International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Carapace for Intranet Security of Linux Harding

Premchand Ambhore†*, B B Meshramr‡ and Archana Wankhade§

†Computer Science and Engineering, GCOE Amravati University, Amravati, (M.S.) India
‡Computer Engineering Department, VJTI Mumbai, (M.S.) India
§Information Technology Department, GCOE Amravati (M.S.) India

Accepted 07 March 2015, Available online 15 March 2015, **Vol.5, No.2 (April 2015)**

Abstract

The dominant operating system on the Internet, Linux is quite prevalent, considering that the overwhelming majority of servers running web services, email services, and name services all depend on other open-source code that works with Linux. And this is where the trouble begins. So we need to secure Linux operating system. The Linux should be secured from three perspectives. Workstation Security, Network security, Server security since most of the network application such as DHCP, Telnet and Web servers are running on Linux it is necessary to secure these applications. All the threat, attacks are mostly are network based so the Linux network service must be so powerful to prevent these kind of attacks.

Keywords*: DHCP, Security, Boot, BIOS*

1. Introduction

Enterprises have solicited the knowledge and skills of security experts to properly audit systems and tailor solutions to fit the operating requirements of their organization. Because most organizations are increasingly dynamic in nature, their workers are accessing critical company IT resources locally and remotely, hence the need for secure computing environments. Unfortunately, many organizations (as well as individual users) regard security as more of an afterthought, a process that is overlooked in favor of increased power, productivity, convenience, ease of use, and budgetary concerns. Proper security implementation is often enacted postmortem — *after* an unauthorized intrusion has already occurred. Taking the correct measures prior to connecting a site to an entrusted network, such as the Internet, is an effective means of thwarting many attempts at intrusion. Although not the dominant operating system on the Internet, Linux is quite prevalent, considering that the overwhelming majority of servers running web services, email services, and name services all depend on other open-source code that works with Linux. And this is where the trouble begins. So we need to secure Linux operating system.

The Linux should be secured from four perspectives. namely Installation of Linux, Workstation Security , Server security and Network security.

*Corresponding author **Premchand Ambhore** is Research scholar; **Dr.B B Meshramr** is working as Professor and **Archana Wankhade** as Assistant Professor

Since most of the network application such as DHCP, Telnet and Web servers are running on Linux, it is necessary to secure these applications. All the threat, attacks are mostly are network based so the Linux network service must be so powerful to prevent these kind of attacks.

2. Secure Installation of Linux

Security begins with the first time you put that CD or DVD into your disk drive to install Red Hat Enterprise Linux. Configuring your system securely from the beginning makes it easier to implement additional security settings later.

Disk Partitions

Create separate partitions for /boot, /, /home, /tmp, and /var/tmp. The reasons for each are different and we will address each partition .

/boot –The boot loader and kernel images that are used to boot your system into Red Hat Enterprise Linux are stored in this partition. This partition should not be encrypted. If this partition is included in / and that partition is encrypted or otherwise becomes unavailable then your system will not be able to boot.

/home - When user data (/home) is stored in / instead of in a separate partition, the partition can fill up causing the operating system to become unstable. Also, when upgrading your system to the next version of Red Hat Enterprise Linux it is a lot easier when you can keep your data in the /home partition as it will not be overwritten during installation. If the root partition

(/) becomes corrupt your data could be lost forever. By using a separate partition there is slightly more protection against data loss. You can also target this partition for frequent backups.

/tmp and /var/tmp - Both the /tmp and the /var/tmp directories are used to store data that doesn't need to be stored for a long period of time. However if a lot of data floods one of these directories it can consume all of your storage space. If this happens and these directories are stored within / then your system could become unstable and crash. For this reason, moving these directories into their own partitions is a good idea.

To get mounted partition information uses the following command.

 # df –lh

Where **df :** displays the amount disk space available on the file system

l: limit listening to local fiel system

h: print in human radable formate (i.e 22k)

Utilize LUKS Partition Encryption

During the installation process an option to encrypt your partitions will be presented to the user. The user must supply a passphrase that will be the key to unlock the bulk encryption key that will be used to secure the partition's data.

Manually Encrypting Directories

1. Enter runlevel 1 by typing the following at a shell prompt as root: **# init 1**

2. Unmount your existing **/home**: **# umount /sd7**

3. If the command in the previous step fails, use **fuser** command to find processes hogging **/home** and kill them: **# fuser -mvk /sd7**

Where **m :** mounted file system

v : verbose output

k : kill process access that file

4. Verify **/home** is no longer mounted:

 # grep sd7 /proc/mounts

5. Fill your partition with random data:

 # dd if= /dev/urandom of= /dev/sda7

This process can take many hours to complete.

6. Initialize your partition:

[root@prem /] # cryptsetup --verbose –verify-passphrase luksformat /dev/sda7

7. Open the newly encrypted device:

[root@prem /] # cryptsetup luksopen /dev/sda7 sd7

8. Make sure the device is present:

[root@prem /1 # ls -1 /dev/mapper ! grep sd7

9. Create a file system:**mkfs.ext3 /dev/mapper/home**

[root@prem /] # mkfs.ext4 dev/mapper/sd7

10. Mount the file system:

[root@prem sd7] # mont /dev/mapper/sd7 /sd7/

11. Make sure the file system is visible:

[root@prem sd]# df -hl

12. Add the following to the **/etc/crypttab** file:

 # home /dev/sd7 none

13. Edit the **/etc/fstab** file, removing the old entry for **/home** and adding the following line:

 # /dev/mapper/home /home ext3 defaults 1 2

14. Reboot the machine:

 # shutdown -r now

15. The entry in the **/etc/crypttab** makes your computer ask your **luks** passphrase on boot.

16. Log in as root and restore your backup.

Plan and Configure Security Updates

All software contains bugs. Often, these bugs can result in a vulnerability that can expose your system to malicious users. Un patched systems are a common cause of computer intrusions. You should have a plan to install security patches in a timely manner to close those vulnerabilities so they cannot be exploited.

Adjusting Automatic Updates

Red Hat Enterprise Linux is configured to apply all updates on a daily schedule. If you want to change how your system installs updates, you must do so via **Software Update Preferences**. In GNOME, you can find controls for your updates at: **System → Preferences → Software Updates**. In KDE, it is located at: **Applications → Settings → Software Updates**.

