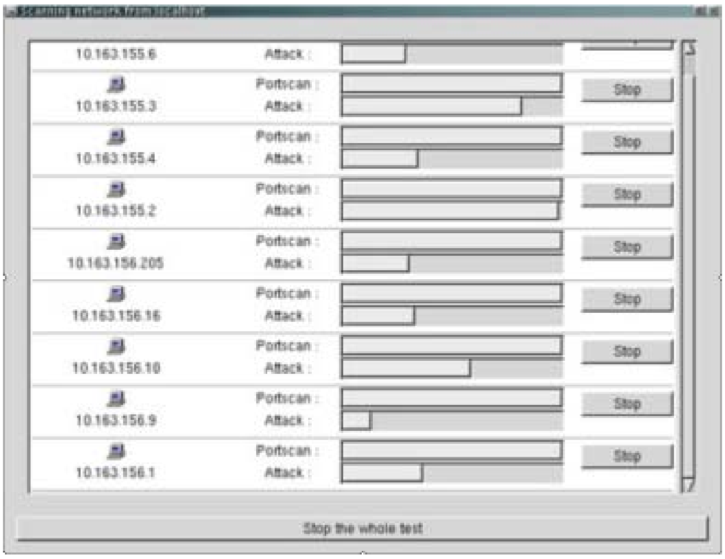
The Nessus scan report

Now that the scan is over, the report window just pops up.

Resolving the security issues

Go through the scan report and try to resolve reported security holes on the respective servers by following the suggestions given in the report OR by referring to the vendor's site.

Nessus screenshot



Scanning network from localhost

10.163.155.6 Attack :
10.163.155.3 Portscan : Attack : Stop
10.163.155.4 Portscan : Attack : Stop
10.163.155.2 Portscan : Attack : Stop
10.163.156.205 Portscan : Attack : Stop
10.163.156.16 Portscan : Attack : Stop
10.163.156.10 Portscan : Attack : Stop
10.163.156.9 Portscan : Attack : Stop
10.163.156.1 Portscan : Attack : Stop

Stop the whole test

[17] Classless Inter Domain Routing.
[18] Fully Qualified Domain Name.

Intrusion Detection System (IDS)

Intrusion detection is the art of detecting inappropriate, incorrect, or anomalous activity. IDS inspects/sniffs all network traffic passing through it for any abnormal content. It has a built-in signature-base and anomaly detection, which provides the capability to look for set 'patterns' in packets. It can also string search signature (i.e., look for confidential information), log and TCP reset features. IDS provides worthwhile information about malicious network traffic and helps identify the source of incoming probes, scans, or attacks. It is similar to a security camera or a burglar alarm as it alerts security personnel that someone is picking the 'lock' – i.e., it alerts security personnel that a network invasion may be in progress.

SNORT

The most common FOSS IDS in use is SNORT.[19] It does the following:

- Monitors network traffic for predefined suspicious activities or patterns
- Alerts system administrators when potential hostile traffic is detected
- Serves as a cross-platform, lightweight network intrusion detection tool
- Can be deployed to monitor small TCP/IP networks
- Detects a wide variety of suspicious network traffic as well as outright attacks
- Deployed rapidly to fill potential holes in a network's security coverage

Three Modes of SNORT

Sniffer mode

Displays the packet headers or data

```
snort –v
snort –vd
```

Packet logger mode

Records packets to the disk

```
snort –dev –l ./log
snort -l ./log –b
```
log all packets in binary mode

```
snort –dvr ./log/snort.log
```
to display packets in ascii mode from the binary log

