IJCSE International Journal of Computer Sciences and Engineering Open Access
Technical Notes Volume-3, Issue-3 E-ISSN: 2347-2693

Server Hardening: Securing Unix-like workstations

Yogender Bhardwaj

Computer Science and Engineering, Guru Gobind Singh Indraprastha University, India

www.ijcseonline.org

Received: Feb/09/2014 Revised: Feb/28/2014 Accepted: Mar/20/2014 Published: Mar/31/ 2014

***Abstract*—** In this fast-paced and dynamic world of data communication, systems and software development, and uncontrolled network traffic, security is becoming more and more of an issue. Surprisingly, many organizations, as well as individual users regard security as more of an afterthought (proper security implementation is often enacted after an unauthorized intrusion has already occurred) , a process that is overlooked in favor of increased productivity, convenience, ease of use, and budgetary concerns. Security on Linux systems never stays static. Once secured, the system does not perpetually stay secure. Indeed, the longer one uses the system, the less secure it becomes.

This document provides general practices, procedures, planning and tools for creating a secured computing environment for the data center, workplace, or at home. It is aimed for engineers, IT managers, security and system administrators, who are assumed to possess basic system administration skills for Unix-like systems. It addresses basic security vulnerabilities, local and remote intrusion, exploitation and malicious activity techniques valid for all Linux systems.

***Keywords*—** Linux, Unix, Hardening, Security, Server

I. INTRODUCTION

Hardening Linux distributions can be considered on two basic levels, workstation level hardening and application level hardening. This document deals only with workstation level hardening. Workstation hardening relates with securing physical controls, administrative controls, as well as Operating System internals. Whereas, application hardening concerns with user applications, network services , firewalls and system applications such as SELinux, GPG, SSL,etc. Application hardening is beyond the scope of this topic.

A. What is to be secured and from whom?

Before one attempts to secure the system, it is to be analyzed what is being protected and why? Also the intensity and frequency of the threat is to be determined, against which the system is being protected.

Identifying the type of intruder [2] :

- The Curious - Intruder is basically interested in finding out the type of system and data
- The Malicious - Intruder forces to spend time and money recovering from the damage he has caused
- The High-Profile Intruder - Intruder is trying to use the system to gain popularity
- The Competition - Intruder is interested in the data on system, that could benefit him, financially or otherwise.
- The Borrowers - Intruder is interested in using system's resources for their own purposes
- The Leapfrogger - Intruder is only interested in the system to use it to get into other systems

B. Standardizing Security

Many security consultants and vendors agree upon the standard security model known as CIA, or Confidentiality, Integrity, and Availability[4].

The following describes the CIA model :

- Confidentiality — Sensitive information must be available only to a set of pre-defined individuals.
- Integrity — Information should not be altered in ways that make it incomplete or incorrect.
- Availability — Information should be accessible to authorized users any time that it is needed.

It is always recommended to create a simple policy for the system that can easily understood and followed.

A generally-accepted security policy phrase is :

" That which is not permitted is prohibited"

II. WORKSTATION HARDENING

A. *Physical Security*

1) BIOS Security

BIOS is the software at the lowest level that interacts with the x86 hardware.
Bootloaders gain booting related information from the BIOS. Securing BIOS will ensure that BIOS settings are not interfered with and the system is not bootable by an intruder.
Most of the system have a boot password.
Although it is not a hard-proof security measure, it will buy some time for taking precautionary steps and in some cases, might even leave sign of tampering.
Setting boot password can not be trusted in terms of security because have the provision of default password. Default BIOS passwords are set by the manufacturers to provide to customers in case of emergencies.
Another problem associated with it is that machine will not be able to boot unattended.

2) *Boot Loader Security*

Bootloader is a software which loads the OS kernel. LILO, GRUB legacy, GRUB 2, linload,etc are examples of bootloaders.
Securing bootloader is necessary to prevent illegal access to single user mode, bootloader console and an insecure operating system(in case more than one OS is present). Every bootloader's settings and security measures differ from one another.
GRUB is a popular bootloader used with all most all Linux distributions. GRUB can be given password by first generating a md5 hash of the password and then mentioning the hash in the GRUB configuration file.

sudo grub-md5-crypt

After running the above command type in a strong password. It will return a md5 hash encrypted password. Copy the hash and enter in the GRUB configuration file(/boot/grub/grub.conf) as:

password –md5 <paste-the-encrypted-string>

In the next boot an illegal access to the editor or the command interface will not be allowed without pressing *p* followed by the GRUB password.
Similar to as mentioned in BIOS security, the machine will not be able to boot unattended.

