Chapter 42 Local System Security - Notes

42.2 Introduction

Essential task of any system administrator is to secure the system(s) against both internal and external threats. Work begins with designing proper security policy, crafted to defend against expected and unexpected threat vectors. In addition, good system hygiene has to be practiced; systems need to be maintained and upgraded in timely fashion, as well as physically protected from usurpers. Furthermore, sensible policies have to guarantee only appropriate users have potentially dangerous privileges, and only those which are absolutely needed.

42.3 Learning Objectives:

- Assess system security risks.
- Fashion and implement sound computer security policies and procedures.
- Efficiently protect BIOS and the boot loader with passwords.
- Use appropriate **mount** options, **setuid** and **setgid** to enhance security.

42.4 Local System Security

Computers are inherently insecure and need protection from people who would intrude in on or attack them. This can be done to simply harm the system, deny services, or steal information.

No computer can ever be absolutely secure. All we can do is slow down and/or discourage intruders so that they either go away and hunt for easier targets, or so we can catch them in the act and take appropriate action.

Security can be defined in terms of the system's ability to regularly do what it is supposed to do, integrity and correctness of the system, and ensuring that the system is only available to those authorized to use it.

The biggest problem with security is to find that appropriate mix of security and productivity; if security restrictions are tight, opaque, and difficult, especially with ineffective measures, users will circumvent procedures.

The four areas we need to protect include physical, local, remote, and personnel. In this section, we will not concentrate on network security, we will concentrate on local factors.

42.5 Creating a Security Policy

Important to create and publicize to you organization a clear security policy. It should:

- Be simple and easy to understand
- Get constantly updated
- Be in the form or a written document in addition to online documentation if needed
- Describe both policies and procedures
- Specify enforcement actions
- Specify actions to take in response to a security breach.

Policies should be generic and not hard to grasp as that makes them easier to follow. Must safeguard the data that needs protection, deny access to required services and protect user privacy.

These policies should be updated on a regular basis; policies need to change as requirements do. Having an out of date policy can be worse than having none.

42.6 What to Include in the Policy

A security policy should include methods of protecting information from being read or copied by unauthorized personnel. Should also include protection of information from being altered or deleted without the permission of the owner. All services should be protected so they are available and not degraded in any manner without authorization.

Essential aspects to cover:

- Confidentiality
- Data Integrity
- Availability
- Consistency
- Control
- Audit

Should make sure that data is correct and the system behaves as it is expected to do. There should be processes in effect to determine who is given access to your system. Human factor is the weakest link in the security chain; it required the most attention through constant auditing.

42.7 What Risks to Assess

Risk analysis is based on the following three questions:

- What do I want to protect?
- What am I protecting against?
- How much time, personnel, and money is needed to provide adequate protection?

Must know what you are protecting and what are you protecting against in order to determine how to protect your systems. This allows you to then plan policies and procedures to protect your systems.

This is the first step taken in constructing a computer security policy. It is a pre-requisite for planning and then enforcing policies and procedures to protect your systems.

42.8 Choosing a Security Philosophy

Two basic philosophies found in use in most computing environments:

- Anything not expressly permitted is denied.
- Anything not expressly forbidden is permitted.

You should decide which philosophy is best for your organization.

First choice is tighter: user is allowed to do only what is clearly and explicitly specified as permissible without privilege. This is the most commonly used philosophy.

Second choice builds a more liberal environment where users are allowed to do anything except what is expressly forbidden. Implies a high degree of assumed trust and is less often deployed for obvious reasons.

42.9 Some General Security Guidelines

Some general guidelines to remember when developing security philosphies:

1. **The human factor is the weakest link**

   You must educate your users and keep them happy. The largest percentage of break-ins are internal and are not often malicious.

2. **No computing environment is invulnerable**

   The only secure system is one that is not connected to anything, locked in a secure room, and turned off.

3. **Paranoia is a good thing**

   Be suspicious, be vigilant, and persevere when securing a computer. It is an ongoing process which must be constantly paid attention to. Check process, users, and look for anything out of the ordinary.

Users should never put the current directory in their path, i.e., do not do something in `~/.bashrc` like `Path=./:$PATH`

This is a significant security risk; a malicious person could substitute for a program with one of the same name that could do harmful things. Just think of a script names **ls** that contains just the line:

```
/bin/rm -rf $HOME
```

If you were to go to the directory that contains this file and type **ls**, you would wipe out your home directory!

42.10 Updates and Security

It is crucial to pay attention to your Linux distributor's updates and upgrades and apply them as soon as possible.

Any system which is not fully updated should be considered vulnerable.

Most attacks exploit known security holes and are deployed in the time period between revelation of a program and patches being applied. **Zero Day** exploits are actually much rarer, where an attacked uses a security hole that either has not been discovered yet or for which a fix has not been released.

System administrators are sometimes reluctant to apply such fixes immediately upon release, based on negative experiences with proprietary operating system vendors who can cause more problems with fixes than they solve. However, in Linux such security **regressions** are extremely rare, and the danger of delaying applying a security patch is probably never justifiable.

