Linux Hardening Recommendations For Cisco Products

Version 1.0

Contents

List of Tables

1 Introduction

1.1 Motivation

Cisco has released a significant number of products that are built on the Linux operating system (OS). In the past, security-related Linux configuration issues have been observed in Cisco products. This document aims to identify security criteria specifically applicable to Linux that should be met in order to ensure a certain baseline level of security for Cisco's Linux-based products. As Cisco releases an increasing number of products on the Linux OS, it is important that their underlying OS is configured securely.

1.2 Applicability

The purpose of this document is to provide a concise and manageable set of security configuration recommendations for Linux when embedded within Cisco products. Cisco Secure Development Lifecycle (CSDL) references many hardening guides for various operating systems. Several of these are for Linux, however their focus is on the secure configuration of general purpose computers running Linux, not on how Linux in an embedded system should be configured by the system developers. This document is intended to assist Cisco engineers who are developing Linux-based systems.

This document is the product of a survey performed across many well-known general purpose Linux Hardening guides and security tools. For the survey, each of the recommendations in the general-purpose guidelines were prioritized and ranked according to their level of applicability within the environment of Cisco's products built on Linux. Only recommendations with general applicability across all Linux distributions were considered for inclusion.

This document is not intended to provide an exhaustive list of hardening recommendations to cover all installation variants (desktop, server, etc.), nor is it intended to cover the security configuration corner cases for all the different Linux distributions in use within Cisco.

1.3 Audience

This document is written for technical personnel and developers with some knowledge about Linux configuration, for the purpose of improving the security posture of Linux-based Cisco products. This document can also be useful for personnel that produce technical requirements.

1.4 Organization

Each major section title describes the high-level objective, followed by subsections that outline specific methods to achieve the objective.

Significantly lengthy implementation instructions (e.g., configuration files) and conditional recommendations are in the Appendix.

1.5 Scope

The recommendations in this document pertain to Cisco products that utilize Linux. Although many of these recommendations can be applied to similar Unix-based platforms, non-Linux operating systems are beyond the scope of this document.

2 Linux Hardening Recommendations

2.1 Remove or Disable Unnecessary Services, Ports, and Devices. Otherwise, Configure to be Secure.

2.1.1 Services and Utilities

Services and utilities that do not have a defined purpose on a system should be removed [2, 3, 7, 9, 10]. If removal is not possible, but the service or utility can be disabled, then it should be disabled. If a service or utility is necessary, available secure configuration best practices should be implemented. Security configurations for some well-known services and utilities are in Appendix A.

The following legacy services are inherently insecure and should be avoided [7, 19, 1, 6]:

rlogind, rshd, rcmd, rexecd, rbootd, rquotad, rstatd, rusersd, rwalld, rhosts, rexd

These services offer insufficient authentication, no encryption, and are not considered secure. They should be removed along with their configuration files.

Telnet, **FTP,** and **NFS** have security weaknesses that are well known, however customers may have requirements to use these services. If remote shell access and file transfer are required, then provide more secure options, such as **SSH** and **SFTP**/**SCP**.

2.1.2 Open Ports

Open network ports provide an attack surface for potential remote exploits. If an open port does not belong to a service/protocol with a defined purpose on the system, then it should be closed. Services should only be run on their "well-known" ports (as identified in the IANA Service Name and Transport Protocol Port Number Registry). Unregistered services/protocols should never be run on ports that are already registered [5]. Remove support for unused protocols by disabling them in the compile-time kernel configuration. For example, if SCTP is not needed, then compile the kernel with `CONFIG_IP_SCTP=n`. To find open ports, use the **netstat** and **lsof** utilities as follows:

```
netstat -lntup
lsof -i
```

To close the port, kill the service process that is listening on the port, and then remove the package associated with the service.

2.1.3 Devices/Interfaces

Devices and interfaces that are not needed should be disabled/removed. This includes:

- Network interfaces (including wireless interfaces)
- Storage devices (including USB)
- Console devices
- Bluetooth interfaces

If the device or interface is not needed, remove the driver. Do not automount devices (such as USB).

