Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 8; October, 2014 pp. 102-107
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

Advanced End-to-End Linux Security for Desktop and Production Systems

Ethirajan D.[1], **Prema S.**[2], **Ramesh R.**[3]

[1, 2]Senior Engineer, C-DAC, Chennai, [1]ethirajand@cdac.in, [2]premas@cdac.in
[3]Senior Consultant, Infosys, Chennai, ramesh_ramamoorthy@infosys.com

Abstract: ***In the ever-changing world of global data communications, inexpensive Internet connections and fast-paced software development, security is becoming more and more of an issue. Security is now a basic requirement because global computing is inherently insecure. Even other users on your system may maliciously transform your data into something you did not intend. This paper explores both theoretical and practical options for all the Security features and Customization for both the Linux desktop and Server machine.***

1. INTRODUCTION

Nowadays there is a move in the Organizations towards Linux desktop and Server for security reasons. These organizations mostly depend on basic Firewall for their Internet and Intranet connections. But that only is not sufficient to make the data to be secure within a network. Unauthorized access to the system may be obtained by intruders, also known as "crackers". There are several security policies listed for Linux to get rid of the issues and viruses, but no single OS will implement all the security needs.

2. NEED OF SECURITY

Linux has some inherent advantages over Windows when it comes to security. Hackers, rather like gamblers, use the laws of odds and averages in their endeavours to find vulnerable computer systems to break into. They will typically target the types of systems that have the most security vulnerabilities. They will also mostly focus attention on areas where there are the most opportunities for unprotected systems- in other words the types are system that are most common on the internet. In the given case Linux, being opensource is a good choice to secure itself against vulnerabilities and attacks.

3. DISK AND PARTITION SECURING

As the first start the Hard disk where the OS to be installed is to secured. This needs to be done during the installation phase itself. The harddisk to be protected from any external attacks or data thefts easily. This can be implemented by

- LVM Encrypted harddisk
- Partition Policies
- Grub password

3.1 LVM Disk Encryption

LVM is a logical volume manager for the Linux kernel that manages disk drives and similar mass-storage devices. LVM is commonly used for the following purposes:

- Managing large hard disk farms by allowing disks to be added and replaced without downtime or service disruption, in combination with hot swapping.
- On small systems (like a desktop at home), instead of having to estimate at installation time how big a partition might need to be in the future, LVM allows file systems to be easily resized later as needed.
- Performing consistent backups by taking snapshots of the logical volumes.
- Creating single logical volumes of multiple physical volumes or entire hard disks (somewhat similar to RAID 0), allowing for dynamic volume resizing.
- the Ganeti solution stack relies on the Linux Logical Volume Manager .

LVM can be considered as a thin software layer on top of the hard disks and partitions, which creates an abstraction of continuity and ease-of-use for managing hard drive replacement, re-partitioning, and backup.

Applying encryption is fast when it is done upon creation, since the initial contents of the partition are ignored, they are not encrypted; only new data will be encrypted as it is written. However, when applying encryption on an *existing* volume (as is typical of TrueCrypt) requires reading, encrypting and writing back all used data sectors; this includes sectors which *were* previously in use, even if they are not in use right now, because they may contain excerpts of some files which were later on copied around. A basic PC will be able to encrypt data at more than 100 MB/s, with AES, using a single core (my underpowered laptop achieves 120 MB/s); with recent x86 cores offering the AES-NI instructions, 1 GB/s is reachable. Thus, the CPU can keep pace with the disk, and, most of the time, the user will not notice any slowdown.

The said customised Linux version enforces a a Full disk LVM Partitioned installation with a stronger Passphrase to encrypt the harddisk. Separate partitions for /usr, /var, /home and / are created thus ensuring separation of data and system binaries.

3.2 Partition Policies

On NFS filesystems, configure /etc/exports with the most restrictive access possible. Do not use any wild cards.
/var/log/wtmp and /var/run/utmp contain the log-in attempts for all users. world-writable directories are dangerous as they allow an intruder to add/delete files. You must locate the world-writable files on your system and make sure that you know why they are writable. We have to locate such files by using the following command:

```
$ find / \( -nouser -o -nogroup \) -print
```
Use this command to locate and find the .rhosts file
```
$ find /home -name .rhosts -print.
```

We have to make sure to setup a separate user-writable data, non-system data, and rapidly changing run-time data to their own partitions. This can be achieved by setting Set nosuid, noexec, nodev mount options in /etc/fstab on ext3 / ext4 partitions such as /tmp

3.3 Grub password

If a system is not kept in a locked data center, and as an alternative to using any password protection mechanism built into the BIOS, you can add a degree of protection to the system by requiring a valid password be provided to the GRUB boot loader. Password protecting GRUB access prevents unauthorized users from entering single user mode and changing settings at boot time.

