Linux Security Administrator's Guide: Host Security

4. Host Security

The next thing to take a look at is the security in your system against attacks from local users. Did we just say *local* users? Yes!

Getting access to a local user is one of the first things that system intruders attempt, while on their way to exploiting the root account. With lax local security, they can then ``upgrade'' their normal user access to root access using a variety of bugs and poorly setup local services. If you make sure your local security is tight, then the intruder will have another hurdle to jump.

Local users can also cause a lot of havoc with your system even (especially) if they really are who they say they are. Providing accounts to people you don't know or have no contact information for is a very bad idea.

4.1 Delete Unnecessary Packages

If you know you are not going to use some particular package, you can also delete it entirely. `/bin/rpm -e <packagename>` under the Red Hat distribution will erase an entire package. Under debian `/bin/dpkg` likely does the same thing.

If you are configuring a new machine to be installed on the network, only initially install the packages that are necessary for its normal operation.

Removing unnecessary setuid and setgid binaries should be a priority. You should always be aware of which ones are available on your system. You can do this using the following:

```
user@myhost$ find / -type f -perm +6000
```

This will find all the setuid and setgid binaries on your system. You can find more about the setuid and setgid permissions in the *File System Security* section.

4.2 Default System Configuration

The default Linux system installation is generally far more secure than other operating systems, due to not having to conform to older standards and traditions.

However, installing any operating system, and connecting it to the network is a foolish idea. Many system defaults are still more lenient than is intended to be used in a production network system.

Spend some time to customize it to your environment. Be sure to follow these guidelines, as well as the ones refered to herein, including disabling any services that are not necessary, configuring auditing, etc, before connecting a machine to the network.

4.3 Make a Full Backup of Your Machine

Discussion of backup methods and storage is beyond the scope of this document, but a few words relating to backups and security:

If you have less than 650Mb of data to store on a partition, a CD-R copy of your data is a good way to go (as it's hard to tamper with later, and if stored properly can last a long time). Tapes and other re-writable media should be write protected as soon as your backup is complete and verified to prevent tampering. Make sure you store your backups in a secure off line area. A good backup will ensure that you have a known good point to restore your system from.

A six-tape cycle is an easy one to maintain. This includes four tapes for during the week, one tape for even Friday's, and one tape for odd Friday's. Perform an incremental backup every day, and a full backup on the appropriate Friday tape. If you make some particular important changes or add some important data to your system, a backup might well be in order.

4.4 Backup Your Red Hat or Debian File Database

In the event of an intrusion, you can use your RPM database like you would use `tripwire`, but only if you can be sure it too hasn't been modified. You should copy the RPM database and */bin/rpm* executable to a floppy or Zip disk, and keep this copy off-line at all times. The Debian distribution likely has something similar. (Would someone fill me in here, until I get Debian re-installed?) See the section on Integrity Checking for further information, and instructions on how to do this.

4.5 Make Use of Your System Accounting Data

It is very important that the information that comes from your system accounting files has not been compromised, and is installed and configured properly. Making the files in `/var/log`, `/var/run/utmp`, and `/var/log/wtmp` readable, and writable only by the root user is a good start. Knowing which tools to use at what times is a good practice.

You can find more information on this in the *User and System Accounting* section.

4.6 Apply All New System Updates

Most Linux users install from a CDROM. Due to the fast paced nature of security fixes, new (fixed) programs are always being released. Before you connect your machine to the network, it's a good idea to check with your distribution's ftp site (ftp.Red Hat.com for example) and get all the updated packages since you received your distribution CDROM. Many times these packages contain important security fixes, so it's a good idea to get them installed.

4.7 Creating New Accounts

You should make sure to provide user accounts with only the minimal requirements for the task they need to do. If you provide your secretary, or another general user, with an account, you might want them to only have access to a word processor or drawing program, but be unable to delete data that is not his or hers.

Several good rules of thumb when allowing other people legitimate access to your Linux machine:

- Limit access privileges given to new users.
- Be aware when/where they login from, or should be logging in from.