42.11 Hardware Accessibility and Vulnerability

Any time hardware is physically accessible security can be compromised by:

- Key logging: Recording the real time activity of a computer user including the keys they press. The captured data can either be stored locally or transmitted to remote machines.
- Network sniffing: Capturing and viewing the network packet level data on your network.
- Booting with a live or rescue disk.
- Remounting and modifying disk content.

Physical access to a system makes is possible for attackers to easily leverage several attack vectors, in a way that makes all operating system level recommendations irrelevant.

Thus, security policy should start with requirements on how to properly physical access to servers and workstations.

42.12 Hardware Access Guidelines

Necessary steps include:

- Locking down workstations and servers
- Protecting your network links against access by people you do not trust
- Protecting your keyboards where passwords are entered to ensure the keyboards cannot be tampered with
- Configuring password protection of the BIOS in such a way that the system cannot be booted with a live or rescue CD/DVD or USB key.

For single user computers and those in a home environment, some of the above features (like preventing booting from removable media) can be excessive, and you can avoid implementing them. However, if sensitive information is on your system that requires careful protection, either it should not be there, or it should be better protected by following the above guidelines.

42.13 Protection of BIOS

BIOS is the lowest level of software that configures or manipulates your system. The boot loader access the BIOS to determine how to boot up the machine. The BIOS:

- Is the lowest level of security
- Should be protected by use of a password
- Should be updated and current.

Setting a BIOS password protects against unauthorized persons changing the boot options to gain access to your system. However, only matters if someone can gain physical access to machine, as it requires a local presence.

Also, generally recommended that BIOS be kept patched to the latest version of firmware. However, most BIOS updates have nothing to do with security, and system administrators have also been instructed to apply new BIOS only with care, as incompetent BIOS code has always been a plague, and unnecessary updates can render a system useless.

42.14 Protecting the Boot Loader with Passwords

Can secure the boot process with a secure password to prevent someone from bypassing the user authentication step. This can work in conjunction with password protection for the BIOS.

Note: using a bootloader password alone will stop user from editing the bootloder configuration during boot process. Will not prevent a user from booting from an alternative boot media such as optical disks or pendrives. Thus, should be used with a BIOS password for full protection.

For older GRUB version 1, relatively easy to set password for **grub**, but for dominant GRUB version 2, things more complicated. However, have more flexibility and can do things like setting individual user-specific passwords (which can be the normal login ones).

Once again, should not edit `grub.cfg` directly. Rather, edit system configuration files in `/etc/grub.d` and then run **update-grub** or **grub2-mkconfig** and save the new configuration file.

One explanation of this can be found on the Grub2/Passwords webpage at Ubuntu.

42.15 Using Secure Mounting Options

When filesystem mounted, either at the command line with a **mount** command, or automatically by inclusion in `/etc/fstab`, various options can be specified to enhance security:

- `nodev`

  Do not interpret character or block special devices on the filesystem.

- `nosuid`

  The **set-user-identifier** or **set-group-identifier** bit are not to take effect (Will shortly discuss **setuid** and **setgid**)

- `noexec`

  Restrict direct execution of any binaries on the mounted filesystem.

- `ro`

  Mount filesystem in **read-only** mode as in:

```
$ mount -o ro,noexec,nodev, /dev/sda2 /mymountpt
```

or on `/etc/fstab`:

```
/dev/sda2 /mymountpt ext4 ro,noexec,nodev 0 0
```

42.16 setuid and setgid

Normally, programs run with privileges of user who is executing the program. This means that no matter who actually owns the binary executable that is running, the process still has constricted privileges.

Occasionally, may make sense to have normal users have expanded capabilities they would not normally have, such as ability to start or stop a network interface, or edit a file owned by superuser.

By setting the **setuid** (**set u**ser **ID**) flag on an executable file, one modified this normal behavior by giving the program the access rights of the **owner** rather than the **user** of the program.

Furthermore, one can also set the **setgid** bit for the process runs with the privileges of the group that owns the file rather than those of the one running it.

Should emphasize that this is generally a **bad idea** and is to be avoided in most circumstances. Often better to write a **daemon** program with lesser privileges for this kind of use. Some recent distributions have actually disabled this ability entirely.

By default, when a file is created in a directory, it is owned by the user and group of the user that created it. Using the **setgid** setting on the directory changes this so that files created in the directory are group owned by the group owner of the directory. Allows you to created a shared directory in which a group of users can share files.

42.17 Setting the setuid/setgid Bits

Simply done by:

```
$ chmod u+s somefile
$ chmod g+s somefile
```

where fist example does **setuid** and second the **setgid** operation.

For directories, setting group bits has very different effect; used to create a shared directory, as in:

```
$ chmod g+s somedir
```

Files created in this directory are group owned by the group owner of the directory.

Note: cannot effectively change the **setuid** on a shell script file; in fact, won't do anything unless you actually change the **setuid** bit *on the shell*, which would be a terrible security hole. Can only do this on executable binary programs.

##

Back to top

Previous Chapter - Table of Contents - Next Chapter