Network intrusion detection mode

Does not record every single packet sent down the wire

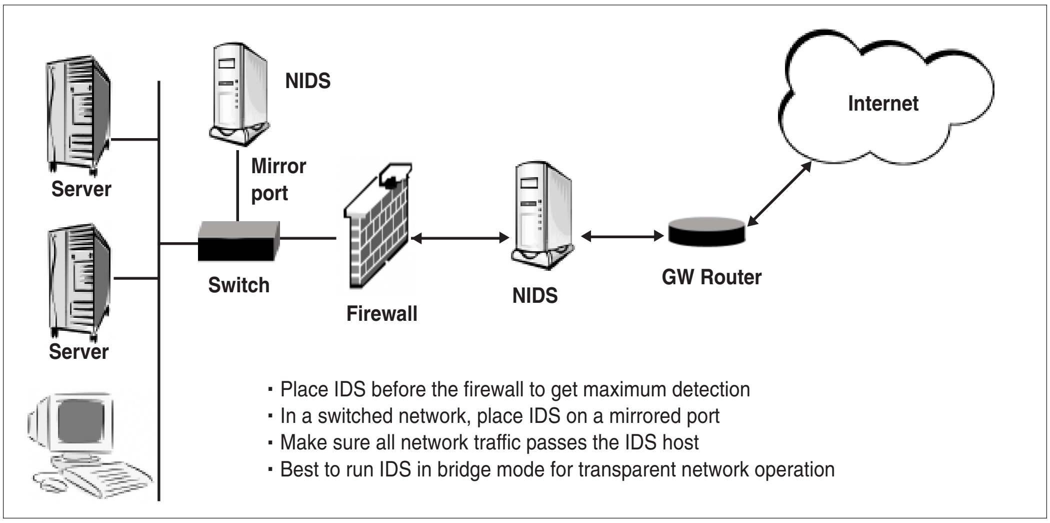
```
snort -d -h 192.0.2.0/24 -l ./log -c snort.conf
```

This will configure SNORT to run in its most basic NIDS mode, logging packets that the rules tell it to in plain ASCII to a hierarchical directory structure (just like the packet logger mode).

To send NIDS alerts to syslog use -s:

```
snort -s -h 192.0.2.0/24 –l ./log -c snort.conf
```

[19] http://www.snort.org


NIDS
Internet
Mirror port
Server
Server
Switch
Firewall
NIDS
GW Router
· Place IDS before the firewall to get maximum detection
· In a switched network, place IDS on a mirrored port
· Make sure all network traffic passes the IDS host
· Best to run IDS in bridge mode for transparent network operation

High Performance Configuration

```
snort -b -A fast -c snort.conf
```

This logs packets in tcpdump format and produces minimal alerts.

To read this file back and break out the data in the familiar SNORT format:

```
snort -d -c snort.conf -l ./log -h 192.0.2.0/24 -r snort.log
```

To run SNORT in daemon mode (provide full paths):

```
usr/local/bin/snort -d -h 192.0.2.0/24 -l /var/log/snortlogs -c /
usr/local/etc/snort.conf -s -D
```

To post packet logs to public mailing lists use –O to obfuscate your IP addresses

```
snort -d -v -r snort.log -O -h 192.0.2.0/24
```

To get help options:

```
snort –h
```

References

http://www.snort.org/docs/writing_rules/
http://www.snort.org/docs/lisapaper.txt

OpenSSH

One of the biggest advantages of UNIX and GNU/Linux is the ability to work remotely on the server systems. This feature is an inherent part of any UNIX-like operating system. In earlier days, most people used Telnet to log remotely into servers but now Telnet has been made obsolete by **ssh** or secure shell.

The main difference between Telnet and ssh is the secure communication protocol of the latter. In Telnet all of the communication was done in plain text, whereas in ssh all communication is encrypted.

OpenSSH[20] is both an ssh server and client. It is installed by default on most GNU/Linux distributions. Since it even encrypts the password sent to set up the session, it effectively eliminates eavesdropping, connection hijacking, and other network-level attacks. It includes the following different server sub-systems:

[20] http://www.openssh.org

sshd – ssh daemon run on the server machine
sftp-server – sftp server sub-system
ssh-agent – authentication agent that holds the keys
ssh-add – used to register new keys with the Agent
ssh-keygen – used to create public authentication keys

OpenSSH Config Files

Since OpenSSH is both a client and a server, there are two configuration files for each purpose:

/etc/ssh/ssh_config – ssh client systemwide configuration file, provides defaults for users
/etc/ssh/sshd_config - sshd server system-wide configuration file

OpenSSH Server Configuration

Installation and basic operation

Install the openssh-server and openssh rpm package included in your Linux distribution. Most GNU/Linux distributions will have the option to install OpenSSH. Please note that the openssh-server package depends on the openssh package. OpenSSH packages also require the OpenSSL package (openssl) which installs several important cryptographic libraries that help OpenSSH provide encrypted communications.

OpenSSH daemon, sshd, uses the configuration file '/etc/ssh/sshd_config'

The default config file is sufficient. For customization, refer to the sshd main page for config options.

