



Interested in learning
more about security?

SANS Institute

Security Consensus Operational Readiness Evaluation

This checklist is from the SCORE Checklist Project. Reposting is not permitted without express, written permission.

SCORE Security Checklist

Linux Hardening

General Checklist

Created June 2012

Updated July 2012

Authors:

Paul Loftness

Simeon Blatchley

Overview

This document is a general checklist for hardening a linux system. It is comprised of two other types of documents which will be referred to at various times throughout this general checklist. They are, **Advanced Checklists** and **Configuration Checklists** (see description below). Both are for the advanced hardening of your system, and require more knowledge, skill and also have more of the potential to break something. The important thing to remember is that there is no 100% right checklist. There are bound to be variables that must be changed, and all this document is intending on doing, is to allow the Linux user to follow the steps and successfully secure any type of system without needing much knowledge. However, they will still have the ability to further their security with the more advanced checklists. Of course with the more advanced checklists, there is more of a chance of “breaking” something,

and thus all “steps” must be researched for **your** specific distro/system. A single user's security settings will be vastly different from a multi-user system.

Note: All commands listed will need to be run as root. You can switch to root by running either ‘sudo -l’ or ‘su.’

Note: Where we use “vi” as the command line editor, you can replace it for “gedit” or a gui editor.

Note: Where use “apt-get” you can insert your distro version of package management. Or if necessary you can download the binaries and compile them (a somewhat easy process of ./configure, make, make install, etc).

Note: Shaded areas are terminal commands, you can cut and paste these, although one should be careful and know what the command actually does.

Advanced Checklists: These are checklists that go into more detail of various security aspects, and are not to be necessarily strictly followed. As the testing environment may differ from your system. However, when deployed properly they can greatly improve the system security.

Configuration Checklists: These are pretty self explanatory. They are just what we are suggesting as the configuration of certain security packages, scripts, etc. (like AppArmor and Bastille). Essentially, when there are variables that need to be inputted and what you put may greatly effect the security, these checklists will help you better decide what options to choose/use. Remember: Although we may say “choose options 'X'”, that is strictly a guideline, and it is your job to know what options will work for your system. We will try to note, where are options will not work on certain systems

Maintenance:

1. Update the Operating System:

Debian/Ubuntu/etc

```
apt-get update  
apt-get upgrade
```

Redhat, YellowDog, CentOS, Scientific Linux, Fedora, etc.

```
yum list updates  
yum update
```

Suse

```
zypper ref    (Refresh the repos)
zypper dup    (Normal update and install)
```

Harden the System

1. Install Bastille.

There are a few options around to harden a linux system, but we have tested Bastille in real life scenarios and found it to be the most resilient. It is rather customizable for various types of configurations.

```
apt-get install bastille
```

Choose yes when it asks if you want to continue. Once it is done installing, run:

```
bastille -c
```

This will start the command line interface, to allow you to configure Bastille. From there, you'll accept their terms of agreement, and be on your way. It is safe to say that you can just accept the default values, however you should also read about them. Please see our Bastille Configuration file for a more detailed look at Bastille. It's safe to ignore most errors it throws at the end and beginning of the configuration.

2. Install Apparmor.

Some packages will install their own enforced profiles. Active profiles for LAM Server:

usr.sbin.mysql

usr.sbin.apache2

All activity will be logged by auditd and saved to
/var/log/audit/audit.log

```
apt-get install apparmor-profiles
apparmor_status (to see current profiles and associated modes)
man apparmor (for more details of what to do with that information)
```

3. Configure and Use SELinux

As this is more complicated and advanced alternative to Apparmor, there is a detailed checklist specifically for completing the below actions:

a) Installation varies greatly. Please look-up the process for your distribution.

b) activate

Temporarily: `setenforce 0|1`

0 activates permissive (monitoring) mode.

1 activates permission enforcement.

c) Service Profiles

Using SELinux on a service:

List available SELinux service profiles:

```
man -k _selinux
```

To explore a specific profile: `man httpd_selinux`.