3. Workstation Security

3.1 Install Minimal Software

It is very critical to look at the default list of software packages and remove unneeded packages or packages that don't comply with your security policy, it is a good practice not to have development packages, desktop software packages (e.g. X Server) etc. installed on production servers. Other packages like FTP and Telnet daemons should not be installed as well unless there is a justified business reason for it (SSH/SCP/SFTP should be used instead).

A good approach is to start with a minimum list of RPMs and then add packages as needed.

To get a list of all installed RPMs you can use the following command:

 # rpm –qa

Where ***rpm :*** *RPM package manager,****-q :*** *query ,****-a :*** *all* If you want to know more about a particular RPM, run the following command:

 # rpm -qi <package_name>

Where ***rpm :*** *RPM package manager,****-q :*** *query ,****-i :*** *information*

To check for and report potential conflicts and dependencies for deleting a RPM, run following command

 # rpm -e --test <package_name>

Where ***rpm :*** *RPM package manager ,****e :*** *erase ,* ***test :*** *don't actually uninstall anything*

[root@prem]# rpm –e --test rpm

3.2 Install Signed Packages

Package signing uses public key technology to prove that the package that was published by the repository has not been changed since the signature was applied.

Verifying Signed Packages: All Red Hat Enterprise Linux packages are signed with the Red Hat *GPG* key. Assuming the disc is mounted in **/mnt/cdrom** , use the following command as the root user to import it into the *keyring* (a database of trusted keys on the system):
[root@prem cdrom]# rpm—import RPM-GPG-KEY-redhat-beta Now, the Red Hat GPG key is located in the **/etc/pki/rpm -gpg/** directory.
To display a list of all keys installed for RPM verification, execute the following command:
[root@prem cdrom] # rpm -qa gpq-pubkey*
Where ***rpm :*** *RPM package manager,-****q :*** *query , -****a :*** *all*
To verify all the downloaded packages at once, issue the following command:
[root@prem packages] # rpm –k rpcbind-0.2.0-8.el6.i686.rpm
Where ***rpm :*** *RPM package manager*
*-****K :*** *check the package key*
For each package, if the GPG key verifies successfully, the command returns **gpg md5 OK**. Packages that do not pass GPG verification should not be installed, as they may have been altered by a third party.

3.3 BIOS and Boot Loader Security

Password protection for the BIOS (or BIOS equivalent) and the boot loader can prevent unauthorized users who have physical access to systems from booting using removable media or obtaining root privileges through single user mode.

3.3.1 BIOS Passwords

The two primary reasons for password protecting the BIOS of a computer are:
1. *Preventing Changes to BIOS Settings*
2. *Preventing System Booting:* If you forget the BIOS password, it can either be reset with jumpers on the motherboard or by disconnecting the CMOS battery. For this reason, it is good practice to lock the computer case if possible.

3.3.2 Stealing/Changing Data Using a Bootable Linux CD

Once an attacker has gained physical access, getting into a box can be as simple as booting to a CD-based Linux distribution, deleting the root user account password in the /etc/shadow file (or replacing it with a known password and salt), and booting into the system, normally with full access.

To mitigate the damage Administrators often take common precautions to prevent further access.
BIOS passwords

3.3.3 Disabling bootable Linux CD

Disabling boot from removable media
Password-protected hard drives (easy to implement for workstations, but for servers requires hardware-level remote administration ability, such as IP KVM, Dell Drac card or the like)

3.3.4 Circumventing BIOS password

BIOS passwords are a very basic form of security and can be set to prevent the system from booting or to prevent the BIOS from being altered by unintended parties. They provide a minimum level of security with a minimum amount of effort. To assist in accessing the BIOS in the event an administrator has forgotten the BIOS password, many of the BIOS providers have included a backdoor BIOS password for easy recovery.

3.3.5 BIOS Password Bypass Techniques

If the machine has a BIOS password and you cannot boot and log in to it, you can bypass the password easily in several ways. The most common ways involve removing the CMOS battery, modifying jumper settings, and using various software utilities.

3.3.6 Preventing BIOS Password Circumvention

Since Linux distributions can be run from any form of removable media (CDs, VDs, floppy drives, and USB devices), disabling the ability to boot from any form of removable media is advisable and will keep out many of the lower-level, script-kiddie- attackers.

3.4 Boot Loader Passwords

The primary reasons for password protecting a Linux boot loader are as follows:
1. *Preventing Access to Single User Mode*
2. *Preventing Access to the GRUB Console*
3. *Preventing Access to Insecure Operating Systems*

3.5 Password Protecting GRUB

To do this, first choose a strong password, open a shell, log in as root, and then type the following command:
[root@prem log] # grub-md5-crypt
When prompted, type the GRUB password and press **Enter**. This returns an MD5 hash of the password. Next, edit the GRUB configuration file **/boot/grub/grub.conf**. Open the file and below the **timeout** line in the main section of the document, add the following line:
password --md5 ***<password-hash>***
Replace ***<password-hash>*** with the value returned by **/sbin/grub-m d5-crypt**.
Whole Disk or Partition Encryption The best way to protect against data tampering or unintended disclosure is to implement one of the many whole disk or partition encryption methodologies available to Linux systems. This entails encrypting the entire contents of the hard drive, or partition, using a cryptographic encryption algorithm.

3.5.1 Password Security

For security purposes, the installation program configures the system to use Secure Hash Algorithm

Table 1 Vulnerabilities, attacks and defense mechanisms

No	Vulnerability	Attacks	Countermeasure
3.1	No separate partition for /boot, /, /home, /tmp, and /var/tmp	System crash and data loss	Create separate partition for /boot, /, /home, /tmp, and /var/tmp
3.2	Unnecessary software's	Software vulnerability attack	Install minimum software's
3.3	maliciously altered package	System instability ,System crash and data loss, data still	Install Signed Packages
3.4	No BIOS password	Stealing/Changing Data Using a Bootable Linux CD	Give BIOS password
3.5	Single User Mode access	Access as root user without password	Password protecting BIOS
3.6	Access to the GRUB Console	change its configuration or to gather information using the ***cat*** command.	Password protecting GRUB
3.7	Access to Insecure Operating Systems	If it is a dual-boot system, an attacker can select an operating system at boot time (for example, DOS)	Password protecting GRUB
3.8	Weak password, no password or default password	Cracking of weak passwords	Enforcing Stronger Passwords Restricting Use of Previous Passwords Locking User Accounts After Too Many Login Failures
3.9	No password Aging	Use of Cracked password over long period of time	Apply good password Aging
3.10	root access to individual users	Machine Misconfiguration 1) Running Insecure Services	Disallowing Root Access Disallow Remote Root Login Disabling root access via any console device (tty)
3.11	Allowed *su* command to users	Access other user data and services	Limit and block *su* access
3.12	Enabled CTRL-ALT-Delete	Unauthorized System Shut down	Disable CTRL-ALT-Delete

512 (SHA512) and shadow passwords. The single most important thing a user can do to protect his account against a password cracking attack is create a strong password.