2.2 Separate System Critical Files and Directories from Dynamic Content

Dynamic content, like logs or temporary files, should be separated from system critical items, like the OS and installed applications [2, 4, 6]. This mitigates scenarios that may result in systems crashes due to dynamic content (e.g. tmp directories) filling up the storage space and making the system unstable or unusable. Because of this, dynamic content and system critical resources should be mounted on separate partitions:

- Mount **/tmp** and **/var/run** on their own filesystems.
- Bind-mount **/var/tmp** to **/tmp**

Although quotas are not supported by some filesystems, they can be utilized to limit the amount of storage space that dynamic content can utilize on products with applications that are not run with root privileges [7]. To enable quotas:

- Add the "usrquota" or "grpquota" qualifier to the partition in **/etc/fstab**.
- Use the **`quotaon`** command to enable disk quotas on the system
- Use the **`edquota`** commands below to launch a text editor, in which quota values can be set for accounts and groups respectively:

```
edquota -u username
edquota -g group
```

Quota information storage varies by filesystem type. However, it is usually stored in **aquota.user** and **aquota.group** or **quota.user** and **quota.group** files in the root of the filesystem.

2.3 Control Access to, and Enforce Permissions on Important Resources

2.3.1 World-Writable Files and Directories

Configuring files and directories to be writable (world-writable) by all users on the system is not recommended. World-writable items can be written to and modified by every account on the system. Because of this, files and directories with any significance should not be world-writable.

If the system must have a world-writable directory (e.g. **/tmp**), it is advisable to set the sticky bit on the directory to ensure that files in that directory may be deleted or modified only by root or their owner [1, 6, 7, 9, 11, 18].
To find all world-writable files on a filesystem, run:

```
% find / -type f -perm -002 -print
```

To set the sticky bit on the **/tmp** directory, run:

```
% chmod +t /tmp
```

File permission and ownership recommendations for common files and directories in Linux are in Appendix B.

2.3.2 Root File Ownership

Files that should only be readable, writable, or executable by root, should be owned and group-owned by root. If these files are owned by an account other then root, then the account controls permissions on the file.

File ownership recommendations for common files and directories in Linux are in Appendix B.

2.3.3 UIDs and GIDs

User identifier (UID) values reference kernel-level user accounts on the system and group identifier (GID) values reference groups. Identifier assignment can result in elevated privileges if an account is assigned a UID/GID with elevated privileges. UID 0 is a privileged UID and should be reserved only for the root account [1]. To find all SUID programs, run:

```
# find / -type f -perm +6000 -ls
```

If periodic privileged access is required through su or sudo, logging should be enabled. Sudo allows programs to be run with root privileges and can lead to easy escalations to root.

When the set user ID (SUID) or set group ID (GUID) attributes are set on executables in the filesystem, those executables will be run as the user or group owning the file (typically the root user). This is commonly used to enable non-root users to perform small tasks that require root privileges, like changing their system password with the SUID-root "**passwd**" program.

```
% ls –l /usr/bin/passwd
  -rwsr-xr-x 1 root root 42824 2011-06-24 04:28 /usr/bin/passwd

The 's' bit above signifies that the SUID attribute is set for passwd
```

Since programs should not run with unnecessarily elevated privileges, remove the SUID/SGID bits from any programs that do not need the elevated privileges [7, 8].

Consider utilizing Access Control Lists (ACLs) and extended filesystem attributes (xattr) as alternatives to SUID/SGID root programs [17]. ACLs allow more flexibility and fine-grained control over the permissions of files and directories than standard permissions. For example, this would allow a program to update a root-owned file while executing without root permissions. Extended file attributes allow additional configurable permissions that include preventing file deletion or modification, and permitting or denying symbolic links on a file.

2.3.4 Umask

Umask is a function that sets a file mode creation mask for processes. When a shell or other program creates a file or directory, it can specify permissions to be granted to the created entity. However, permissions that the file mode creation mask does not allow, are not set. Because of this, the following umask recommendations should be set for each daemon/user account on the system:

- Umask values should allow read, write, and execute permissions for **owner**
- Umask values should not allow write permissions for **group** and **other**
- If read and/or execute privileges are not needed for **group** and/or **other**, then umask values should not allow this

The effective umask can be set in the shell interpreter that is used to launch applications. The following example sets the umask to 022, which sets default directory permissions to `chmod 755` and default file permissions to `chmod 664`):

```
% umask 022
```

2.3.5 Permissions

The Principle of Least Privilege requires that in a particular abstraction layer of a computing environment, modules should be able to only access resources that are necessary for their legitimate purpose. This principle should be applied within the context of filesystem ownership and permissions. Files and directories should only grant the minimum permissions for a program or a user to accomplish their intended duty.