This will provide the commands to engage SELinux for the service for your distribution.

d) Service Settings

SELinux provides a number of boolean (on or off) settings for each service.

```
semanage boolean -l
```

Lists the current status, permanent status, and an explanation of each boolean

To turn a boolean on:

```
setsebool example_boolean on
```

-P makes the change permanent

4. Configure and use PAM authentication daemon

The instructions below are assuming that you do not have SELinux installed.

These configurations may change with the installation of SELinux. They will be covered in the SELinux detailed checklist. Also for further PAM info, refer to the PAM Configurations checklist.

```
vi /etc/pam.d/common-password
```

change:

```
password requisite pam_unix.so nullok obscure sha512
```

to:

```
password requisite pam_unix.so nullok obscure sha512 min=8
```

Change min=8 with whatever password policy length.

Shadow File Password Policy

Change minimum and maximum password ages (most likely set to 0:99999 in the file) I suggest changing those to 1:60 for all entries. . Here is a good example of changing password aging from the the shadow file.

<http://www.cyberciti.biz/faq/understanding-etcshadow-file/>

5. Shutdown unnecessary services

```
netstat -anp | grep LISTEN | grep -v STREAM
```

Analyze the services and the process id/process name. Determine which services to terminate.

```
cd /etc  
find . -print | grep XXX (where XXX is part of the name of the program)
```

For those entries in the "/etc/rc#.d" directory, delete them (rm)

Some suggestions to disable:

a. Remove or disable the "r" commands

This includes rlogind, rshd, rcmd, rexecd, rbootd, rquotad, rstatd, rusersd, rwalld and rexd. These services are inadequately authenticated. It is better to remove these and use SSH and scp instead.

b. Remove or disable fingerd

Remove or disable fingerd if present. Apart from the possibility of a software vulnerability, fingerd allows an attacker to enumerate usernames on the system and to determine the timing and frequency of system administrator logins.

c. Remove or disable tftpd

Tftpd is unauthenticated and not protected against brute-force attacks seeking to enumerate and download files. Do not use tftpd (trivial file transfer protocol) unless unavoidable.

d. Remove or disable telnet

Telnet sends commands unencrypted over the wire. This enables the sniffing of passwords and other information as well as man-in-the-middle attacks. Replace with SSH.

e. Disable SNMP daemon

If present by default, disable any SNMP daemon unless this is really required for the role of the computer.

6. Disable unnecessary boot services.

Some services are needed but not all the time. In the interests of speed and security they should be disabled when not in use. We've created a simple script for this. It can be easily edited and must be run as root. Please see folder titled "Scripts" and look for the "DisableBootServices" script.

```
cd /etc/init or /etc/xinit (should match /etc/init.d)  
cd /etc/init.d (examine the two to make sure they match)  
cd /etc  
find rc*.d | xargs ls -l
```

All entries should be links to the `../init.d` directory. Investigate those that aren't.

```
cd /etc/init or /etc/xinit (should match /etc/init.d)
cd /etc/init.d (examine the two to make sure they match)
cd /etc
find rc*.d | xargs ls -l
```

All entries should be links to the `../init.d` directory. Investigate those that aren't.

Startup scripts (00755 is the norm, but 00700 is ok here as well)
`rc.*` (as `rc.1-6` or `rc1-6.d`) and `/init.d/*` files

```
chmod 0700 /etc/rc*
chmod 0700 /etc/init.d*
```

Here's a good article about services, and runlevels:

<https://www.linux.com/news/enterprise/systems-management/8116-an-introduction-to-services-runlevels-and-rcd-scripts/>

Lock-down user User Sessions:

1. Secure terminals:

The relevant configuration file may be called `/etc/ttys`, `/etc/default/login`, `/etc/security` or `/etc/securetty` depending on the system. See the manual pages for file format and usage information. Check that the `secure` option is removed from any local entries that don't need root login capabilities. The `secure` option should be removed from console if you do not want users to be able to reboot in single user mode. [Note: This does not affect usability of the `su` command.] If it is not already the default, consider using a special group (such as the `wheelgroup` on BSD systems) to restrict which users can use `su` to become root.