3) *Security of local devices*

The workplace must be secure by using surveillance cameras, alarm systems, biometrics and security guards for cheeking physical infiltration.
Devices such as webcam and microphones, that can be accessed remotely, must be secured, by switching them off or disconnecting them when not in use.

B. Software Maintenance

Software maintenance is extremely important to maintaining a secure system. It is vital to patch software as soon as it becomes available in order to prevent attackers from using known holes to infiltrate the system.

1) *Minimal Package Install*

It is a recommended practice to regularly check and remove unwanted packages from the system. It is necessary to install only the packages required and refrain from installing unnecessary packages. Unnecessary packages may contain vulnerabilities that can harm the system.
Nowadays, every Linux distribution come with an option of Minimal Install, which gives only few packages and services. User or administrator can add-on packages as needed. This ensures no unsafe package is preloaded into the system.

2) *Regular Updates*

Softwares products are continuously updated and patched to rectify small to intermediate to serious level bugs and vulnerabilities. Hence, packages must be regularly updated.
Also, kernel should be kept updated.

3) *Verify Packages*

All software packages are published through repositories, which support package signing. Package signature is an authentication technique to signify that the package is safe to install, and it has not been altered with, by an illegal source. Thus, it is a good practice to install signed packages through well-known repositories.

C. *System Auditing and System Logging*

Successful local or network intrusion on systems never necessarily leave clear evidence. It is necessary to build a well-configured logging and audit infrastructure in advance, that collects this evidence.
Firstly, the logging and audit configuration will allow to trace the malicious activity that has happened so that prompt actions are taken immediately. Secondly, it will allow to check for misconfiguration, which might leave the system vulnerable for future attacks.

1) *System Logging*

A logging infrastructure provides a framework for individual programs running on the system to report whatever events are considered peculiar.
Logging has the advantage of being compatible with a wide variety of client applications, reporting only information considered most important by each application, but the disadvantage that the information reported is not consistent between applications.
The syslog daemon can be configured to automatically send log data to a central syslog server, but this is typically sent unencrypted, allowing an intruder to view data as it is being transferred. There are syslog daemons available that encrypt the data as it is being sent, for example, rsyslog.
Rsyslog provides improvements such as connection-oriented transmission of logs, the option to log to database formats, and the encryption of log data en route to a central logging server. Rsyslog daemon can be configured by modifying the */etc/rsyslog.conf* configuration file.

Common Linux default log files name and usage:

a) */var/log/message* – Where whole system logs or current activity logs are available.
b) */var/log/auth.log* – Authentication logs.
c) */var/log/kern.log* – Kernel logs.
d) */var/log/cron.log* – Crond logs (cron job).
e) */var/log/maillog* – Mail server logs.
f) */var/log/boot.log* – System boot log.
g) */var/log/mysqld.log* – MySQL database server log file.
h) */var/log/secure* – Authentication log.
i) */var/log/utmp or /var/log/wtmp :* Login records file.
j) */var/log/yum.log:* Yum log files.

Log file rotation is an important aspect of logging. Without rotation, log files would grow and consume more and more disk space, eventually interfering with the system operation. By default, log files are rotated weekly and four archival copies are stored of each log. The settings can be modified from the log rotation configuration file */etc/logrotate.conf*

Software packages such as logcheck and logwatch make it easy to read and understand syslog reports. Logwatch, for example, is valuable because it provides a parser for the syslog entry format and a number of signatures for types of lines which are considered to be mundane or noteworthy.

2) *System Auditing*

An auditing infrastructure, on the other hand, reports each instance of certain low-level events, regardless of which program caused the event to occur.
Auditing has the advantage of being more comprehensive, but the disadvantage of reporting a large amount of information, most of which is uninteresting.
The Linux Audit system provides a way to track security-relevant information on the system. Based on pre-configured rules, Audit generates log entries to record as much information about the events that are happening on the system as possible.
The following list summarizes some of the information that Audit is capable of recording in its log files:

a) *Watching file access*
b) *Monitoring system calls*

c) *Recording commands run by the user*

d) *Recording security events*

e) *Running summary reports*

f) *Searching for events*

g) *Monitoring network access*

Auditd daemon can be controlled by several commands and files:

h) *auditctl:* to control the behavior of the daemon, adding rules

i) */etc/audit/audit.rules:* contains the rules and various parameters of the auditd daemon

j) *aureport:* generate report of the activity on a system

k) *ausearch:* search for various events

l) *auditspd:* the daemon which can be used to relay event notifications to other applications instead of writing them to disk in the audit log

m) *autrace:* this command can be used to trace a process, in a similar way as strace.

n) */etc/audit/auditd.conf:* configuration file related to the logging.

