

Configuring Secure Linux Hosts

By Landon Curt Noll

Introduction

Congratulations! You have just installed Linux, an open source operating system that does not suffer from the type of fundamental design flaws that plagues some other well-known operating systems. Your host is not a willing vector for the typical worm, virus, or spyware raging around the Internet. However, your security efforts do not end with the simple selection and installation of Linux. You can do more to harden your Linux host and improve its reliability, availability and security.

Why should you harden your Linux host? Industry standards call for doing more than just depending on the decent out-of-the-box security that comes with Linux. Best practices call for the principle known as defense in depth, which will help preserve host integrity from future attacks that are currently unknown.

While defense in depth is a term bandied about with regularity, we rarely stop to reflect on its literal definition, which is:

Defense in depth: having multiple complementary layers of security, avoiding single points of security failure.

Let's explore a few of the more common ways to harden a Linux host after the initial software installation, while leveraging the defense in depth concept. This article will not exhaustively cover the topic of Linux host hardening. Instead, it will address some of the more important aspects that you will need to consider.

There are a number of excellent Linux distributions to choose from for purposes of illustration. However, for the sake of this article, we had to pick one distribution, so we picked the one installed on a test system that happened to be Red Hat Enterprise Linux WS version 3. You will find that nearly all of the examples listed in this article can easily be adapted to other Linux distributions.

Filesystem Layout

Your host hardening process should start in the planning stages, even before you begin to install Linux. At a minimum, you should have the following filesystems:

- ▲ / (typically 8G or larger)
- ▲ /boot (typically 256M)
- ▲ /home (typically the rest of the disk)
- ▲ /tmp (typically 768M or larger)
- ▲ /var (typically 1G or larger)
- ▲ /var/tmp (typically 2G or larger)

There are several important principles at work here. First, the root filesystem, "/", should not be modified on a frequent basis. Scratch directories that are frequently written to such as "/tmp" and "/var/tmp" should

be in their own filesystems. The "/home" and "/var" directory trees also undergo frequent modification and so should be in its own filesystem.

The "/tmp" and "/var/tmp" directories are publicly writable. The "/home" filesystem contains home directories that are writable by users. System crackers sometimes attempt to compromise the integrity of a critical system file by forming a hard link in a publicly writable area where temporary files are commonly formed. You should place "/home", "/tmp", and "/var/tmp" into their own separate filesystems because hard links cannot cross filesystem boundaries.

The Linux kernel and other boot files stored in a filesystem are located under the "/boot" directory. By placing "/boot" in its own filesystem, it is possible to mount it read-only. Not only does it make it slightly harder for a system cracker to modify the next kernel or boot procedure, it reduces the chance that filesystem damage will negatively impact the next kernel or boot procedure.

Note that other filesystems are common (i.e., "/usr/local", "/usr/src", etc.) and have their advantages. You may list filesystems without impacting the principles listed above.

Filesystem Mount Options

You can improve the performance, reliability and security of your filesystems by adding the appropriate mount options to /etc/fstab after installation. You should strongly consider the following mount point options:

- ▲ noatime – Do not update the last access time
- ▲ nosuid – Ignore any set-user-ID or set-group-ID
- ▲ nodev – Ignore any special files
- ▲ ro – Read-only

By using the "noatime" mount option, you improve disk performance because Linux does not have to modify the "last access time" component of each inode. Accessing the file "/usr/local/bin/program" can result in the inodes for "/", "/usr", "/usr/local", "/usr/local/bin", and "/usr/local/bin/program" being accessed. The "noatime" option eliminates those disk writes.

Even if you move "/home", "/tmp", and "/var/tmp" out of the root filesystem "/" without "noatime", you will still be writing to the root filesystem on a regular basis. Additional disk writes come with the additional risk of filesystem damage due to a hardware or software problem or due to a system cracker-induced problem—even with ext3 journaling.

There is a very slight loss of forensic evidence when the kernel does not keep the last access time. We say slight because we have found very little value in examining the last access time of frequently accessed inodes such as "/", "/bin", "/usr/bin", "/etc", and the files that

are frequently accessed under them by automatic system activity. Any access time value is slim compared to the value of the loss of system performance and reliability.

Linux will happily operate with every disk-based filesystem mounted with the "noatime" option. You may have an application that absolutely requires a directory tree with last access times. If this is the case, you should place that tree into a special filesystem without the "noatime" option.