To configure a GRUB password, Use the following command to generate the MD5 hash of your password:

```
# /sbin/grub-md5-crypt
Password: clydenw
Retype password: clydenw
$1$qhqh.1$7MQxS6GHg4IlOFMdnDx9S.
```

Edit /boot/grub/grub.cfg, and add a password entry below the timeout entry near the top of the file, for example:

```
timeout=5
password --md5 <pwhash>
```

where pwhash is the hash value that **grub-md5-crypt** returned. Now-onwards the The GRUB menu does not allow access to the editor or command interface (bootup screen) without first pressing 'p' followed by the GRUB password.

4. SECURING KERNEL

The Linux kernel is the core of all Linux systems. If any malicious code controls or damages any part of the kernel, then the system can get severely damaged, files can be deleted or corrupted, private information can be stolen, etc. Linux has may security features and programs, but only the Linux Security Modules (LSM) and other kernel security are important to be implemented.

4.1. Linux Security Modules

The Linux Security Modules (LSM) API implements hooks at all security-critical points within the kernel. A user of the framework (an “LSM”) can register with the API and receive callbacks from these hooks. All security-relevant information is safely passed to the LSM, avoiding race conditions, and the LSM may deny the operation. This is similar to the Netfilter hook-based API, although applied to the general kernel. The LSM API allows different security models to be plugged into the kernel—typically access control frameworks.To ensure compatibility with existing applications, the LSM hooks are placed so that the Unix DAC checks are performed first, and only if they succeed, is LSM code invoked.

4.1.1 SELinux

Security Enhanced Linux (SELinux) is an implementation of fine-grained Mandatory Access Control (MAC) designed to meet a wide range of security requirements, from general purpose use, through to government and military systems which manage classified information. MAC security differs from DAC in that the security policy is administered centrally, and users do not administer policy for their own resources. This helps contain attacks which exploit userland software bugs and misconfiguration. In SELinux, all objects on the system, such as files and processes, are assigned security labels. All security-relevant interactions between entities on the system are hooked by LSM and passed to the SELinux module, which consults its security policy to determine whether the operation should continue. The SELinux security policy is loaded from userland, and may be modified to meet a range of different security goals. Many previous MAC schemes had fixed policies, which limited their application to general purpose computing.

4.1.2 Grsecurity

grsecurity is a set of patches for the Linux kernel which emphasizes security enhancements using RBAC – Role Based

Access Control. RBAC is intended to restrict access to the system further than what is normally provided by UNIX access control lists, with the aim of creating a fully least-privilege system, where users and processes have the absolute minimum privileges to work correctly and nothing more. This way, if the system is compromised, the ability of the attacker to damage or gain sensitive information on the system can be drastically reduced. RBAC works through a collection of roles. Each role can have individual restrictions on what they can or cannot do, and these roles and restrictions form an access policy, which can be amended as needed.

4.1.3 AppArmor

AppArmor is a MAC scheme for confining applications, and was designed to be simple to manage. Policy is configured as application profiles using familiar Unix-style abstractions such as pathnames. It is fundamentally different to SELinux and Smack in that instead of direct labeling of objects, security policy is applied to pathnames. AppArmor also features a learning mode, where the security behavior of an application is observed and converted automatically into a security profile.

4.1.4. Audit

The Linux kernel features a comprehensive audit subsystem, which was designed to meet government certification requirements, but also actually turns out to be useful. LSMs and other security components utilize the kernel Audit API. The userland components are extensible and highly configurable.

Audit provides tools that help the administrative user extract specific types of audit events, audit events for specific users, audit events related to specific file system objects or audit events within a specific time frame .It's responsible for writing audit records to the disk. Linux audit files to see who made changes based on program, database files and system calls. Audit logs are useful for analyzing system behavior, and may help detect attempts at compromising the system. To enable audit log level in kernel, append the audit=1 parameter to your kernel boot line, either in the /etc/grub.conf file or on the GRUB menu at boot time.

This is an example of a full audit log entry when httpd is denied access to ~/public_html because the directory is not labeled as Web content. Notice that the time and serial number stamps in the audit(...) field are identical in each case. This makes it easier to track a specific event in the audit logs:

```
Aug 25 18:45:56 hostname kernel:
audit(110580234236.075:232423892): \
 avc: denied { getattr } for pid=2239 exe=/usr/sbin/httpd \
 path=/home/boss/public_html dev=sda7 ino=9212345 \
 scontext=user_u:system_r:httpd_t \
 tcontext=system_u:object_r:user_home_t tclass=dir
```

4.1.5 TOMOYO

The TOMOYO module is another MAC scheme which implements path-based security rather than object labeling.It's also aimed at simplicity, by utilizing a learning mode similar to AppArmor's where the behavior of the system is observed for the purpose of generating security policy.