/etc/passwd file
[root@prem ~] #tail /etc/passwd
/etc/shadow file
[root@prem ~] # tail /etc/shadow
Disallow Remote Root Login: Any actions requiring a direct log on to the system via 'root' should be restricted to the local console. Edit /etc/securetty to reflect the following changes*:*

Disabling root access via any console device (tty)

To further limit access to the root account, administrators can disable root logins at the console by editing the ***/etc/securetty*** file.
To prevent the root user from logging in, remove the contents of this file by typing the following command at a shell prompt as root:

[root@prem]#echo > /etc/securetty

The following programs are prevented from accessing the root account:

Disabling root SSH logins

To prevent root logins via the SSH protocol, edit the SSH daemon's configuration file, **/etc/ssh/sshd_config**, and change the line that reads:**#PermitRootLogin yes** to read as follows:**PermitRootLogin no**
3.6 Disable CTRL-ALT-Delete
For those machines with poor or non-existent physical security, to disable the CTRL-ALT-Delete function that allows an attacker to shut down the machine. Edit /etc/inittab to comment out the following line:

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

Save the change and to restart the service for it to take effect run the following command:

[root@hod BBM]# */sbin/init q*

Thus we summarized the vulnerabilities, attacks and defense mechanisms is given in table 1

Server Security

When a system is used as a server on a public network, it becomes a target for attacks. Hardening the system and locking down services is therefore of paramount

importance for the system administrator. Before delving into specific issues, review the following general tips for enhancing server security: Keep all services current, to protect against the latest threats.,

Use secure protocols whenever possible ,- Serve only one type of network service per machine whenever possible , Monitor all servers carefully for suspicious activity.

4.1 Enhancing Security with TCP Wrappers

TCP Wrappers are capable of much more than denying access to services. This section illustrates how they can be used to send connection banners, warn of attacks from particular hosts, and enhance logging functionality. Refer to the **hosts options** man page for information about the T CP Wrapper functionality and control language.

TCP Wrappers and Connection Banners

To implement a TCP Wrappers banner for a service, use the banner option. For this example, the file is called */etc/banners/vsftpd* and contains the following lines:

220-Hello, %c
220-All activity on ftp.exam ple.com is logged.
220-Inappropriate use will result in your access privileges being removed.

The **%c** token supplies a variety of client information, such as the username and hostname, or the username and IP address to make the connection even more intimidating.

For this banner to be displayed to incoming connections, add the following line to the */etc/hosts.allow* file:
vsftpd : ALL : banners /etc/banners/

TCP Wrappers and Attack Warnings

If a particular host or network has been detected attacking the server, TCP Wrappers can be used to warn the administrator of subsequent attacks from that host or network using the **spawn** directive.

Assume that a cracker from the 206.182.68.0/24 network has been detected attempting to attack the server. Place the following line in the */etc/hosts.deny* file to deny any connection attempts from that network, and to log the attempts to a special file:
ALL : 206.182.68.0 : spawn /bin/echo `date` %c %d >> /var/log/intruder_alert
The **%d** token supplies the name of the service that the attacker was trying to access. To allow the connection and log it, place the **spawn** directive in the */etc/hosts.allow* file.

TCP Wrappers and Enhanced Logging

If certain types of connections are of more concern than others, the log level can be elevated for that service using the severity option. For this example, assume that anyone attempting to connect to port 23 (the T elnet port) on an FT P server is a cracker. To denote this, place an **emerg** flag in the log files instead of the default flag, **info**, and deny the onnection.
To do this, place the following line in */etc/hosts.deny*:
in.telnetd : ALL : severity emerg

This uses the default **authpriv** logging facility, but elevates the priority from the default value of **info** to **emerg**, which posts log messages directly to the console.

Protect portmap With TCP Wrappers

It is important to use TCP Wrappers to limit which networks or hosts have access to the **portmap** service since it has no built-in form of authentication. Further, use *only* IP addresses when limiting access to the service. Avoid using hostnames, as they can be forged by DNS poisoning and other methods.

4.2 Enhancing Security With xinetd

This section focuses on using xinetd to set a trap service and using it to control resource levels available to any given xinetd service. Setting resource limits for services can help thwart Denial of Service (DoS) attacks.
Setting a Trap :One important feature of xinetd is its ability to add hosts to a global no_access list. Hosts on this list are denied subsequent connections to services managed by xinetd for a specified period or until xinetd is restarted. You can do this using the SENSOR attribute. This is an easy way to block hosts attempting to scan the ports on the server. The first step in setting up a SENSOR is to choose a service you do not plan on using. For this example, Telnet is used. Edit the file /etc/xinetd.d/telnet and change the flags line to read:
flags = SENSOR
Add the following line:
deny_time = 30

This denies any further connection attempts to that port by that host for 30 minutes. Other acceptable values for the **deny_tim e** attribute are FOREVER, which keeps the ban in effect until **xinetd** is restarted, and NEVER, which allows the connection and logs it. Finally, the last line should read:
disable = no

An attacker who knows that a **SENSOR** is running can mount a Denial of Service attack against particular hosts by forging their IP addresses and connecting to the forbidden port.

Controlling Server Resources

Another important feature of **xinetd** is its ability to set resource limits for services under its control. It does this using the following directives:

1.cps = <num ber_of_connections> <wait_period> — Limits the rate of incoming connections. This directive takes two arguments:
WHERE <num ber_of_connections> — The number of connections per second to handle. If the rate
of incoming connections is higher than this, the service is temporarily disabled. The default value is fifty (50).
<wait_period> — The number of seconds to wait before re-enabling the service after it has
been disabled. The default interval is ten (10) seconds.
2.instances = <num ber_of_connections> — Specifies the total number of connections allowed to a service. T his directive accepts either an integer value or UNLIMIT ED.
per_source = <num ber_of_connections> — Specifies the number of connections allowed to a service by each host. T his directive accepts either an integer value or UNLIMIT ED.
rlim it_as = <num ber[K|M]> — Specifies the amount of memory address space the service
can occupy in kilobytes or megabytes. This directive accepts either an integer value or UNLIMIT ED.
rlim it_cpu = <num ber_of_seconds> — Specifies the amount of time in seconds that a
service may occupy the CPU. This directive accepts either an integer value or UNLIMIT ED.
Using these directives can help prevent any single **xinetd** service from overwhelming the system, resulting in a denial of service.