D. Password and Encryption

Passwords are the most important security features used today. It is important for all users to have secure passwords which can not be cracked easily.

Encryption is the best way of achieving security of data both in motion and at rest. There are all sorts of methods of encrypting data, each having its own set of characteristics. Encryption modifies the data which renders it useless and difficult for the intruders to decrypt and use it illegally.

1) Creating Strong Passwords

Password security is very important for the protection of user, workstation and network, since, password is the primary method employed to verify identity. Nowadays, with very fast and efficient password cracking softwares available, it has become imperative to keep a strong password handy.

Do's and dont's of creating a strong passwords [4]:

a) *Do not use only words or only numbers*

b) *Mix upper and lower case letters*

c) *Do not use direct or inverted recognizable words*

d) *Mix letters and numbers*

e) *Do not use foreign language words*

f) *Do not use hacker terminology*

g) *Include non-alphanumeric characters i.e special characters*

h) *Do not use personal information*

i) *Pick a password that can be remembered*

j) *Do not write down the password*

k) *Do not use the same password for all machines*

2) Password Aging

Password aging allows one to specify a time period for which a password is valid. After the time period has expired, the user will be forced to enter a new password. This has the benefit of ensuring passwords are changed regularly and that a password that is stolen, cracked, or known will have a time-limited value.

chage command is used for the above purpose :

chage -M 90 <username>

where -M option followed by a number, specifies the number of days the password is valid

Running the chage command without options will make it run in interactive mode.

chage <username>

3) Encrypting Data at rest

Mobile systems such as laptops, CD, DVD and similar devices put sensitive data at the risk of compromise. If the data is encrypted , it lowers the chances of the data being accessed.

File system encryption

Partition encrypted can be accomplished through Linux Unified Key Setup-on-disk-format (LUKS) technology. Encrypting a partition is easy at installation time but LUKS can also be configured post-installation.

Using kickstart (for automated installations) to create encrypted partitions :

part /my-part –fstype=ext4 –size=10000 --onpart=sdb2 –encrypted –passphrase=<secret-pass>

where /my-part is a logical volume to be encrypted on device sdb2

For encryption post-installation, follow the following steps :

a) Create a disk partition or a logical volume.

b) Encrypt the device and assign it a passphrase. (cryptsetup luksFormat /dev/my-vol)

c) Unlock the encrypted volume and assign it a logical name.(cryptsetup luksOpen /dev/my-vol secret)

d) Create a file system in the decrypted volume. (mkfs.ext4 /dev/mapper/secret)

e) Create a mount point of the filesystem to access its contents.

f) After working, unlock the file system and lock the encrypted volume (cryptsetup luksClose secret)

In the above steps, my-vol is the partition created and secret is the logical name assigned to it

4) Data in Motion

Data in motion is data that is being transmitted over a network. The biggest threats to data in motion are interception and alteration. Username and password must never be transmitted over a network without protection as it could be intercepted and used to gain access to sensitive information. Encrypting the network session ensures a higher security level for data in motion. Data in motion is particularly vulnerable to attackers because the attacker does not have to be near the computer in which the data is being stored.

Following are some tools used to secure data in motion :

a) SSH

SSH stands for Secure Shell and is a command interface and protocol for establishing secure connections between systems. OpenSSH is a free implementation of SSH.

OpenSSH a collection of tools including ssh, which replaces telnet and rlogin; scp, which replaces rcp; and sftp, a secure replacement for ftp. It also contains sshd (daemon), which is a SSH server, and ssh-agent, ssh-keygen, and ssh-add, which handle key generation and management for OpenSSH.

The standard TCP port for the SSH service is 22, however this can be changed. SSH operation can be configured by modifying the file /etc/ssh/sshd_config.

b) VPN

VPN's(Virtual Private Network) are a way to establish a "virtual" network on top of some already-existing network. This virtual network often is encrypted and passes traffic only to and from some known entities that have joined the network.

VPNs are very common and are simple to use and setup.

E. File and file system security

One of the defining features of Linux and other UNIX-like operating systems is that "everything is a file". The file system is a universal "name space" where everything is accessible. Linux systems contain a large number of files, so it is often time-consuming to ensure that
every file on a machine has exactly the permissions required, because security relies heavily on file and directory permissions. But adhering to some simple guidelines can help minimize risk to a lot of extent.