- Make sure to remove inactive accounts
- The use of the same user-ID on all computers and networks is advisable to ease account maintenance, as well as permit easier analysis of log data (but I'm sure someone will dispute this). However, it's practically essential if using NFS. There are several other protocols that use UIDs for local and remote access as well.
- The creation of group user-IDs should be absolutely prohibited. User accounts also provide accountability, and this is not possible with group accounts.
- Be sure shadow passwords are enabled. See the *Password Security* section for more information.
- Regularly audit user accounts for invalid or unused accounts, expired accounts, etc.
- Check for repeated login failures
- Be sure to enable quotas, to prevent denial of service attacks involving filling disk partitions, or appending exploits to group-writable files.
- Disable group accounts, and unused system accounts, such as `sys` or `uucp`. These accounts should be locked, and given non-functional shells.

Many local user accounts that are used in security compromises are ones that have not been used in months or years. Since no one is using them they provide the ideal attack vehicle.

4.8 Root Security

The most sought-after account on your machine is the superuser account. This account has authority over the entire machine, which may also include authority over other machines on the network. Remember that you should only use the root account for very short specific tasks and should mostly run as a normal user. Running as root all the time is a very very very bad idea.

Several tricks to avoid messing up your own box as root:

- When doing some complex command, try running it first in a non destructive way...especially commands that use globbing: e.g., you are going to do a `rm foo*.bak`, instead, first do: `ls foo*.bak` and make sure you are going to delete the files you think you are. Using echo in place of destructive commands also sometimes works.
- Provide your users with a default alias to the `/bin/rm` command to ask for confirmation for deletion of files.
- Only become root to do single specific tasks. If you find yourself trying to figure out how to do something, go back to a normal user shell until you are **sure** what needs to be done by root.
- The command path for the root user is very important. The command path, or the PATH environment variable, defines the location the shell searches for programs. Try and limit the command path for the root user as much as possible, and never use '.', meaning 'the current directory', in your PATH statement. Additionally, never have writable directories in your search path, as this can allow attackers to modify or place new binaries in your search path, allowing them to run as root the next time you run that command.
- Never use the *rlogin/rsh/rexec* (called the ``r-utilities'') suite of tools as root. They are subject to many sorts of attacks, and are downright dangerous run as root. Never create a *.rhosts* file for root.
- The `/etc/securetty` file contains a list of terminals that root can login from. By default (on Red Hat Linux)

this is set to only the local virtual consoles (vtys). Be very careful of adding anything else to this file. You should be able to login remotely as your regular user account and then use *su* if you need to (hopefully over ssh or other encrypted channel), so there is no need to be able to login directly as root.

- Always be slow and deliberate running as root. Your actions could affect a lot of things. Think before you type!

If you absolutely positively need to allow someone (hopefully very trusted) to have superuser access to your machine, there are a few tools that can help. *sudo* allows users to use their password to access a limited set of commands as root. *sudo* keeps a log of all successful and unsuccessful *sudo* attempts, allowing you to track down who used what command to do what. For this reason sudo works well even in places where a number of people have root access, but use *sudo* so you can keep track of changes made.

Although *sudo* can be used to give specific users specific privileges for specific tasks, it does have several shortcomings. It should be used only for a limited set of tasks, like restarting a server, or adding new users. Any program that offers a shell escape will give the user root access. This includes most editors, for example. Also, a program as innocuous as `/bin/cat` can be used to overwrite files, which could allow root to be exploited. Consider *sudo* as a means for accountability, and don't expect it to replace the root user yet be secure.

4.9 Workstations and DialUp Security

User of computers to connect to the Internet via a dial-up line, or workstations that otherwise offer no services to external hosts can also improve their security with relatively easy modifications to the stock Linux installation.

If there is never have a need to connect to your machine from another one on the network, the quickest solution is to simply disable `/usr/sbin/inetd` from even being started. This is the master Internet daemon, which controls some normal server services, such as `telnet`, `ftp`, etc. If you retrieve your mail from a remote host, and your Internet Service Provider is hosting your web page, then most likely there is not a need to enable these services.

On stock Red Hat systems, the file `/etc/rc.d/rc3.d/S50inet` controls the starting and stopping of the `inetd` server. Simply rename the `S50inet` file to `s50inet` to disable it, or see your Red Hat administration manual for further information.