Securing Portmap

The **portmap** service is a dynamic port assignment daemon for RPC services such as NIS and NFS. It has weak authentication mechanisms and has the ability to assign a wide range of ports for the services it controls. For these reasons, it is difficult to secure.
If running RPC services, follow these basic rules.

Securing the Apache HTTP Server

Always verify that any scripts running on the system work as intended *before* putting them into production. Also, ensure that only the root user has write permissions to any directory containing scripts or CGIs. To do this, run the following commands as the root user:
 # chown root <directory_name>
 # chmod 755 <directory_name>
System administrators should be careful when using the following configuration options (configured in */etc/httpd/conf/httpd.conf*):
1. FollowSym Links: This directive is enabled by default, so be sure to use caution when creating symbolic links to the document root of the Web server. For instance, it is a bad idea to provide a symbolic link to
2. Indexes: This directive is enabled by default, but may not be desirable. To prevent visitors from browsing files on the server, remove this directive.
3. UserDir: The **UserDir** directive is disabled by default because it can confirm the presence of a user account on the system. To enable user directory browsing on the server, use the following
directives:
UserDir enabled
UserDir disabled root
These directives activate user directory browsing for all user directories other than */root/*. T o
add users to the list of disabled accounts, add a space-delimited list of users on the **UserDir**
disabled line.
4.ServerT okens: The **ServerTokens** directive controls the server response header field which is sent back to clients. It includes various information which can be customized using the following parameters:
5. ServerT okens Full (default option) — provides all available information (OS type and
used modules), for example:
Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2
6. ServerT okens Prod or **ServerT okens**
ProductOnly

It is recommended to use the **ServerTokens Prod** option so that a possible attacker does not gain any valuable information about your system.

Removing httpd Modules

In certain scenarios, it is beneficial to remove certain **httpd** modules to limit the functionality of the HTTP Server. To do so, simply comment out the entire line which loads the module you wish to remove in the */etc/httpd/conf/httpd.conf* file. For example, to remove the proxy module, comment out the following line by prepending it with a hash sign:
#LoadModule proxy_m odule m odules/m od_proxy.so
Securing FTP
The *File Transfer Protocol* (FTP) is an older TCP protocol designed to transfer files over a network.
Red Hat Enterprise Linux provides three FTP servers.
1.gssftpd — A Kerberos-aware
2.xinetd-based FTP daemon that does not transmit authentication information over the network.
3.Red Hat Content Accelerator (**tux**) — A kernel-space Web server with FTP capabilities.
vsftpd — A standalone, security oriented implementation of the FTP service. The following security guidelines are for setting up the **vsftpd** FTP service.

Configuring vsftpd for Anonymous FTP

The vsftpd.conf file controls the vsftpd daemon. The vsftpd binary has only one ommandline option, which allows you to specify the location of the vsftpd.conf configuration file.
 # vsftpd /etc/vsftpd.conf
Following shows a simple configuration file for a secure stand-alone anonymous FTP server that allows only downloads.

The /etc/vsftpd.conf file

 # General Configuration
listen=YES
background=YES
listen_address=192.168.0.1
nopriv_user=ftp_nopriv
xferlog_enable=YES
 # Mode and Access rights
anonymous_enable=YES
local_enable=NO
write_enable=NO
cmds_allowed=PASV,RETR,QUIT
 # Security
ftpd_banner=Puppy.YourDomain.Net FTP Server
connect_from_port_20=YES
hide_ids=YES
pasv_min_port=50000
pasv_max_port=60000
 # DoS
ls_recurse_enable=NO
max_clients=200
max_per_ip=4

Securing SSH

Many network services like telnet, rlogin, and rsh are vulnerable to eavesdropping which is one of several reasons why SSH should be used instead. The Restricting System Access from Servers and Networks shows how direct logins can be disabled for shared and system accounts including root.
For securing SSH Use the following parameters :
PermitRootLogin no
UsePrivilegeSeparation yes
AllowTcpForwarding no
StrictModes yes
Ensure that all host-based authentications are disabled. These methods should be avoided as primary authentication.
IgnoreRhosts yes
HostbasedAuthentication no
RhostsRSAAuthentication no
Disable sftp if it's not needed:
#Subsystem sftp /usr/lib/misc/sftp-server
After changing any directives make sure to restart the sshd daemon run the following command:
root@prem] # /etc/init.d/sshd restart

ssh passphrase

Passwords aren't very secure, Anyone who gains access to your drive has gained access to every system you use that key with. This is also a Very Bad Thing. The solution is obvious, add a passphrase.
Step by step procedure to create passphrase for ssh as follow.

- Ssh server is running sshd so it will allow remote ssh login and which stores public key .
- ssh client has ssh client which is used to get ssh access to ssh server . Passphrase is created on ssh client and store private key.
- Create user ***prem*** on both server and client .
- **<u>On client</u>**

1) Generate an RSA key pair by typing the following command at a shell prompt:

 # ssh-keygen –d

Where: -d : dsa It will create .ssh directory in user home directory .Then it will ask for the passphrase ,Enter the passphrase , the two files are created in .ssh directory,
1 id_dsa (private key) id_dsa.pub (public key)
2) Login on server from client using ssh

- **<u>On sever</u>**

1) Create .ssh directory in home directory (/home/hod) of user ***prem.*** Use the following command .

 # mkdir .ssh

2) To Go inside .ssh directory use follwing command ***.***

 # cd .ssh

3) Use the following command to copy public key into server.

 # scp hod@client_ip:.ssh/id_dsa.pub authorized_keys

Where :
scp : *secure copy command*

- Set owner permissions on /home/hod/.ssh directory and authorized_keys file.
- To set permission use the following commands.