1) DISK PARTITIONING

Some system directories should be placed on separate and independent partitions (or logical volumes). This allows for better separation and protection of data.
The installer's default partitioning scheme creates separate partitions for /, /boot, and swap[3].

- The *root* file system, represented by a forward slash (/), is the top of the directory tree, and contains all the subsequent linux files and folders.
- /boot partition is the first partition that is read by the system during boot up. The boot loader and kernel images that are used to boot the system are stored in this partition. This partition should not be encrypted or password-protected.
- Swap partition is a reserved area of disk space that used to supplement the computer's RAM.

If possible, create separate partition or logical volume for the following :

a) /tmp: The /tmp directory is a world-writable directory used for temporary file storage.

b) /var : The /var directory is used by daemons and other system services to store frequently-changing data.

c) /var/log and /var/log/audit : these directories store system logs and audit logs respectively .

d) /home : Home directory of local users and users over the network.

2) Restrict mount options

System partitions can be mounted with certain options which limit what files on those partitions can do. These options are set in the file /etc/fstab, and can be used to make certain types of malicious behavior more difficult.

	Filesystem	Mount Option	
a)	*non-root loccal partitions*	*nodev*	
b)	*removable media partitions*	*nodev, noexec,*	*nosuid*
c)	*/tmp and /dev/shm*	*nodev, noexec,*	*nosuid*

- nodev mount option prevent files from being interpreted as character or block devices.
- noexec mount option prevents execution of binaries.
- nosuid mount option prevents setUID and setGID permissions from taking their normal effect

3) File System attributes

The ext4 and XFS file system have additional file attributes that can help secure a file system from being overwritten or deleted.
lsattr command displays the file attributes and chattr command modifies the file attributes. For chattr, + operator sets whereas – operator clear the specified file attribute to the given file.

File Attribute	Description
e	extent map
a	append only
i	immutable
S	synhronous update
j	journal

The immutable bit (i) can be used to prevent accidentally deleting or overwriting a file that must be protected. It also prevents someone from creating a hard link to the file[1].

F. File Permissions and access controls

Linux security is largely dependent on file permissions. Basic principle involves giving the least privilege possible to each non-root account, file, directory or file system.

Following are some good practices and procedures regarding permissions, access[3] :

1) Change permissions on passwd, shadow, group and gshadow files as :

 # chown root:root /etc/passwd /etc/shadow /etc/group /etc/gshadow
 # chmod 644 /etc/passwd /etc/group
 # chmod 400 /etc/shadow /etc/gshadow

2) All world-writable directories should have sticky bit set

chmod +t /dir

where /dir is a world writable directory

When the so-called sticky bit is set on a directory, only the owner of a given file may remove that file from the directory. Without the sticky bit, any user with write access to a directory may remove any file in the directory

3) Find unauthorized world-writable files and SUID/SGID system executables

- SUID and SGID files on the system are a potential security risk, and should be monitored closely. Because these programs grant special privileges to the user who is executing them, it is necessary to ensure that insecure programs are not installed. If the file does not require a setuid or setgid bit, then these bits should be removed.
- World-writable files, particularly system files, can be a security hole if an intruder gains access to the system and modifies them. Also, world-writable directories are dangerous, since they allow to add or delete files as one wishes. It is generally a good idea to remove global write access to a file when it is discovered.

4) Configure umask daemon

Configure the users file-creation umask to be as restrictive as possible. Normally the umask is set in /etc/profile, so it applies to all users on the system. Be sure to make root's umask 077, which will disable read, write, and execute permission for other users, unless explicitly changed using chmod.

5) Find and repair unowned files

Unowned files may also be an indication an intruder has accessed the system.

6) Integrity Checking

Integrity checking cannot prevent intrusions into the system, but can detect that they have occurred.

The integrety checkers run a number of checksums on all the important binaries and config files and compares them against a database of former, known-good values as a reference. Thus, any changes in the files will be flagged. Many integrity checking softwares are around such as tripwire and AIDE(advanced intruder detection environment)

G. Account and Access Control

Once an intruder gains shell access to account, it can use any file that the account has access to. Hence, high security must be enforced on all the privileged accounts to prevent unauthorized access.

Some guidelines to remember while configuring accounts :

1) Restrict root login access to system console

The */etc/securetty* file contains a list of terminals that root can login from. Nothing else must be added to this file. One should be able to login remotely as regular user account and then su if needed (so there is no need to be able to login directly as root).

2) Limit su access to the root account

The su command allows a user to gain the privileges of another user by entering the password for that users account. It is desirable to restrict the root user so that only known administrators are ever allowed to access the root account. This restricts password-guessing against the root account by unauthorized users or by accounts which have been compromised.