Directories that contain files or sub-directories that are not writable should also not be writable. If the containing directory is writable, another account could copy the file, modify the file, then replace the original file with the copied file.

2.3.6 Shell Access

In a Linux environment, privilege escalation can be achieved by accessing a root shell through a service that has root privileges. Any user/process that gains access to a root shell will possess root privileges on the system. Where it is possible, programs that run with elevated privileges should be configured to prevent unnecessary file reads/writes, shell escapes, and execution of other programs.

Login shell access should be disabled for non-root system/daemon accounts on the system [4].

If remote shell access is necessary, using ssh instead of telnet is strongly encouraged. Unlike ssh, the telnet protocol does not encrypt the information that it transfers, this includes authentication credentials. This is insecure and should be avoided. If telnet is not needed on the system, then it should be removed.

Configuration recommendations to restrict shell access are in Appendix C.

2.3.7 File Transfer

For non-anonymous file transfer, using **scp** or **sftp** instead of **ftp** is strongly encouraged because **ftp** allows insecure authentication and does not encrypt transmitted data.

2.3.8 Account Management

Linux user/daemon accounts that are not necessary should be deleted. If single-user mode is not necessary, then it should also be disabled [7, 10].

2.3.9 Partition Mount Options

There are several security restrictions that can be set on a filesystem when it is mounted. Some common security options include, but are not limited to:

- **nosuid** - Do not allow set-user-identifier or set-group-identifier bits to take effect
- **nodev** - Do not interpret character or block special devices on the filesystem
- **noexec** - Do not allow execution of any binaries on the mounted filesystem
- **ro** - Mount filesystem as read-only

Filesystems options can be set by adding/editing lines formatted like the following line in the **/etc/fstab** file:

	<file system>	<mount point>	<type>	<options>	<dump>	<pass>
Example:	/dev/sda2	/boot	ext3	nosuid,nodev	0	0

Changes to fstab are applied at boot-time. To mount all of the filesystems in fstab at run-time, run:

```
sudo mount -a
```

The following are recommendations for mounting common filesystems:

- For /boot partition, use **nosuid** and **nodev** and consider using **noexec** [7, 17]
- For **/var** and **/tmp** in your **/etc/fstab** or **vfstab** file, add **nosuid**, **nodev** and **noexec** [7, 10]

- For non-Root local partitions (If the filesystem type is ext2 or ext3 and The mount point is not '/'), add **nodev** option [1]
- For to removable storage partitions, add **nodev**, **nosuid**, and **noexec** options [1]
- For temporary storage partitions add **nodev**, **nosuid**, and **noexec** options [1]
- For **/dev/shm**, add **nodev**, **nosuid**, and **noexec** options [1]

2.3.10 Filesystems

Filesystems should support POSIX permissions at a minimum.

2.4 Ensure the Integrity of Critical Files and Environment Parameters

Remove configuration files associated with services and applications that are not on the system.

Root's login files should not source any file owned by another user or that are group/world writable [7].

2.4.1 Environment Variables

Environmental variables, including the PATH variable, should only contain absolute paths. They should not contain relative directories (eg. ../.. or . or ..), the root directory(/), or empty strings [7]. Environmental variables should only reference existing directories and should not reference world/group writable directories [6, 7, 9, 14, 18]. This will prevent other accounts from writing to directories in the $PATH.

2.5 Enabled Logging With Log Rotation

In order to keep an accurate audit of activity on the system, logging of all major system events should be enabled. In Linux, logging is typically managed by the **syslogd** daemon. Systems should run with logging enabled at all times.

Logging services (eg. syslogd) should be enabled in the system's startup scripts. These can vary based on the Linux distribution, but they are typically located in the **/etc/rc.d/** or **/etc/init.d/** directory.

Log data can grow to fill disks quickly. Because of this, log rotation should be enabled [1]. Separate programs from the actual logging services (eg. savelog, logrotate) typically handle file rotation. Log rotation program(s) must run in conjunction with logging services.
Similar logging programs (eg. rsyslog) attempt to listen to the same sockets (as syslog) and compete for resources. Using multiple loggers to monitor the same resources is not recommended.