The "ro" read-only mount option eliminates the need to fsck the filesystem on a reboot. In the case of "/boot", some additional actions are needed when you install a new kernel. Prior to installing a new kernel, you need to make "/boot" temporarily read-write:

```
mount -o remount,rw /boot
```

Prior to rebooting the new kernel, you need to temporarily modify "/etc/fstab" to mount "/boot" read-write just once. The "/etc/fstab" line becomes:

```
LABEL=/boot /boot ext3 defaults,noatime,nodev,nosuid 1 2
```

After rebooting, you should restore the read-only status of "/boot":
You should also change the "/etc/fstab" line back to:

```
LABEL=/boot /boot ext3 default,noatime,nodev,nosuid,ro 0 0
```

By using "nodev" and "nosuid", mount options where appropriate, you can reduce the chance of a system cracker creating an inappropriate set-user-id or special file. Unlike "noatime", you cannot use these mount options everywhere. The following "/etc/fstab" entries for the six critical filesystems indicates where you can use these mount options:

```
LABEL=/ / ext3 default,noatime 1 1
LABEL=/boot /boot ext3 default,noatime,nodev,nosuid,ro 0 0
LABEL=/home /home ext3 default,noatime,nodev,nosuid 1 2
LABEL=/tmp /tmp ext3 default,noatime,nosuid 1 2
LABEL=/var /var ext3 default,noatime 1 2
LABEL=/var/tmp /var/tmp ext3 default,noatime,nosuid 1 2
```

Avoid Installing Software You Do Not Need

Software has bugs. More software has even more bugs. Sometimes system crackers use those bugs to compromise the integrity of the system. If you do not need a software package, then do not install it. You can always change your mind and install it later if you need it. It's simple and effective.

Turn Off Services You Do Not Need

If you do not need a service, turn it off. For example, if you do not need any network-based filesystems, you disable those services with the following command:

```
for i in autofs netfs portmap nfs nfslock; do
    chkconfig --level 0123456 "$i" off
done
```

You should only run the kudzu once after rebooting with new or modified hardware. After booting with new hardware and allowing kudzu to do its job, you can disable kudzu by:

```
chkconfig --level 0123456 kudzu off
```

If you will not be printing from the server, turn cups off:

```
chkconfig --level 0123456 cups off
chkconfig --level 0123456 cups-lpd off
```

If you will not be exporting samba mounts and samba printers to Windows-based systems:

```
chkconfig --level 0123456 smb off
chkconfig --level 0123456 rhdb off
chkconfig --level 0123456 windbind off
```

Turn off dangerous network services that authenticate with passwords sent over in the clear:

```
for i in rlogin rsh telnet; do
    chkconfig --level 0123456 "$i" off
done
```

You should disable services that external system crackers use to investigate current system activity:

```
for i in rhod rstatd rusersd; do
    chkconfig --level 0123456 "$i" off
done
```

You should also minimize the "xinetd" services that you enable. To disable all "xinetd" services:

```
cd /etc/xinetd.d
for i in *; do
    chkconfig --level 0123456 "$i" off
done
```

There are "xinetd" services that may be appropriate to use on your system. If there are, then "chkconfig" them back on. If, however, you do not need any "xinetd" service, then disable "xinetd" as well:

```
chkconfig --level 0123456 xinetd off
```

You can determine which services are on with the following command:

```
chkconfig --list | grep '.*on'
```

There are things you can do to learn more about a given service. To learn about the service "xyzy", you can do the following:

```
rpm -q -i -f /etc/rc.d/init.d/xyzy
head /etc/rc.d/init.d/xyzy
```

If in doubt, turn the service off. If the loss of the service impacts system functionality, you can always turn it back on by doing:

```
chkconfig --level 0123456 xyzyy on
```

You should note that using "chkconfig" changes if the system will start or stop the service on startup and shutdown. To disable an already-running service:

```
service xyzyy stop
```

Another method is to simply reboot the system after disabling a collection of services.

Keep Your Software Up-To-Date

From time to time, software is updated to fix a bug. Sometimes that bug fix is security related. It is important to keep your host software current.

Some distributions offer an update service. These update services usually offer notification of updates and the ability to automatically download them to your host. For example, Red Hat offers the Red Hat Network service. With a subscription, you can receive e-mail when one of your installed software packages has been updated. With a management option, your system can have your host download the update.

We recommend that hosts automatically download all updates, but do not install them. This allows you to schedule your software updates and test your software updates for an appropriate time.

Large installations may want to designate a host to be their software update host. They should have a software update host for each CPU architecture that they use. They should install onto such hosts any package used on another server of the same CPU architecture. In the case of Red Hat Network, subscribe with a management entitlement to enable automatic downloads. You can make this a proxy server or a satellite server (with a provisioning service) and use that system to distribute tested changes to your production hosts.