3) Block shell and login access for non-root system accounts

Using /etc/passwd, identify the system accounts. These will primarily be the accounts with UID numbers less than 500, other than root. For each identified system account , lock the account:
 # usermod -L acct and disable its shell:
 # usermod -s /sbin/nologin acct
where acct is the system account

4) Verify that No Accounts Have Empty Password Fields

If an account has an empty password, anybody may log in and run commands with the privileges of that account.

5) Verify that password hashes of all account are shadowed

The hashes for all user account passwords should be stored in the file /etc/shadow and never in /etc/passwd(which is readable by all users).

6) Verify that no non-root accounts have uid 0

7) Disable 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

H. Pluggable Authentication Modules (PAM)

Pluggable Authentication Modules, is a system which implements modular authentication for Linux programs. PAM is the framework which provides the system's authentication architecture and can be configured to minimize the system's exposure to unnecessary risk.
PAM looks in the directory /etc/pam.d for application-specific configuration information.
One very important file in /etc/pam.d is /etc/pam.d/system-auth, which is included by many other PAM configuration files and defines "default" system authentication measures.

Each PAM configuration file contains a group of directives formatted as follows:

<interface> <control flag> <name> < arguments>

Module Interface
Four types of PAM module interface are available.

- auth — This module interface authenticates use.
- account — It verifies that access is allowed.
- password — It is used for changing user passwords.
- session — It configures and manages user sessions.

Control Flag
All PAM modules generate a success or failure result when called. Control flags tell PAM what do with the result.

Some of the control flags are :

- required
- requisite
- sufficient
- optional

- include

Some important PAM modules are :

1) ***pam_cracklib*** PAM module provides strength checking for passwords. It performs a number of checks, such as making sure passwords are not similar to dictionary words, are of at least a certain length, are not the previous password reversed, and are not simply a change of case from the previous password. It can also require passwords to be in certain character classes. It is used for the password interface type.

2) ***pam_tally2*** PAM module provides the capability to lock out user accounts after a number of failed login attempts. It performs a check on the number of consecutive failed logins, after which it will lock the account. It also sets an unlock time.

3) ***pam_deny.so*** PAM module denies access to a service.

4) ***pam_limits.so*** PAM module is used to place limits on users logging on to the system. Limits can be placed on user or a group and can be soft(can be exceeded) or hard(must not be exceeded).The pam_limits.so module is controlled by a configuration file called limits.conf that is located in /etc/security. PAM provides various types of limits such as **nproc, maxlogins, cpu, stack, data, priority**, etc. It is used for the session interface type.

5) ***pam_time*** PAM module allows to control when and where from users can log onto the system. It is defined as an account interface type. The module reads the access rules from the /etc/security/time.conf file. It is used for the account interface type.

6) ***pam _access*** PAM module allows an administrator to customize access control based on login names, host or domain names, or IP addresses. The module reads the access rules from the /etc/security/access.conf file.

III. CONCLUSION

Amount of effort required may differ from securing home system to that of servers operating within a large organization. But, Linux can indeed provide a safe, secure platform, better in contrast to other major operating systems. Being open source by nature, Linux is a platform where many developers can code and test the system. Large number of bugs, security vulnerabilities and patches are reported and handled simultaneously, making it difficult to operate malicious intrusion and unauthorized attempts. Employing some easy practices and procedures, and being cautious while operating the machine, one can definitely achieve and sustain a secure environment.

IV. ACKNOWLEDGMENT

This work could not have been possible, had there not have existed numerous open source articles, documents, papers, and guides relating to GNU/Linux. Special thanks to Red Hat Inc. for providing specific documentation on security and related topics. Also, The Linux Documentation Project, which has been providing rich quality guides and HOW-TO's from over a decade, with open access for all, deserves special mention.

V. REFERENCES

[1] J. Turnbull, “Hardening Linux”, Apress, 2005
[2] K. Fenzi, Dave Wreski, “Linux Security HOWTO ”, v2.3, 22 January 2004, http://www.tldp.org/HOWTO/Security-HOWTO/
[3] National Security Agency, “Guide to the Secure Configuration of Red Hat Enterprise Linux 5 ”, Revision 4.1, February 28, 2011, https://www.nsa.gov/ia/_files/os/redhat/rhel5-guide-i731.pdf
[4] M. Prpič, T. Čapek, S. Wadeley, Y. Ruseva, M. Svoboda, R. Krátký, “Red Hat Enterprise Linux 6 Security Guide”, Red Hat, Inc., 2013,https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security_Guide/