2.6 Restrict direct access to memory through device files

The **/dev/mem** and **/dev/kmem** files in Linux systems are directly mapped to physical and kernel memory respectively. This can be disastrous if an attacker gains root access, as they would have direct access to physical and kernel memory. Write access to memory through the **/dev/mem** device file should be disabled [5, 18]. To do so, the following kernel option should be set in the compile-time kernel configuration:

```
CONFIG_STRICT_DEVMEM=y
```

To disable the **/dev/kmem** file, the following kernel option should also be set in the compile-time kernel configuration:

```
CONFIG_DEVKMEM=n
```

2.7 Configure Secure Development and Runtime Defense Features

2.7.1 ASLR

Address Space Layout Randomization (ASLR) randomizes where specific memory components (eg. stack, heap, libraries) are mapped in the address space of a process. This makes the exploitation of traditional memory corruption vulnerabilities probabilistically difficult. ASLR should be enabled for systems that support it [14].

2.7.2 X-Space Protection

Executable Space protection (X-Space) is a security feature found in microprocessors that augments the memory area attributes to indicate whether software can be executed from that memory location. It is also known at nX, Xbit, xD, and MMU from different microprocessor vendors. When applied to processes, this can prevent memory areas on the stack and from being executable. X-Space protection is dependent on:

- Whether the CPU supports No eXecute technology.
- Whether the kernel utilizes X-Space protection.
- Whether the build tool-chain produces executables that have proper ELF headers.

If the system supports X-Space protection, then it should be enabled.

2.7.3 Object Size Checking

C and C++ should be compiled with Built-in Object Size Checking (BOSC) to provide limited buffer overflow protection mechanisms that are designed to detect buffer overflows at compile time. If the compiler is unsure about the safety of a particular call, it will remap the call to a wrapper that offers some run time buffer overflow protection.

If the GCC compiler supports BOSC, then it can be enabled at compile time by passing GCC the "-D_FORTIFY_SOURCE=2" option. Note that GLIBC must also be (re)compiled with this flag in order to fully enable runtime protections. Example:

```
% gcc -D_FORTIFY_SOURCE=2 abcd.c
```

2.7.4 Stack/Heap Smashing Protection

Applications should be compiled with available stack protection features enabled. GCC has **fstack-protector** and **fstack-protector-all** flags that enable ProPolice's stack protection mechanisms [5, 15].

The GNU C Library heap protector provides corrupted-list/unlink/double-free/overflow protections to the glibc heap memory manager [5]. This prevents heap memory overflows that attempt to corrupt the control structures of the malloc heap memory areas.

2.7.5 Compilers

Compilers are utilized in development environments. Since software components in Cisco's Linux-based products are typically compiled and built prior to the product being shipped, the products should not be shipped with compilers. If a product is compromised, compilers provide an attacker with the ability to compile code.

2.7.6 Debuggers

If the customer model does not require debugging capabilities, then the product should not be shipped with debuggers.

Additional recommended kernel hardening recommendations are in Appendix D.

2.8 Utilize Additional Security Features and Tools When Necessary

It may be beneficial to utilize existing tools to achieve additional security. The following tools may not be necessary in all scenarios, however they offer useful security features. These tools include:

- **Linux Pluggable Authentication Modules (PAM)** - PAM provides dynamic authorization for users, applications, and services on a Linux system. Account verification criteria can include expiration date and duration, time of day, and access to a requested service.

- **Linux Security Modules (LSM)** - LSM is a framework that allows the Linux kernel to support a variety of access control related computer security models. Some modules that utilize the LSM framework are described below:

    - **Security-Enhanced Linux (SELinux)** - SELinux is a Linux feature that provides a mechanism for supporting access control security policies, including mandatory access controls, through the use of Linux Security Modules (LSM) in the Linux kernel.

    - **Application Armor (AppArmor)** - AppArmor is a security module for the Linux kernel, released under the GNU General Public License. AppArmor allows the system administrator to associate with each program a security profile that restricts the capabilities of that program.

- **Grsecurity** – Grsecurity is a set of security related patches for the Linux kernel. This allows the kernel to support role-based access control, program memory protection, and chroot restrictions. Grsecurity includes PaX, a patch that implements ASLR and X-Space protections above and beyond those included in the mainline kernel. Recommended grsecurity configurations are in Appendix E.