Editing/viewing the sshd_config file

[root@mail /] # vi /etc/ssh/sshd_config

```
#Port 22 - Specifies the port number that sshd listens on
#Protocol 2,1 - Specifies the protocol versions sshd supports
#ListenAddress 0.0.0.0 - Specifies the local addresses sshd
should listen on
#HostKey /etc/ssh/ssh_host_rsa_key - Specifies a file containing
a private host key used by SSH
#SyslogFacility AUTH - Gives the facility code that is used when
sshd logs messages
#LogLevel INFO - Gives the verbosity level that is used when sshd
logs messages
#LoginGraceTime 600 – time limit for a user to log in
#PermitRootLogin yes - Specifies whether the root can login using
ssh
#StrictModes yes - Specifies whether sshd should check file modes
and ownership of the user's files and home directory before
accepting login
#PubkeyAuthentication yes - Specifies whether public key
authentication is allowed
#PasswordAuthentication yes - Specifies whether password
authentication is allowed
#PermitEmptyPasswords no - When password authentication is
allowed, it specifies whether the server allows login to accounts
with empty password strings
#MaxStartups 10 - Specifies the maximum number of concurrent
unauthenticated connections to the sshd daemon
#KeepAlive yes - Specifies whether the system should send TCP
keepalive messages to the other side
#VerifyReverseMapping no - Specifies whether sshd should try to
verify the remote host name
Subsystem sftp /usr/libexec/opensshsftp-server
```

Managing the sshd service

To start the service: `#/etc/rc.d/init.d/ sshd start`
To stop the service: `#/etc/rc.d/init.d/ sshd stop`
To restart the service: `#/etc/rc.d/init.d/ sshd restart`
To automatically run the service: `# chkconfig sshd on`

OpenSSH Client Configuration

To connect to an OpenSSH server from a client machine, you must have the openssh-clients and openssh packages installed on the client machine.

System-wide ssh client configuration is stored in **/etc/ssh_config** and is used as the default for all system-wide users.

The configuration values can be changed in per user configuration files or on the command line. The default home directory is **~/.ssh**

Using the ssh with password authentication

1. Make sure you have the following enabled in **/etc/sshd_config** file on the ssh server

```
PasswordAuthentication yes
```

2. To log in to a host named server.com, type the following:

```
$ ssh gaurab@server.com
```

The first time you ssh to a remote machine, you will see a message similar to the following:

```
The authenticity of host ‘server.com (192.0.2.22)’ can’t be
established.
RSA key fingerprint is
0f:41:6c:52:31:59:43:0f:dd:49:5f:3d:47:9d:b5:9e.
Are you sure you want to continue connecting (yes/no)? yes
```

Type **yes** to continue.
This will add the server to your list of known hosts as seen in the following message:

```
Warning: Permanently added ‘server.com,192.0.2.22’ (RSA) to the
list of knownhosts.
gaurab@server.com’s password:
Last login: Sat Oct 16 19:34:13 2004 from gw-ktm.lahai.com
[gaurab@server.com gaurab]$
```

3. **known_hosts** file in **.ssh** folder contains the remote servers’ host keys:

```
$ more known_hosts
server.com,192.0.2.22 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA5IwKqHiqSeXjrX3dCpS1gXZo9GqJ0hkk+clf+WmABYbEH6IGMyy2CeA
RtaR6QLpqB1SaGEFsn84dqA6kWLYfn4FuDVDc8KTyABLVEMOm6NnLZkHPKr3Cb0RgivDYSYHlwgWu
Di7XBvmoC44WA2EbM7eBy5h1kHrXZ5yPXq3rxI0=
```

Using the ssh command with public key authentication

1. Make sure you have the following enabled in /etc/sshd_config file on the ssh server.

   PubkeyAuthentication yes

2. Key pair generation:
   ssh-keygen - authentication key generation, management and conversion

To generate a RSA key pair to work with version 2 of the protocol:

```
$ ssh-keygen -t rsa

Generating public/private rsa key pair.
Enter the file in which to save the key (/home/gaurab/.ssh/
id_rsa):

Enter a passphrase that is different from your account password
and confirm it by entering it again:
```

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

Your identification has been saved in /home/gaurab/.ssh/id_rsa.
Your public key has been saved in /home/gaurab/.ssh/id_rsa.pub.
The key fingerprint is:
e5:a1:ac:ce:8d:16:3a:b9:3a:e4:e9:06:9c:90:b6:da gaurab@lahai.com