Ssh Configuration

The "ssh" tool is a very useful tool for remote access. It allows for safe authenticated system access and file transfers without the risk of a password going over the network in the clear.

You should configure "ssh" to add additional defenses in depth by changing the "/etc/ssh/sshd_config" file. The examples below give lines that you should be adding or change to this file.

You should disable protocol 1 and only use "ssh" protocol 2:

```
Protocol 2
```

You should change the port from 22 to another port and use that port throughout the hosts in your organization. We recommend picking a TCP port above 1024 that is not a registered protocol. The following URL gives the complete list of registered protocols:

<http://www.iana.org/assignments/port-numbers>

The lines of the form "*PortNumber*/tcp" indicate that "*PortNumber*" is a reserved port.

You should disallow root access by "ssh":

```
PermitRootLogin no
```

To become root on a remote machine, you should first "ssh" login to your own account and then use "sudo -s" to get a root shell.

You can give someone file copy access over the "ssh" version 2 protocol by means of the Open Source tool "rssh":

<http://sourceforge.net/projects/rssh/>

The "rssh" tool allows you to restrict on a per-user basis access to any of "rsync", "rdist", "cvs", "sftp", and/or "scp". You can control the file creation on a per-user basis. You can also force a given non-privileged user to be confined inside a chroot jail. If properly setup, the chroot jail will prevent the user from accessing any files outside of the chroot jail.

The "rssh" tool needs to be installed on any remote host into which you want to give selected users "rsync", "rdist", "cvs", "sftp", and/or "scp" access. The shells for those users must be "/usr/bin/rssh". Also the "/etc/rssh.conf" needs to be configured on that remote host. This will allow users to use any of the standard "rsync", "rdist", "cvs", "sftp", and/or "scp" tools to access the remote host via the "ssh" protocol 2.

You can further restrict access by means of TCP wrappers. See the next section for details.

TCP Wrappers

TCP wrappers allow you to restrict access to services that are TCP wrapper aware or that are controlled by the "tcpd" service. For general information on TCP wrappers, use the command:

```
man 5 hosts_access
man 8 tcpd
```

We recommend that you disable access by default to all services. The only non-comment/non-empty line in "/etc/hosts.deny" should be:

```
ALL: ALL
```

The "/etc/hosts.allow" is used to allow access that would otherwise be denied. For example, to restrict access to the "ssh" daemon to localhost and the Class C private network, add the following line to "/etc/hosts.allow":

```
sshd: 127.0.0.1, \
      192.168.0.0/255.255.0.0
```

To allow anyone to access the very secure ftp daemon, add the following line to "/etc/hosts.allow":

```
vsftpd: ALL
```

We recommend that you avoid using hostnames in the TCP wrapper "client-lists". You should use only IP addresses or IP/netmasks.

Network Parameter Modifications

You can modify the network parameters to harden your Linux host by adding lines to the "/etc/sysctl.conf" file. These parameters are documented in the following kernel file:

</usr/src/linux-2.4/Documentation/networking/ip-sysctl.txt>

To handle SYN floods better:

```
net.ipv4.tcp_max_syn_backlog = 4096
```

Unless your host is a routing gateway between two or more networks, you should disable redirects:

```
net.ipv4.conf.lo.secure_redirects = 0
net.ipv4.conf.lo.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
```

For each interface ("eth0", "eth1", ...) disable redirects as well:

```
net.ipv4.conf.eth0.secure_redirects = 0
net.ipv4.conf.eth0.accept_redirects = 0
```


Conclusion

Linux provides an excellent foundation to build a secure and reliable host. Linux, unlike some other popular operating systems, does not suffer from the kind of fundamental design flaws that allow viruses, worms, spyware, and spambots to rage throughout the Internet. Nevertheless, you can build on top of Linux security by building a defense in depth.

We have given you some of the more common ways to further harden your Linux host. Our list is by no means exhaustive. It does represent some of the more important changes you can do to further harden your Linux host.

For those who want to go beyond these steps, we recommend referencing the Center for Internet Security's CIS Level-1 Benchmark and Scoring Tool for Linux:

http://www.cisecurity.org/bench_linux.html

Last, we highly recommend that you consider hiring the services of an outside security-consulting firm to evaluate and test the security of your critical production hosts. Competent consultants can give you a fresh perspective, catch problems that you may have otherwise overlooked, and mitigate against known security threats. 

Landon Curt Noll is a consultant with SystemExperts Corporation (www.system-experts.com), a recognized leader in computer and network security.