- **Tripwire** - Tripwire functions as a file-modification detection system. Rather than attempting to detect intrusions at the network interface level (as in network intrusion detection systems), Open Source Tripwire detects changes to files, which may reveal if the system has been compromised in some fashion.

- **Samhain** – Samhain another host-based intrusion detection system. However, unlike Tripwire, it utilizes cryptographic checksums of files to monitor file integrity. Samhain can also:
    - Find rogue SUID executables anywhere on disk.
    - Monitor which ports are open on the local host and compare against a list of allowed or required port/services.
    - Check for hidden processes (i.e. not listed in the output from 'ps')

- **ExecShield -** ExecShield is a security Linux kernel patch to primarily found in Red Hat's Linux distribution, to implement some ASLR and X-Space protections. If the system supports ExecShield, then it should be enabled. Add the following line in the run-time kernel configuration to enable ExecShield:

```
kernel.exec-shield=1
```

Third party software vulnerabilities can provide an attack surface for potential exploits. Because if this, it is important to make sure that security patches and upgrades for third party software have been applied.

3 References

[1] NSA, Guide to the Secure Configuration of Red Hat Enterprise Linux 5
http://www.nsa.gov/ia/_files/os/redhat/NSA_RHEL_5_GUIDE_v4.2.pdf

[2] Werner Puschitz, Securing and Hardening Red Hat Linux Production Systems
http://www.puschitz.com/SecuringLinux.shtml

[3] SANS Institute, Linux Security Checklist
http://www.sans.org/score/checklists/linuxchecklist.pdf

[4] Red Hat, Hardening Red Hat Enterprise Linux 5
http://www.redhat.com/promo/summit/2008/downloads/pdf/hardening-rhel5.pdf

[5] Ubuntu, Ubuntu Security Feature Matrix & Tools
https://wiki.ubuntu.com/Security/Features

[6] CIS, Security Configuration Benchmark For Red Hat Enterprise Linux 5
http://benchmarks.cisecurity.org/tools2/linux/CIS_RHEL_5.0-5.1_Benchmark_v1.1.2.pdf

[7] AusCERT , UNIX and Linux Security Checklist v3.0
http://www.auscert.org.au/5816

[8] CERT, UNIX Configuration Guidelines
http://www.cert.org/tech_tips/unix_configuration_guidelines.html

[9] Kevin Fenzi, Linux Security HOWTO
http://www.linuxdoc.org/HOWTO/Security-HOWTO/file-security.html#umask

[10] Bob Cromwell, How To Harden a Linux or OpenBSD Installation
http://www.cromwell-intl.com/security/linux-hardening.html

[11] David A. Ranch, A Guide to Configuring Your Linux Server
http://www.trinityos.com/LINUX/TrinityOS/mHTML/TrinityOS-m.html

[12] Linux Security Auditing Tool (LSAT)
http://freshmeat.net/projects/lsat/

[13] Tiger, Security Audit and Intrusion Detection System
http://www.nongnu.org/tiger/

[14] checksec.sh
http://www.trapkit.de/tools/checksec.html

[15] CIS Security/Benchmark Assessment Tools
http://benchmarks.cisecurity.org/en-us/?route=downloads.audittools#other_tools

[16] Lynis
http://www.rootkit.nl/projects/lynis.html

[17] Rootkit hunter
http://www.rootkit.nl/projects/rootkit_hunter.html

[18] UNIX/Linux Local Audit Tool
http://www.boran.com/security/sp/solaris/audit_tool.html

Appendix

A. Secure Configuration for Well Known Services and Utilities

This section outlines hardening recommendations for well-known Linux services and utilities.