The public key is written to ~/.ssh/id_rsa.pub.
The private key is written to ~/.ssh/id_rsa.
Never distribute your private key to anyone and change the
permissions of your .ssh directory using the command chmod 755 ~/
.ssh

$ ll .ssh/
total 8
-rw——— 1 gaurab gaurab 951 Oct 16 19:37 id_rsa
-rw-r—r— 1 gaurab gaurab 236 Oct 16 19:37 id_rsa.pub
```

Copy the contents of **~/.ssh/id_rsa.pub** to **~/.ssh/authorized_keys** on the machine to which you want to connect.

If the file **~/.ssh/authorized_keys** does not exist, you can copy the file **~/.ssh/id_rsa.pub** to the file **~/.ssh/authorized_keys** on the remote SSH server.

Securely copy your public key file to the remote ssh server:

```
$ scp ~/.ssh/id_rsa.pub gaurab@server.com:
```

Logon to the remote ssh server using password authentication, one more time:

```
$ ssh gaurab@server.com
```

Copy the content of the public key file to authorized_keys file on the remote host:

```
$ cat id_rsa.pub >> ~/.ssh/authorized_keys
```

Now, logon to the remote ssh server via public key authentication:

```
$ ssh gaurab@server.com

Enter passphrase for key '/home/gaurab/.ssh/id_rsa':
Last login: Sat Oct 16 19:40:11 2004 from myhost.com
[gaurab@server.com gaurab]$
```

To run a command on the remote host with ssh and exit:

```
$ ssh ns.lahai.com ls /usr/share/doc
```

Using the scp command

The scp command is used to transfer files between machines over a secure, encrypted connection.

To transfer the local file shadowman to /home/username/shadowman on penguin.example.net:

```
$ scp shadowman username@penguin.example.net:/home/username
```

To transfer a remote file to the local system:

```
$ scp username@tohostname:/remotefile /newlocalfile
```

To transfer the contents of the directory /downloads to an existing directory called "uploads" on the remote machine penguin.example.net:

```
$ scp /downloads/* username@penguin.example.net:/uploads/
```

Using the sftp command

The sftp command is used to open a secure, interactive FTP session via an encrypted channel.