Alternatively, if you are a home dialup user, it is also possible to deny all incoming connections using TCP Wrappers. TCP Wrappers, `/usr/sbin/tcpd`, also logs failed attempts to access services, so this can give you an idea that you are under attack. If you add new services, you should be sure to configure it to use tcp_wrappers TCP based. For example, a normal dial-up user can prevent outsiders from connecting to your machine, yet still have the ability to retrieve mail, and make network connections to the Internet. To do this, you might add the following to your `/etc/hosts.allow`:

```
ALL: 127.
```

(including the ending period) And of course `/etc/hosts.deny` would contain:

```
ALL:
ALL
```

which will prevent external connections to your machine, yet still allow you from the inside to connect to servers on the Internet. TCP Wrappers can be combined with several other services, such as `sendmail` and `sshd` to give even further control over access. See the respective documentation for further information.

4.10 X11, SVGA and display security

X11

It's important for you to secure your graphical display to prevent attackers from doing things such as grabbing your passwords as you type them without you knowing it, reading documents or information you are reading on your screen, or even using a hole to gain superuser access. Running remote X applications over a network also can be fraught with peril, allowing sniffers to see all your interaction with the remote system.

X has a number of access control mechanisms. The simplest of them is host based. You can use xhost to specify what hosts are allowed access to your display. This is not very secure at all. If someone has access to your machine they can xhost + their machine and get in easily.

When using xdm (X Display Manager) to login, you get a much better access method: MIT-MAGIC-COOKIE-1. A 128bit cookie is generated and stored in your .Xauthority file. These cookies need to be transferred in confidence, and you really don't gain anything if your home directory is shared via NFS. If you need to allow a remote machine access to your display, you can use the xauth command and the information in your .Xauthority file to provide only that connection access. See the Remote-X-Apps mini-howto, available at http://sunsite.unc.edu/LDP/HOWTO/mini/Remote-X-Apps.html.

You can also use ssh (see ssh, below) to allow secure X connections. This has the advantage of also being transparent to the end user, and means that no un-encrypted data flows across the network.

Take a look at the `Xsecurity(1)` man page for more information on X security. The safe bet is to use `xdm(1)` to login to your console and then use ssh to go to remote sites you wish to run X programs off of.

SVGA

SVGAlib programs are typically `setuid-root` in order to access all your Linux machine's video hardware. This makes them very dangerous. If they crash, you typically need to reboot your machine to get a usable console back. Make sure any SVGA programs you are running are authentic, and can at least be somewhat trusted. Even better, don't run them at all.

GGI (Generic Graphics Interface project)

The Linux GGI project is trying to solve several of the problems with video interfaces on Linux. GGI will move a small piece of the video code into the Linux kernel, and then control access to the video system. This means GGI will be able to restore your console at any time to a known good state. They will also allow a secure attention key, so you can be sure that there is no Trojan horse login program running on your console. http://synergy.caltech.edu/~ggi/

4.11 identd

`identd` is a small program that typically runs out of your `inetd`. It keeps track of what user is running what tcp service, and then reports this to whoever requests it.

Many people misunderstand the usefulness of `identd`, and so they disable it or block all off site requests for it. `identd` is not there to help out remote sites. There is no way of knowing if the data you get from the remote identd is correct or not. There is no authentication in `identd` requests.

Why would you want to run it then? Because it helps *you* out, and is another data-point in tracking. If your `identd` has not been compromised, then you know it is telling remote sites the user-name or user-ID of people using TCP services. If the admin at a remote site comes back to you and tells you a user was trying to hack into their site, you can easily take action against that user at your site who is misusing a service. If you are not running `identd`, you will have to look at lots and lots of logs, figure out who was on at the time, and in general take a lot more time to track down the user.

The `identd` that ships with most distributions is more configurable than many people think. You can disable `identd` for specific users (they can make a `.noident` file), you can log all `identd` requests, which is recommended, and `identd` can return a uid instead of a user name or even NO-USER. Keep in mind it is really only useful is on a network where nobody hostile has root access. Then it can help in catching mail forgeries, for instance.