OpenSSH

In the **/etc/ssh/sshd_config** file [1, 2, 3, 4, 6, 7, 10, 19]:

```
# SSH-1 is obsolete and should be avoided at all cost.
# Only Use SSH Protocol 2.
Protocol 2

# Configure Idle Log Out Timeout Interval
ClientAliveInterval 300
ClientAliveCountMax 1

# Don't read ~/.rhosts and ~/.shosts files
IgnoreRhosts yes

# Disable Host-Based Authentication
HostbasedAuthentication no

# Disable root Login via SSH
PermitRootLogin no

#prevent users from being able to present
#environment options to the SSH daemon
PermitUserEnvironment no

#Disable Empty Passwords
PermitEmptyPasswords no

# Turn on privilege separation
UsePrivilegeSeparation yes

# Disable TCP Forwarding
AllowTcpForwarding no

# Disable X11 Forwarding
X11Forwarding no

# Disable port forwarding (tunneling)
PermitTunnel no
```

DHCP Server

This section consists of best practices for DHCP server.
In the **/etc/dhcpd.conf** file, Add or correct the following parameters [1]:

```
# To prevent the DHCP server from receiving DNS information from clients
ddns-update-style none;

# Preventing the DHCP server from responding to DHCPDECLINE messages.
deny declines;

# Unless your network needs to support older BOOTP clients,
# disable support for the bootp protocol
deny bootp;
```

For each address range within **/etc/dhcpd.conf**, ensure that the following options are not defined unless there is an operational need to provide this information:

```
option nis-servers
option ntp-servers
option time-offset
```

DHCP Client

This section outlines best practices for DHCP client configuration.
The following DHCP client parameters are set in the **/etc/dhclient.conf** file. If this file does not exist, then create it.
For settings that should not be configured remotely by the DHCP server, select an appropriate static value, and add the following line [1]:

```
supersede setting value;
```

If the DHCP server should configure the setting remotely, add the following lines [1]:

```
request setting;
require setting;
```

The following is an example of a DHCP configuration for a server that should only provide an IP address and the subnet mask [1]:

```
supersede domain-name "example.com";
supersede domain-name-servers 192.168.1.2;
supersede nis-domain "";
supersede nis-servers "";
supersede ntp-servers "ntp.example.com";
supersede routers 192.168.1.1;
supersede time-offset -18000;
request subnet-mask; require subnet-mask;
```

SNMP Server

Listed below are best practices for SNMP server configuration [1]:

- Use only SNMP version 3 security models and enable the use of authentication and encryption

- Write access to the MIB (Management Information Base) should be allowed only if necessary
- All access to the MIB should be restricted following a principle of least privilege
- Ensure SNMP agents send traps only to, and accept SNMP queries only from, authorized management stations
- Ensure that permissions on the **snmpd.conf** configuration file (by default, in **/etc/snmp**) are 640 or more restrictive
- Ensure that any MIB files' permissions are also 640 or more restrictive

B. Permission and File Ownership Recommendations

Resource	Owned	Group-owned	Mode
system audit logs top-level directories like /var, /usr, /sbin, /sys	root	root, bin, sys, or system	0640
root account's home directory (other than /)	root	root	0700
time synchronization configuration file (such as /etc/ntp.conf)	root	root, bin, sys, or system	0640
network services daemon files			0755
system command files	system account	system group	0755
system log files			0640
library files			0644
NIS/NIS+/yp files	root, sys, or bin	root, sys, bin, other, or system	0755
/etc/resolv.conf file /etc/hosts file /etc/nsswitch.conf file /etc/passwd file /etc/group file	root	root, bin, sys, or system	0644
/etc/shadow file	root	root, bin, sys, or system	0400
user home directories	respective user	owner's primary group	0750
files and directories contained in user home directories		group of which the home directory's owner is a member	
system start-up files	root	root, sys, bin, other, or system	
global initialization files	root	root, sys, bin, other, system or the system default	0644
skeleton files (typically those in /etc/skel)	root or bin	root, bin, sys, system, or other	0644
local initialization files	user or root	user's primary group or root	0740
shell files	root or bin	root, bin, sys, or system	
Public directories	root or an	root or an application group	

	application account		
cron.allow	root, bin, or sys	root, bin, sys, or cron	0600
crontab files			0600
files in cron script directories			0700
cron and crontab	root or bin	root, sys, bin or cron	0755
cronlog			0600
cron.deny	root, bin, or sys	root, bin, sys, or cron	0700
at.deny	root, bin, or sys	root, bin, sys, or cron	0600
at.allow	root, bin, or sys	root, bin, sys, or cron	0600
"at" directory	root, bin, or sys	root, bin, sys, or cron	0755
alias file	root	root, sys, bin, or system	0644
snmpd.conf	root	root, bin, sys, or system	0600
Management Information Base (MIB) files			0640
/etc/syslog.conf file	root	root, bin, sys, or system	0640
SSH public host key files			0644
SSH private host key files			0600