What's different about TOMOYO is that what's recorded are trees of process invocation, described as “domains”. For example, when the system boots, from init, as series of tasks are invoked which lead to a logged in user running a shell, and ultimately executing a command, say ping. This particular chain of tasks is recorded as a valid domain for the execution of that application, and other invocations which have not been recorded are denied.

5. FILE SYSTEM SECURITY

A solid house needs a solid foundation, otherwise it will collapse. In Linux's case this is the ext (EXTended, version) filesystem. Files and directories have permission sets for the owner of the file, the group associated with the file, and all other users for the system. The filesystem can be secured by applying

- Access control list
- Extended Attributes
- File system limits and Controls

5.1 Access Control List

Access Control List (ACL) provides an additional, more flexible permission mechanism for file systems. It is designed to assist with UNIX file permissions. ACL allows you to give permissions for any user or group to any disc resource. However, these permission sets have limitations.

When a subject requests an operation on an object in an ACL-based security model, the operating system first checks the ACL for an applicable entry to decide whether the requested operation is authorized. A key issue in the definition of any ACL-based security model is determining how access control lists are edited, namely which users and processes are granted ACL-modification access.

The Linux filesystem is to be mounted with ACL enabled through /etc/fstab

LABEL=/home /home ext3 acl 1 2

If an ext3 file system is accessed via Samba and ACLs have been enabled for it, the ACLs are recognized because Samba has been compiled with the --with-acl-support option. No special flags are required when accessing or mounting a Samba share.

5.2 Extended Attributes

Extended file attributes is a file system feature that enables users to associate computer files with metadata not interpreted by the filesystem, whereas regular attributes have a purpose strictly defined by the filesystem (such as permissions or records of creation and modification times). Unlike forks, which can usually be as large as the maximum file size, extended attributes are usually limited in size to a value significantly smaller than the maximum file size.

In a ext3 / ext4 filesystem Extended Attributes should be enabled in the kernel configuration. Any regular file or directory may have extended attributes consisting of a name and associated data. The name must be a null-terminated string prefixed by a name-space identifier and a dot character. Currently, four name-spaces exist:

- User
- Trusted
- Security
- System

Extended attributes can be accessed and modified using the 'attr' in a ext3 / ext4 filesystem.

5.3 File-system limits and Controls

cgroups is a Linux kernel feature to limit, account, and isolate resource usage (CPU, memory, disk I/O, etc.) of process groups. A control group is a collection of processes that are bound by the same criteria. These groups can be hierarchical, where each group inherits limits from its parent group. The kernel provides access to multiple controllers (subsystems) through the cgroup interface. For instance, the "memory" controller limits memory use, "cpuacct" accounts CPU usage, etc.

Control groups can be
used in multiple ways:

- By accessing the cgroup virtual file system manually.
- By creating and managing groups on the fly using tools like cgcreate, cgexec, and cgclassify (from libcgroup).
- Through the "rules engine daemon" that can automatically move processes of certain users, groups, or commands to cgroups as specified in its configuration.

6. FILE-SYSTEM INTEGRITY

The kernel's integrity management subsystem is used to maintain the integrity of files on the system. The Integrity Measurement Architecture (IMA) component performs runtime integrity measurements of files using cryptographic hashes, comparing them with a list of valid hashes. The list itself is verified via an aggregate hash stored in the TPM. Measurements performed by IMA may be logged via the audit subsystem, and also used for remote attestation, where an external system verifies their correctness.

IMA may also be used for local integrity enforcement via the Appraisal extension. Valid measured hashes of files are stored as extended attributes with the files, and subsequently checked on access. These extended attributes (as well as other security-related extended attributes), are protected against offline attack by the Extended Verification Module (EVM) component, ideally in conjunction with the TPM. If a file has been modified, IMA may be configured via policy to deny access to the file. The Digital Signature extension allows IMA to verify the authenticity of files in addition to integrity by checking RSA-signed measurement hashes.

6.1 Tripwire

Tripwire is a file integrity checker for linux systems. If Tripwire detects that a monitored file has been changed, it notifies the system administrator via email. Because Tripwire can positively identify files that have been added, modified, or deleted, it can speed recovery from a break-in by keeping the number of files which must be restored to a minimum.