 # chmod 700 /home/prem/.ssh
 # chmod 600 authorized_keys

Where :
chmod : change the file or directory permissions

<u>On client:</u> ssh agent will send passphrase on behalf of user.

1) Edit /home/prem/.bashrc and add the following line

eval `ssh-agent`

This will start the ssh-agen at the time of login***.***
 # .bashrc

- Restart the system or run the following command to reflect chages in .bashrc to kernel
- [scholar@prem ~]$ source .bashrc

This wiil start the ssh agent.

- Register passphrase to ssh agent run the following command.

 # ssh-add
[scholar@prem ~]$-ad

- ssh acces using passphrase

- [ssh@prem packages] $ 192.168.10.11

This will ask passphrase for the private key to use and to get authenticate.

Securing website

- Public websites deployed on the web server as unsecure because anyone can have access to these website. this will lead to anonymous access to website .
- To allow only employee or only authorized user to access the websites , make authenticating website on web server.
- Step by step procedure to make password authenticating website as follow.

Creating web user

- create the local linux user, to crate linux user add the following command.

 # useradd web

- Convert local linux user to web user who will have access to website, run the following command.

 # htpasswd –c /etc/httpd/conf/.htpasswd web

Where:
htpasswd : creates web user.
-c: crate the file
.**htpasswd:** user databse file for the website.

2) Create **.htaccess** file in document root directory i.e /var/www/hod/www.nims.vjti.in/.htaccess

- Add the following lines in **.htaccess** file.

2) Opern **/etc/httpd/conf/httpd.conf** and add following lines.

No	Vulnerability	Attacks	Countermeasure
4.1	**FTP** Anonymous access Too many user access	Unauthorized access Denial of Service Attacks (DoS*)*	Apply proper security parameter Apply DOS security parameters
4.2	ssh password	Cracking of password	Use passphrase
4.3	Unauthorized websites	Unauthorized access	Authenticate the website

This will allow web server to authenticate the web site.

5. Network Security

The Network Manager Daemon

The Network Manager daemon runs with root privileges and is usually configured to start up at boot time. You can determine whether the Network Manager daemon is running by entering this command as root:

 # service NetworkManager status

NetworkManager (pid 1527) is running...

The service command will report Network Manager is stopped if the Network Manager service is not running. To start it for the current session:

 # service NetworkManager start

Run the *chkconfig* command to ensure that NetworkManager starts up every time the system boots:

 # chkconfig NetworkManager on

Insecure Services

Potentially, any network service is insecure. This is why turning off unused services is so important. Exploits for services are routinely revealed and patched, making it very important to regularly update packages associated with any network service. Some network protocols are inherently more insecure than others. These include any services that:
Transmit Usernames and Passwords Over a Network Unencrypted — Many older protocols, such as Telnet and FTP, do not encrypt the authentication session and should be avoided whenever possible.
Transmit Sensitive Data over a Network Unencrypted — Many protocols transmit data over the network unencrypted. These protocols include Telnet, FTP, HTTP, and SMTP. Many network file systems, such as NFS and SMB, also transmit information over the network unencrypted. It is the user's responsibility when using these protocols to limit what type of data is transmitted. Remote memory dump services, like *netdump*, transmit the contents of memory over the network unencrypted. Memory dumps can contain passwords or, even worse, database entries and other sensitive information. Other services like *finger* and *rwhod* reveal information about users of the system. Examples of inherently insecure services include **rlogin**, **rsh**, **telnet**, and **vsftpd**.
All remote login and shell programs (**rlogin**, **rsh**, and **telnet**) should be avoided in favor of SSH. FTP is not as inherently dangerous to the security of the system as remote shells, but FTP servers must be carefully configured and monitored to avoid problems.
Services that should be carefully implemented and behind a firewall include:
Finger : user information lookup programme
Authd : mail policy server; this server talks to postfix, sendmail, and anything else that wants to link to libauthd

nfs : network file system
sendmail:
smb (Samba)

Security Enhanced Communication Tools

As the size and popularity of the Internet has grown, so has the threat of communication interception. Over the years, tools have been developed to encrypt communications as they are transferred over the network. Red Hat Enterprise Linux 6 ships with two basic tools that use high-level, public-key-cryptography-based encryption algorithms to protect information as it travels over the network.
OpenSSH — A free implementation of the SSH protocol for encrypting network communication.
Gnu Privacy Guard (GPG) — A free implementation of the PGP (Pretty Good Privacy) encryption application for encrypting data.

OpenSSH is a safer way to access a remote machine and replaces older, unencrypted services like **telnet** and **rsh**. OpenSSH includes a network service called **sshd** and three command line client applications:
ssh — A secure remote console access client.
scp — A secure remote copy command.
sftp — A secure pseudo-ftp client that allows interactive file transfer sessions.

Available Network Services

While user access to administrative controls is an important issue for system administrators within an organization, monitoring which network services are active is of paramount importance to anyone who administers and operates a Linux system. Many services under Red Hat Enterprise Linux 6 behave as network servers. If a network service is running on a machine, then a server application (called a *daemon*), is listening for connections on one or more network ports. Each of these servers should be treated as a potential avenue of attack

Risks to Services

Network services can pose many risks for Linux systems. Below is a list of some of the primary issues:
Denial of Service Attacks (DoS) — By flooding a service with requests, a denial of service attack can render a system unusable as it tries to log and answer each request.

Distributed Denial of Service Attack (DDoS)

A type of DoS attack which uses multiple compromised machines (often numbering in the thousands or more) to direct a coordinated attack on a service, flooding it with requests and making it unusable.

Script Vulnerability Attacks

If a server is using scripts to execute server-side actions, as Web servers commonly do, a cracker can attack improperly written scripts. These script vulnerability attacks can lead to a buffer overflow condition or allow the attacker to alter files on the system.

Buffer Overflow Attacks

Services that connect to ports numbered 0 through 1023 must run as an administrative user. If the application has an exploitable buffer overflow, an attacker could gain access to the system as the user running the daemon. Because exploitable buffer overflows exist, crackers use automated tools to identify systems with vulnerabilities, and once they have gained access, they use automated rootkits to maintain their access to the system.