Table 1: Permission and Ownership Recommendations

File permissions for **user**, **group**, and **others** are checked and set using the `chmod` command. The following example grants read, write, and execute permissions to the **owner** of the file. It also grants only read and execute permissions to **group** and **other**:

```
% chmod 755 file
```

Because of the **user**, **group**, and **other** tuple in POSIX permissions, the file owner directly affects how permissions dictate the ability to perform certain actions on a file or directory. To change the file owner, use the chown command. The following example changes the file owner to `another-user`:

```
% chown another-user file
```

To change the group owner, use the `chgrp` command. The following example sets the group owner to `new-group`:

```
% chgrp new-group file
```

C. Disable Root Shell

This section outlines configurations to restrict shell access in well-known Linux services.

Linux Services

To prevent `gdm`, `kdm`, `login`, `xdm`, `su`, `ssh`, `scp`, and `sftp` from gaining root shell access, change root's shell to **/sbin/nologin** in the **/etc/passwd** file (`nologin` may be in **/usr/sbin/nologin**, depending on the system.) [4].

Console Devices

To disable root access via any console device (tty), remove lines in the **/etc/securetty** file that grant root access to specific devices [6, 10, 23].

OpenSSH

To disable root shell access for ssh, add the following line in **/etc/ssh/sshd_config** file [2, 3, 4, 10, 18]:

```
PermitRootLogin no
```

FTP Server

To disable shell access to the ftp user account, set the shell to **/bin/false** and **/bin/nologin** or **/usr/sbin/nologin**.

D. Secure Run-time Kernel Configuration

Run-time kernel parameters can be enabled/disabled by modifying the **/etc/sysctl.conf** file. Kernel parameters can also be set at run-time using the **sysctl** command. However, parameters that are set using the **sysctl** command are nullified during a system shutdown or restart. Unlike the **sysctl** command, kernel parameters in the **/etc/sysctl.conf** file are set at boot-time, thus persist through system shutdowns and restarts.

The following are recommendations for Linux kernel settings. These parameters are set in **/etc/sysctl.conf** [1, 2, 4, 6]:

```
# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename
# Useful for debugging multi-threaded applications
kernel.core_uses_pid = 1


########## IPv4 networking ##############

# Unless system is going to be used as a firewall
# or gateway to pass IP traffic between different networks
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
```

```
# Accept packets with SRR option? No
net.ipv4.conf.all.accept_source_route = 0

# Disable Accept Redirects
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0

# Log packets with impossible addresses to kernel log
net.ipv4.conf.all.log_martians = 1

net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0

# Ignore all ICMP ECHO and TIMESTAMP requests sent to it via
broadcast/multicast
net.ipv4.icmp_echo_ignore_broadcasts = 1

# Prevent against the common 'syn flood attack'
net.ipv4.tcp_syncookies = 1

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

########## IPv6 networking ##############

net.ipv6.conf.all.accept_source_route=0
net.ipv6.conf.default.accept_redirects = 0
net.ipv6.conf.default.secure_redirects = 0

# net.ipv6.conf.default.max_addresses  should be set to
# the number of ipv6 addresses that should be assigned to
# the interface.
net.ipv6.conf.default.max_addresses = 1
```

The following additional kernel parameters should be set in the kernel configuration file (file location can vary by distribution) [5]:

```
# Enable -fstack-protector buffer overflow detection
# As of Linux 3.0, this only enables overflow protection for x86 and ARM.
# This excludes x86 and MIPS
CONFIG_CC_STACKPROTECTOR=y

# Write protect kernel read-only data structures
# As of Linux 3.0, this only impacts x86 and PARISC
CONFIG_DEBUG_RODATA=y
```

E. Grsecurity Configuration

Recommended grsecurity settings are below [18]:

```
# Disable writing to kmem/mem/port
CONFIG_GRKERNSEC_KMEM=y

#Disable privileged I/O
# CONFIG_GRKERNSEC_IO will break XFree86.
CONFIG_GRKERNSEC_IO=y

#Harden module auto-loading
CONFIG_GRKERNSEC_MODHARDEN=y

# Hide kernel symbols
CONFIG_GRKERNSEC_HIDESYM=y
```