7. TCP / IP WRAPPERS

To determine if a client machine is allowed to connect to a service, TCP wrappers reference the following two files, which are commonly referred to as hosts access files:
/etc/hosts.allow
/etc/hosts.deny

When a client request is received by a TCP wrapped service, it takes the following basic steps:

- The Service references /etc/hosts.allow - The TCP wrapped service sequentially parses the /etc/hosts.allow file and applies the first rule specified for that service. If it finds a matching rule, it allows the connection. If not, it moves on to step 2.
- The Service references /etc/hosts.deny. - The TCP wrapped service sequentially parses the /etc/hosts.deny file. If it finds a matching rule is denies the connection. If not, access to the service is granted

8. USER SECURITY

The User level security is to be applied on Account level and Services level.

8.1 Account level Security

The Account level security greatly depends on the strength of the user password. PAM – Pluggable Authentication Modules are at the core of the user authentication in any linux operating system. The security modules thus need to be incorporated in the PAM modules only. The strength of the user password of a secured linux system should satisfy below criteria. Password should be combination of

- Upper case
- Lower case
- Numerals
- Special Characters
- 10>length>40

Enable the above said criteria in /etc/pam.d/

```
$ cat login
# PAM configuration for login
auth    requisite pam_securetty.so
auth    required  pam_nologin.so
auth    required  pam_env.so
auth    required  pam_unix.so nullok
account  required  pam_unix.so
session  required  pam_unix.so
session  optional  pam_lastlog.so
password   required   pam_unix.so nullok obscure min=10
      max=40
```

8.2 Securing SSH

Securing the Root account is most important in any Linux system, because on accessing the Root account leads to damage the whole file-system to a greater extent. The remote login of Root account is to be disabled for a secured Linux system. The X11 forwarding of a linux machine leads to easier attacks.
In /etc/ssh/sshd_config, keep

```
#     X11 Forwarding NO
#     PubkeyAuthentication NO
```

always. Unwanted services and ports to be closed to the maximum by restricting the user access to limited applications and programs.

9. NETWORK SECURITY

Linux has a very comprehensive and capable networking stack, supporting many protocols and features. Linux can be used both as an endpoint node on a network, and also as a router, passing traffic between interfaces according to networking policies.

Netfilter is an IP network layer framework which hooks packets which pass into, through and from the system. Kernel-level modules may hook into this framework to examine packets and make security decisions about them.

Iptables is one such module, which implements an IPv4 firewalling scheme, managed via the userland iptables tool. Access control rules for IPv4 packets are installed into the kernel, and each packet must pass these rules to proceed through the networking stack.Also implemented in this codebase is stateful packet inspection and Network Access Translation (NAT). Firewalling is similarly implemented for IPv6.

The networking stack also includes an implementation of IPsec, which provides confidentiality, authenticity, and integrity protection of IP networking. It can be used to implement VPNs, and also point to point security

10. SNORT – INTRUSION DETECTION

Snort's open source network-based intrusion detection system (NIDS) has the ability to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks. Snort performs protocol analysis, content searching, and content matching.

Snort runs as a daemon and it should be attached to the current active NIC card for monitoring the network. First put the active device in promiscuous mode and set it to not to respond to any arp packcts.

```
# ifconfig eth1 up –arp
# service snort start
# snort –i eth1 –v
```

The above commands monitors the network and logs the network attacks and event like ping, ssh and telnet events on the machine.

11. CONCLUSION

This paper explores all type of Security levels that need to be incorporated in a Linux desktop and Server machines. The various levels of security ranging from the harddisk, filesystem, OS and network level are discussed elaborately and the components that need to be integrated in a standard secured Linux operating system are discussed throughly.

The paper has covered the Linux security in both at a very high-level and low-level. The requirements and the necessity of the security in Linux have been driven both by external changes, such as the continued growth of the Internet and the increasing value of information stored online, as well as the increasing scope of the Linux user base.

Ensuring that the security features of the Linux continue to meet such a wide variety of requirements in a changing landscape is an ongoing and challenging process.

12. ACKNOWLEDGEMENTS

This work was supported in part by NRCFOSS – National Resource Center for Free and OpenSource Software by C-DAC, Center for Development of Advanced Computing – Chennai.

REFERENCES

[1] Bob Toxen, “Real-world Linux Security: Intrusion, Prevention, Detection, and Recovery “, Prentice Hall

[2] Daniel Barrett, Richard Silverman, Robert Byrnes, “Linux Security Cookbook”, O'Reilly Media

[3] Scott Mann, Ellen L. Mitchell, Mitchell Krell, “Linux System Security: An Administrator's Guide to Open Source Security Tools”, Prentice Hall

[4] James Lee, George Kurtz, “Hacking Linux Exposed: Linux Security Secrets & Solutions”, Osborne/McGraw-Hill