Note

The threat of buffer overflow vulnerabilities is mitigated in Red Hat Enterprise Linux by *ExecShield*, an executable memory segmentation and protection technology supported by x86-compatible uni- and multi-processor kernels. ExecShield reduces the risk of buffer overflow by separating virtual memory into executable and non-executable segments. Any program code that tries to execute outside of the executable segment (such as malicious code injected from a buffer overflow exploit) triggers a segmentation fault and terminates. Execshield also includes support for *No eXecute* (NX) technology on AMD64 platforms and *eXecute Disable* (XD) technology on Itanium and Intel® 64 systems. T hese technologies work in conjunction with ExecShield to prevent malicious code from running in the executable portion of virtual memory with a granularity of 4KB of executable code, lowering the risk of attack from stealthy buffer overflow exploits.

Identifying and Configuring Services

To enhance security, most network services installed with Red Hat Enterprise Linux are turned off by default. There are, however, some notable exceptions:
Cupsd — The default print server for Red Hat Enterprise Linux.
lpd — An alternative print server.
xinetd— A super server that controls connections to a range of subordinate servers, such as gssftp and telnet.
sendmail— The Sendmail *Mail Transport Agent* (MTA) is enabled by default, but only listens for connections from the localhost.
sshd— The OpenSSH server, which is a secure replacement for Telnet.

When determining whether to leave these services running, it is best to use common sense and err on the side of caution. For example, if a printer is not available, do not leave cupsd running. The same is true

for portmap . If you do not mount NFSv3 volumes or use NIS (the ypbind service) , then portmap should be disabled.

Detecting Listening Network Ports

One of the most important tasks is to detect and close network ports that are not needed. To get a list of listening network ports (TCP and UDP sockets), you can run the following command:

 # netstat –tulp

Where ***-t :*** *list TCP socket ,* ***-u :*** *list UDP socket,* ***-l :*** *show only linsting socket*

-p: *show the PID and name of the program to which socket belongs*

 # nmap -sTU <remote_host>

Starting nmap 3.70 (http://www.insecure.org/nmap/) at 2004-12-10 22:51 CST
Interesting ports on jupitor (172.16.0.1):
(The 3131 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE

22/tcp open ssh
113/tcp open auth
Nmap run completed -- 1 IP address (1 host up) scanned in 221.669 seconds

Note that the above nmap command can take a while. If you remove the UDP port scan (without the option -U), then nmap will finish the port scan immediately. If you run
it on the local machine it will also complete very fast. Also note that nmap might not show all listening network sockets if a firewall is being used to block ports.
From the output above you can see that the xinetd daemon is listening on port auth (port 113) for IDENT
Another method to list all of the TCP and UDP sockets to which programs are listening is lsof:

 # lsof | egrep 'COMMAND|LISTEN|UDP'

Where losf : *shows the list of open files*

[root@hod BBM]# lsof | egrep 'COMMAND|LISTEN|UDP'

COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
*sshd 2317 root 3u IPv6 6579 TCP *:ssh (LISTEN)*
*xinetd 2328 root 5u IPv4 6698 TCP *:auth (LISTEN)*
sendmail 2360 root 3u IPv4 6729 TCP 127.0.0.1:smtp (LISTEN)
Closing Network Ports and Disabling Runlevel System Services
One of the most important tasks is to remove any network services from the system startup process that are not needed. On Red Hat systems you can list all services which are started at bootup using the following command:

 # chkconfig --list |grep on

You will notice that there are quite a lot of services enabled on your system. But many runlevel services (Stand-Alone Services) are not network related services like kudzu which is responsible for detecting and configuring new and/or changed hardware on your system. This service is only run during the boot process. Ensure not to disable runlevel services that are needed by the system to run smoothly.
Here are examples of Red Hat Runlevel System Services which you may or may not want to enable:

gpm needed if you want to use the mouse at the console
kudzu important for detecting new hardware
syslog important for syslog services
netfs needed only if there are NFS shares that should be mounted at boot time network important for starting network interfaces (e.g. eth0, eth1, bonding,...)
random used for the system entropy pool
atd needed if the at(1) service is used instead of cron
apmd Advanced Power Management (APM) daemon is used for laptops and some desktops
isdn needed if ISDN is being used
iptables needed if Netfilter (iptables) Firewall is being used
ip6tables needed if ip6tables Firewall is being used
pcmcia not needed on servers - needed for laptops
irqbalance important for distributing interrupts across all CPUs
sendmail needed if Sendmail is used - Procmail should be used which is more secure
autofs needed if automounter is used - production applications should not be dependent on automounter
sshd important for logins via SSH
portmap needed if e.g. NFS is being used
nfslock needed if NFS shares are mounted
nfs needed if server runs the NFS server
mdmonitor needed only if software RAID is being used
crond important for running cron jobs
xinetd needed if xinetd services are being used, see /etc/xinetd.d/ for list of services
cups needed if CUPS is used for the printing system
rhnsd needed if server should connect to RHN to check for software updates etc.
sysstat needed to reset system statistics logs
audit needed only if Linux Audit Subsystem (LAuS) should run for collecting system call audit records
psacct needed only if kernel process accounting information is needed
smartd important for monitoring disk problems if hard disks support SMART technology

netdump important if kernel oops data and memory dumps should be sent to a Netdump server for server crashes

The start/stop scripts of all runlevel services can be found in the */etc/init.d* directory. For example, if you don't know what the atd service does, go to */etc/init.d* and open the file atd. And in the script look for lines that start programs. In the atd script the daemon /usr/sbin/atd line starts the binary atd. Now having the name of the program that is started by this service, you can check the online pages of atd by running man atd. This will help you to find out more about a system service.

To permanently disable e.g. the runlevel service nfs, run:

#chkconfig nfs off

To immediately disable the runlevel service nfs, run:

 # /etc/init.d/nfs stop

Closing Network Ports and Disabling Xinetd Services

The xinetd daemon [2] is a replacement for inetd, the internet services daemon. It monitors the ports for all network services configured in */etc/xinetd.d,* and starts the
services in response to incoming connections.

To check if xinetd is enabled and running, execute the following two commands:

 # chkconfig --list xinetd

xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off

 # /etc/init.d/xinetd status

xinetd (pid 2619) is running...

If xinetd is active, it is important to check which Unix services are active and controlled by xinetd.