```
$ sftp username@hostname.com
```

Once authenticated, you can use a set of commands similar to using FTP.

NETWORK PLANNING

A network in any computing environment is always a long-term investment. It is imperative that proper planning be done before going all out to deploy a network. A few pointers on network design and development are given below.

Network Planning Basics

Capacity: Plan for at least two to three years
Infrastructure: Build for at least one year
Business Models: Look after the next quarter

As illustrated above, there are three factors to consider when planning a network. The first is capacity, both in terms of bandwidth as well as human resource. Second, you need to consider the infrastructure that you need to build to support the capacity. Ultimately, your network is good only for as long as it can help you meet business needs and costs. For both service providers and non-profit organizations, planning should be for at least two to three years. However, at the same time, the infrastructure should be able to handle at least another year of operation.

Major Considerations

Here are some more points to consider:

- Identify the components of a LAN/WAN and determine the type of network design that is most appropriate for a given site.
- Identify the different media used in network communications, distinguish between them, and determine how to use them to connect servers and workstations in a network.
- Differentiate between the different networking standards, protocols, and access methods, and determine which would be most appropriate for a given situation.
- Recognize the primary network architectures, identify their major characteristics, and determine which would be most appropriate for a proposed system.
- Identify the primary functions of network operating systems and distinguish between a centralized computing environment and a client/server environment.
- Determine how to implement and support the major networking components (including the server, operating system and clients), and propose a system for adequately securing data on a given LAN and protecting the system's components.
- Distinguish between LANs and WANs, identify the components used to expand a LAN into a WAN, and determine how to implement an appropriate modem in the larger LAN/WAN environment.
- Identify strategic LAN support tools and resources, and determine how to use these in troubleshooting basic network problems.

Services Planning

Mail

Choose a reliable MTA, but at the same time be cautious about spam, as it is a big headache.
Make redundant servers.
Separate user servers from real servers.

DNS

Use the latest BIND releases.
Plan nomenclature properly, but do not make it too obvious.
Redundancy is most important.
Arrange to host alternative DNS servers at off site/multiple locations.

User Services

E-mail access – POP, Web.
Web access.
Transparent caching/proxy.

Core Services – Infrastructure

Multi home: try to buy bandwidth from an IX facility.
Buying capacity is cheaper than managed capacity.
Plan to peer with other ISPs as much as possible.

Routers

Use loopback address in a separate subnet.
Use the lookback address as RouterID.
For core routers, memory is important.
For edge routers, ports are important.
Take configuration backups regularly.

Switching

Switching capacity is never enough. Invest in large switches, if that is what you will need in future.
Use VLANs to separate different groups of machines/networks.
For backbones, gigabit Ethernet is now more commonly used.

Backbone

Switched vs. routed backbone.
The same decisions apply as in LAN connections.
The backbone itself can be switched, and the traffic between different subnets can be routed.
Switching has its advantages if there is large local broadband use.

Branch Offices

Branch offices (BOs) need to be planned well in advance.
BOs tend to grow faster than you think they will.

Basic considerations for BOs.
Multihoming.
Distributed user services.
Authentication/remote management.

Hosts

Core services.
Use separate servers for separate functions.
1U servers are more manageable and also consume less power and space.
Use standardized platform as much as possible.

Using FOSS

In an integrated environment, FOSS tools are used alongside proprietary software and tools. This is a common scenario, but when it comes to network infrastructure, resources and security, FOSS provides an established and proven track record as the best software choice available in this area. In a networked environment, the ability to quickly diagnose and solve problems is critical. Experienced network administrators know that 80 percent of all network-related problems have to do with cabling and physical problems. Many problems can be minimized by using the best software in each category and, today, as argued above, the best software in most cases are FOSS.

FURTHER REFERENCES

The most useful resource for further reading is the Internet. Here are some useful sites on the Internet.

Resources on the Web

General reading/references

a. Free Software Foundation <http://www.fsf.org>
b. IOSN <http://www.iosn.net>
c. Linux Documentation Project <http://www.ldp.org>, <http://www.linuxdoc.org>
d. Internet Standards and RFCs <http://www.faqs.org/rfcs>
e. Internet Society / NSRC Workshop Resources <http://www.ws.isoc.org>
f. Network Startup Resource Center <http://www.nsrc.org>
g. UNESCO Free Software Portal <http://www.unesco.org/webworld/portal_freesoft/index.shtml>
h. Networking and Information Technology Observatory <http://www.sdnp.undp.org/observatory> - click on the "open source" theme