The following command shows all services configured in */etc/xinetd.d* and wheter xinetd monitors the ports for these services:

 # chkconfig --list | awk '/xinetd based services/,//'

xinetd based services:
krb5-telnet: off
rsync: off
eklogin: off
gssftp: off
klogin: off
chargen-udp: off
kshell: off
auth: on
chargen: off
daytime-udp: off
daytime: off
echo-udp: off
echo: off
services: off
time: off
time-udp: off
cups-lpd: off

To get a list of only active services for which xinetd monitors the ports, run following command:

 # chkconfig --list | awk '/xinetd based services/,//' | grep -v off

xinetd based services:
auth: on

In the above example you can see that the telnet-server RPM is not installed on the system. If the Telnet Server package telnet-server would be installed, it would show up on the list whether it's active or not. Here is an example how to disable a service. Assuming the telnet service is active, run the following commands to disable it and to see how the telnet service entries are being updated:

Run the following command to check telnet status

 # chkconfig --list telnet

telnet on

Run the following command to read the configuration file value:

 # cat /etc/xinetd.d/telnet | grep disable

disable = no

Run the following command to disable telnet:

 # chkconfig telnet off

Run the following command to check telnet status

 # chkconfig --list telnet

telnet off

Run the following command to read the configuration file value:

 # cat /etc/xinetd.d/telnet | grep disable

disable = yes

For the telnet service it would be better to remove the package from the system since SSH should be used instead:

Run the following command to remove the telnet package:

 # rpm -e telnet-server

Where

-e : *erase i.e remove the package*

It is important to investigate all active xinetd services and to disable them if they are not needed. Here is an example how to find out what a service does. Assuming you don't know what the auth service does which is listed as active in the list above, run the following

Commands: run the following command to get service information:

 # rpm -qi authd-1.4.1-1.rhel3 | awk '/Description/,//'

Where ***-q :*** *query* ***-i :*** *information of package*

Description :

authd is a small and fast RFC 1413 ident protocol daemon with both xinetd server and interactive modes that supports IPv6 and IPv4 as well as the more popular features of pidentd.

 # rpm -ql authd-1.4.1-1.rhel3

Where ***-q :*** *query,****-l :*** *list the files in package*

/etc/ident.key
/etc/xinetd.d/auth
/usr/sbin/in.authd
/usr/share/doc/authd-1.4.1

/usr/share/doc/authd-1.4.1/COPYING
/usr/share/doc/authd-1.4.1/README.html
/usr/share/doc/authd-1.4.1/rfc1413.txt
/usr/share/locale/ja/LC_MESSAGES/authd.mo
This example shows what can be done if there exists no online manuals for the binary in.authd that is started by xinetd. The steps above should be helpful for finding out more about services. The auth service (aka IDENT, see RFC 1413) allows remote daemons to query information about users establishing TCP connections on the local server. In a trusted

Environment it helps a server to identify who is trying to use it. For example, it can provide vital information for troubleshooting and who has done what. IDENT requests are needed by some applications like IRC. However, IDENT can be a security risk.

To disable the auth service, run the following command:

 # chkconfig auth off

The xinetd daemon is quite flexible and has many features. Here are just a few functionalities of Xinetd:
- Acces control for TCP, UDP, and RPC services
- Acess limitations based on time
- Provides mechanisms to prevent DoS attacks

Reviewing Inittab and Boot Scripts

The inittab file */etc/inittab* also describes which processes are started at bootup and during normal operation. For example, Oracle uses it to start cluster services at bootup.

Therefore, it is recommended to ensure that all entries in /etc/inittab are legitimate in your environment. At least remove the CTRL-ALT-DELETE trap entry to prevent accidental reboots:

 # sed -i 's/ca::ctrlaltdel:/#ca::ctrlaltdel:/g'

The default runlevel should be set to 3 since in my opinion X11 (X Windows System) should not be running on a production server. In fact, it shouldn't even be installed.

 # grep ':initdefault' /etc/inittab

id:3:initdefault:

And depending on your environment you might want to comment out the UPS entries as well. To have changes in /etc/inittab become effective immediately, you can run:

 # init q

The */etc/rc.local* script is used for commands or startup scripts which are pertinent only to a specific server. (*/etc/rc.local* is a link to */etc/rc.d/rc.local*). Ensure that all startup scripts in */etc/rc.d/rc.local* are legitimate.

Restricting System Access from Servers and Networks

Usually a firewall is used to protect a server from other servers and networks. However, in some cases you may also want to protect a server within a network by using a TCP Wrapper. The Xinetd super server that comes with most Linux distributions includes a built-in TCP wrapper. It can be used to explicitly define network services to accept incoming connections from specified servers and networks. The TCP wrappers implements access control through the use of two files, */etc/hosts.allow* and */etc/hosts.deny*. Note that the hosts.allow file takes precedence over the hosts.deny file. And you may want to change the permissions on the two configuration files since they are both world readable.

A recommended security-strategy is to block all incoming requests by default, but allow specific hosts or networks to connect. This is the strategy I will describe here.

To deny everything by default, add the following line to */etc/hosts.deny:*

ALL: ALL

To accept incoming SSH connections from e.g. nodes rac1cluster, rac2cluster and rac3cluster, add the following line to */etc/hosts.allow:*

sshd: rac1cluster rac2cluster rac3cluster

To accept incoming SSH connections from all servers from a specific network, add the name of the subnet to */etc/hosts.allow*. For example:

sshd: rac1cluster rac2cluster rac3cluster .subnet.example.com

To accept incoming portmap connections from IP address 192.168.0.1 and subnet 192.168.5, add the following line to */etc/hosts.allow*:

portmap: 192.168.0.1 192.168.5.

To accept connections from all servers on subnet .subnet.example.com but not from server cracker.subnet.example.com, you could add the following line to */etc/hosts.allow:*

ALL: .subnet.example.com EXCEPT cracker.subnet.example.com

Here are other examples that show some features of TCP wrapper:

If you just want to restrict ssh connections without configuring or using */etc/hosts.deny,* you can add the following entries to */etc/hosts.allow*:

sshd: rac1cluster rac2cluster rac3cluster
sshd: ALL: DENY

Kernel Tunable Security Parameters

The following list shows tunable kernel parameters you can use to secure your Linux server against attacks. For each tunable kernel parameters I will show the entry that needs to be added to the */etc/sysctl.conf* configuration file to make the change permanent after reboots.

To activate the configured kernel parameters immediately at runtime, use the following command:

 # sysctl –p

Where **-p :** load the system setting from the file specifief or /etc/sysctl.conf

Enable TCP SYN Cookie Protection: A SYN Attack is a denial of service attack that consumes all the resources

on a machine. Any server that is connected to a network is potentially subject to this attack. To enable TCP SYN Cookie Protection, edit the */etc/sysctl.conf* file and add the following line:

net.ipv4.tcp_syncookies = 1

Disable IP Source Routing: Source Routing is used to specify a path or route through the network from source to destination. This feature can be used by network people for diagnosing problems. However, if an intruder was able to send a source routed packet into the network, then he could intercept the replies and your server might not know that it's not communicating with a trusted server.

To enable Source Route Verification, edit the */etc/sysctl.conf* file and add the following line:

net.ipv4.conf.all.accept_source_route = 0

Disable ICMP Redirect Acceptance

ICMP redirects are used by routers to tell the server that there is a better path to other networks than the one chosen by the server. However, an intruder could potentially use ICMP redirect packets to alter the hosts's routing table by causing traffic to use a path you didn't intend.

To disable ICMP Redirect Acceptance, edit the */etc/sysctl.conf* file and add the following line:

net.ipv4.conf.all.accept_redirects = 0

Enable IP Spoofing Protection

IP spoofing is a technique where an intruder sends out packets which claim to be from another host by manipulating the source address. IP spoofing is very often used for denial of service attacks. To enable IP Spoofing Protection, turn on Source Address Verification. Edit the */etc/sysctl.conf* file and add the following line:

net.ipv4.conf.all.rp_filter = 1

Enable Ignoring to ICMP Requests

If you want or need Linux to ignore ping requests, edit the /etc/sysctl.conf file and add the following line:

net.ipv4.icmp_echo_ignore_all = 1

This cannot be done in many environments.

Enable Ignoring Broadcasts Request

If you want or need Linux to ignore broadcast requests, edit the /etc/sysctl.conf file and add the following line:

net.ipv4.icmp_echo_ignore_broadcasts = 1

Enable Bad Error Message Protection: To alert you about bad error messages in the network, edit the */etc/sysctl.conf* file and add the following line:

net.ipv4.icmp_ignore_bogus_error_responses = 1

Enable Logging of Spoofed Packets, Source Routed Packets, Redirect Packets

To turn on logging for Spoofed Packets, Source Routed Packets, and Redirect Packets, edit the */etc/sysctl.conf* file and add the following line:

net.ipv4.conf.all.log_martians = 1

No	Vulnerability	Attacks	Countermeasure
5.1	OS fingerprinting	Get os information like OS version etc.	Place login banner
5.2	Local log monitoring	Remove of log entries and log files	Remote log monitoring
5.3	Insecure Services FTP , Telnet	Get user name and password.	Avoid these services and use behind the firewall,
	Transmit Usernames and Passwords Over a Network Unencrypted	Denial of Service Attacks (DoS*)*	Use tcp_wrappers and xinetd Use SSH
5.4	/etc/sysctl.conf configuration file vulnerability	SYN Attack IP Source Routing IP Spoofing Broadcasts Request	Properly configure/etc/sysctl.conf

Conclusion

One of the biggest threats to your security is simply doing nothing. No matter how secure your hosts are at this point in time, they will, at varying rates, become less secure as time goes by. This is a consequence of simple entropy, as changes to your applications, environment, and requirements alter the configuration and potentially purpose of your systems. It is also a consequence of the changing nature of the threats against you. What you have protected yourself against now may not be what you need to protect yourself against in the future. This is most obviously manifested as new vulnerabilities and exploits of those vulnerabilities are discovered in the operating systems, applications, and tools you have running. You need to ensure you include security administration and monitoring as part of your regular system administration activities. Check your logs, audit your users and groups, and monitor your files and objects for suspicious activity. Know the routines and configuration of your hosts; the more you understand about the normal rhythms of your hosts, the easier it is to spot anomalies that could indicate you are under attack or have been penetrated.Finally, the truly vigilant test. And test again. Perform regular security assessments of your hosts and environment. Scan for vulnerabilities using tools such as Nessus or commercial tools such as ISS Security Scanner. Consider using independent third parties to perform penetration testing of your environment and hosts.

Ongoing security assurance is vital to make sure you stay protected and hardened from attack.

Because of the increased reliance on powerful, networked computers to help run businesses and keep track of our personal information, entire industries have been formed around the practice of network and computer security. Enterprises have solicited the knowledge and skills of security experts to properly audit systems and tailor solutions to fit the operating requirements of their organization. Because most organizations are increasingly dynamic in nature, their workers are accessing critical company IT resources locally and remotely, hence the need for secure computing environments has become more pronounced. Thus the Linux security is centered on proper system, application and server configuration and setting the security related attribute.

References

P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. . NSA 1998 The Inevitabil-ity of Failure: The Flawed Assumption of Security in Modern Computing Environments. In *21st National Information Systems Security Conference*, pages 303– 314.

C. J. PeBenito, F. Mayer, and K. MacMillan. 2006. Refer-ence Policy for Security Enhanced Linux. In *SELinux Symposium*,

R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. An-dersen, and J. Lepreau. August 1999. The Flask Security Architecture: System Support for Diverse Security Policies. In *The Eighth USENIX Security Symposium*, pages 123– 139.

R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. An-dersen, and J. Lepreau. August 1999. The Flask Security Architecture: System Support for Diverse Security Policies. In *The Eighth USENIX Security Symposium*, pages 123– 139.

Nigel Edwards, Joubert Berger, and Tse Houng Choo. , November 2001. A Secure Linux Platform. In *Proceedings of the 5th Annual Linux Showcase and Conference* Crispin Cowan, Steve Beattie, Calton Pu, PerryWagle, and Virgil Gligor. December 2000 SubDomain: Parsimonious Server Security. In *USENIX 14th Systems Administration Conference (LISA)*, *New Orleans, LA.*

R. Wita and Y. Teng-Amnuay. IEEE, 2005. Vulnerability profile for linux. In Proceedings of the 19th International Conference on Advanced Information Networking and Applications, pages 953–958.

Rhat enterprise linux 6 security guide (Red Hat Engineering Content Services). Implementation of Object Oriented Database Security P.B.Ambhore, V.B.Waghmare B.B.Meshram August 2007, IEEE International conference SERA 2007 at Bhusan Korea.20-22.

Ranjit Shrirang Nimbalkar, Dr. B. B. Meshram June – 2013 , *Survey On Integrated Threat Management (ITM)* International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 6, 290-297.