Linux software download and documentation

a. Source Forge <http://www.sourceforge.net>
b. Kernel .org <http://www.kernel.org>
c. Freshmeat <http://www.freshmeat.net>
d. RPMFind <http://www.rpmfind.net>
e. DNS/BIND <http://www.isc.org/bind>
f. Sendmail <http://www.sendmail.org>
g. SSL <http://www.openssl.org>
h. SSH <http://www.openssh.org>

Security-related sites

a. Security Focus <http://www.securityfocus.com>
b. CERT <http://www.cert.org>

Forums

a. Slashdot <http://www.slashdot.org>
b. Linux Online <http://www..linux.com>

Events

a. APRICOT <http://www.apricot.net>
b. NANOG <http://www.nanog.org>
c. AfNOG <http://www.afnog.org>
d. SANOG <http://www.sanog.org>
e. O'Reilly Open Source Conference <http://www.oreilly.com>

Useful mailing lists

a. ISP-Linux <http://www.isp-linux.com>
b. NANOG <http://www.nanog.org>
c. AfNOG <http://www.afnog.org>
d. SANOG <http://www.sanog.org>
e. NSP Security Mailing List <http://puck.nether.net/nsp-sec>

List of FOSS most commonly used by ISPs

Operating system

Debian Linux <http://www.debian.org>
FreeBSD <http://www.freebsd.org>

Accounting authorization authentication

Cistron Radius <http://www.radius.cistron.nl>
FreeRadius <http://www.freeradius.org>
Freeside <http://www.sisd.com/freeside>

Domain Name Server

BIND <http://www.isc.org/products/BIND>

MAIL servers and related software

Sendmail <http://www.sendmail.org>
Exim <http://www.exim.org>
CourierImap <http://www.inter7.com/courierimap>
SpamAssassin <http://www.spamassassin.org>
Majordomo <http://www.greatcircle.com/majordomo>

Web -related software

Apache <http://www.apache.org>
PHP <http://www.php.net>

Proxy and caching servers

Squid <http://www.squid-cache.org>

Security-related

Iptables <http://www.netfilter.org>
Snort <http://www.snort.org>
OpenSSH <http://www.openssh.org>
OpenSSL <http://www.openssl.org>

Network management system

MRTG <http://www.mrtg.org>
Netsaint <http://www.netsaint.org>
Nmap <http://www.nmap.org>
Ntop <http://www.ntop.org>

Browsers and e-mail clients

Firefox Internet Browser <http://www.firefox.com>
Mozilla <http://www.mozilla.org>
Thunderbird E-mail Client <http://www.firefox.com>

Books

Linux Books tends to get obsolete very quickly. But they are still a handy reference, when your access to the Internet is limited.

O'Reilly Books

Practical Internet and UNIX Security
Linux in a Nut Shell
Building Secure Servers with Linux
DNS and BIND
Essential System Administration
Linux Security Cookbook
Network Troubleshooting Tools

Other O'Reilly books on FOSS are also highly recommended.

Other Books

Unix Unleashed. System Administrator's Edition. India: Techmedia.
Special Edition - Using Linux. 3rd Edition. India: Prentice Hall of India Ltd.
Tanenbum, A. *Computer Networks*. Prentice Hall.

Free BSD References

The FreeBSD handbook is a full guide on FreeBSD.
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-communication.html
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/advanced-networking.html

GLOSSARY

Client	A client is a computer on the network which uses the server to get resources which may be e-mails, documents, or other forms of data.
Daemon	In UNIX and GNU/Linux, daemons are server software that constantly run on top of the operating system.
Host	Any device connected to the Internet or a TCP/IP network is referred to as the host.
IETF	The Internet Engineering Task Force is the Internet standards making organization. IETF publishes the RFC series documents as Internet standards. See http://www.ietf.org.
Internet	The Internet is commonly referred to as the 'networks of networks'. In technical terms, the Internet is the global network of computers that are able to communicate in TCP/IP with each other.
Intranet	A private network, which operates on the same principles as the Internet, is called Intranet. Intranets also use the TCP/IP protocol.
Localhost	Computers with TCP/IP installed refer to their loopback interface as the local host. This is defined by default in the DNS software.
Loopback	Loopback is a virtual interface defined in all TCP/IP computers. This virtual interface is assigned the IP address 127.0.0.1 and is required for TCP/IP to work properly.
PoP	Point of Presence – an Internet service provider's branch office. This should not be confused with POP3, which is an e-mail protocol and stands for Post Office Protocol.
Port	The points of contacts with any TCP/IP host. TCP/IP computers with IP addresses have ports ranging from 1 to 65336, which act as points of contact for different services running on that computer. For example, the SMTP protocol uses port 25, HTTP uses port 80, and so on.
RFC	Request for Comments is the series of documentation for Internet standards documents.
Server	A server is a powerful computer that serves resources to other computers on the network. The server can also be a big storage space as well as a database. The key here is the special software that enables the hardware to provide services.
SSL	Secure Sockets Layer is a secure communications layer that makes it possible to use the Web for e-commerce transactions. It makes sure that all communications between a client and server is encrypted.
VLAN	A virtual LAN is a feature provided by many Ethernet switches, to allow the creation of multiple logical networks over the same physical connections. A single switch with three VLANs behaves the same way as three different physical switches.
VPN	Virtual Private Networks are encrypted private networks which are running on top of the public network like the Internet.

ABOUT THE AUTHOR

Gaurab Raj Upadhaya is currently employed as Internet Economics Analyst/Staff Engineer at Packet Clearing House (http://www.pch.net), a research non-profit based in Berkeley, California. He works mostly in Internet backbone operations, analyzing peering relationships between operators and roles of Internet Exchange Points in different parts of Asia. Much of the work involves training ISPs in developing countries about best practices on network operations. He also runs the PCH INOC-DBA (http://www.pch.net/inoc-dba) hotline phone system for service providers. He initiated the Nepal Internet Exchange and currently serves as its voluntary CEO.

In 2003, Gaurab started the South Asian Network Operators Group (SANOG), a non-profit educational event and forum for ISPs in the South Asian Region (http://www.sanog.org). SANOG has a track dedicated to in-depth hands-on workshops on ISP operations using FOSS, along side other workshops on BIND/DNS and Security. He currently chairs SANOG. He is also the vice-chair of Asia Pacific Internet Association (http://www.apia.org), and Management Committee member for Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT) (http://www.apricot.net).

In the past he has served as a National UN Volunteer, working for the UNDP/Cisco/USAID Least Developing Countries Initiative (http://www.cisco.com/edu/ldci) as Unites volunteer (http://www.unites.org). He has been using Linux and FOSS since 1996. He previously worked as system and network administrator for United Mission to Nepal (http://www.umn.org.np).

His personal website is http://www.gaurab.org.np. He likes travel, photography and reading books when he is not on the network.

ACKNOWLEDGMENTS

I would like to acknowledge all those who have helped and contributed to the preparation of this e-primer. My colleagues at PCH, Bill Woodcock and Daniel S. Silverstein, have provided both structural and content inputs. Ritesh Raj Josti of Mercantile Communications, who is my co-instructor at APRICOT, has provided the basis for a lot of security-related materials.

Thanks to Bill Manning (EP.net) for reviewing DNS-related materials and Merike Kaeo (Doubleshot Security) for reviewing the security-related materials. Vicky Shrestha (WorldlinkCommunications) for doing a sanity checks on all the configurations.

I'd also like to thank the IOSN Team, Kenneth Wong, Sunil Abraham, Khairil Yusof, Phet Sayo and Shahid Akhtar, for being patient.

Lastly, thanks to all the reviewers – Philippe Langlois, Dinesh Nair, Devdas Bhagat, Nah Soo Hoe, Raj Mathur, Satyakam Goswami, Vimal Joseph and Simos Xenitellis.

APDIP

The Asia-Pacific Development Information Programme (APDIP) is an initiative of the United Nations Development Programme (UNDP) that aims to promote the development and application of information and communication technologies for sustainable human development in the Asia-Pacific region. APDIP aims to meet its goals by focusing on three inter-related core areas: (i) policy development and dialogue; (ii) access; and (iii) content development and knowledge management.

APDIP collaborates with national governments, regional, international and multi-lateral development organizations, UN agencies, educational and research organizations, civil society groups, and the private sector in integrating ICTs in the development process. It does so by employing a dynamic mix of strategies – awareness raising, capacity building, technical assistance and advice, research and development, knowledge sharing and partnership building.

http://www.apdip.net

IOSN

The International Open Source Network (IOSN) is an initiative of APDIP and supported by the International Development Research Centre of Canada. IOSN is a Centre of Excellence for Free/Open Source Software (FOSS), Open Content and Open Standards in the Asia-Pacific region. It is a network with a small secretariat based at the UNDP Regional Centre in Bangkok and three centres of excellence – IOSN ASEAN+3, IOSN PIC (Pacific Island Countries), and IOSN South Asia, based in Manila, Suva and Chennai respectively.

IOSN provides policy and technical advice on FOSS to governments, civil society and the private sector. It produces FOSS awareness and training materials and distributes them under open content licenses. It also organizes awareness raising, training, research and networking initiatives to assist countries in developing a pool of human resources skilled in the use and development of FOSS. IOSN works primarily through its web portal http://www.iosn.net that is collectively managed by the FOSS community. The web portal serves as a clearinghouse and a platform for knowledge sharing and collaborations.

http://www.iosn.net

International
Open
Source
Network

An initiative of the UNDP Asia-Pacific Development Information Programme and supported by the International Development Research Centre, Canada

IOSN
c/o Asia-Pacific Development Information Programme
UNDP Regional Centre in Bangkok
3rd Floor, United Nations Service Building
Rajdamnern Nok Avenue
Bangkok 10200, Thailand
Tel: +66 2 288 1234

IOSN ASEAN+3
National Telehealth Center
University of the Philippines Manila
Taft Avenue, Manila, Philippines 1000
Tel: +63 2 525 6501

IOSN PIC (Pacific Island Countries)
The University of the South Pacific
Private Bag
Suva, Fiji
Tel: +67 9 323 1000

IOSN South Asia
Centre for Development of Advanced Computing
Block-II, 6/13, Park Avenue, Keshava Perumal Puram
Chennai-600 028, India
Tel: +91 44 2461 0880

UNDP

IDRC CRDI