2. PATH advice

Check that the current directory `."` is not in the `PATH`. Note that an empty string is interpreted to mean the same as `."` so also make sure the `PATH` does not contain any empty strings. For example, the following `PATH` is insecure:

```
/sbin:/bin:/usr/sbin::/usr/bin
```

This `PATH` advice is especially important for the root account. Including `."` in the `PATH` variable can be used by an attacker to fool a root user into running a malicious binary by substituting `./ls` instead of `/bin/ls` for example.

3. Configure user login sessions to time out automatically.

After a certain period of inactivity, in particular for the root user. To do this, set the appropriate variable in your shell's startup files.

```
typeset -r TMOU=900 (15 minutes = 900 seconds)
```

4. Securing History

```
chattr +a .bash_history (append)
chattr +l .bash_history
```

Users history is being locked and they will have to agree before they use your services.

Lock-down Config files Contents:

1. Analyze DNS – looking for rogue entries

```
vi /etc/resolv.conf
```

Essentially here you should just see the DNS server that the router/modem passed on to your computer, and whatever you have added. Other entries can be considered to be rouge (remember to scroll down). However, before you go and delete your whole file, be sure and lookup the listed server and do your research.

Here is a good link for some basic DNS finding info:

<http://www.cyberciti.biz/faq/how-to-find-out-dns-for-router/>

2. Analyze host files

```
vi /etc/hosts
```

3. Analyze contents of permission files

If you are running , root should have * as the password. If you are running su, it will have a password. Nobody else aside from you and known users should have a password (the big long hash). If they do, make sure they shouldn't be there, and delete that line. Make sure system users have /bin/null set as their shell. Check for rogue users.

```
vi /etc/passwd
vi /etc/shadow
```

Set permissions on sensitive files:

1. Configuration Files

a. Firewall

```
chmod 0700 /etc/profile
chmod 0700 /etc/hosts.allow
chmod 0700 /etc/mtab,
chmod 0700 /etc/utmp
chmod 0700 /var/adm/wtmp (or /var/log/wtmp),
chmod 0700 /etc/syslog.pid (or /var/run/syslog.pid)
```

b. Kernel

```
/etc/sysctl.conf
```


/etc/inittab

c. Users

Make sure the owner & group are set to root.root and the permissions are set to 0644 (except on the /etc/shadow file which should be 400). Here is a good link for permission changing in Linux:

<http://articles.slicehost.com/2010/7/17/checking-linux-file-permissions-with-ls>

```
ls -la /etc/fstab
```

Verify: root.root and -rw-r--r-- (644)

```
ls -la /etc/passwd
```

Verify: root.root and -rw-r--r-- (644)

```
ls -la /etc/shadow
```

Verify: root.root and -rw-r----- (400)

```
ls -la /etc/group
```

Verify: root.root and -rw-r—r-- (644)

```
ls -la /etc/sudoers
```

Verify: root.root and -rw-r—r-- (644)

2. Log Files

(usually located in /var/log/, /var/adm, or var/tmp) are only writable by root.

3. Any World-Writable Files

Ensure that there are no unexpected world writable files or directories on your system. Use the find command to locate these:

```
find / -type d -perm +2 -ls  
chmod 750  
rm
```

5. Set permissions on sensitive binaries

Another good security practice is to set the permissions on certain commands. However, it is **very** important to remember that what you change here depends on what system you're using. Also, the location of binaries will differ based upon the system (for instance /bin, /usr/bin, and /usr/sbin). For instance a server used for development would need the "make" command to be able to be run by any user. Whereas, on a production server it would not be needed. Some examples (you'll need to run these as root):

Set uid:

```
—i / su  
find / \( -perm -2000 \)
```

```
chown root:admin /bin/example
chmod 02750 /bin/example
find / \( -perm -4000 \)
chown root:admin /bin/example
chmod 04750 /bin/su
```

Some Suggestions:

```
Privilege Escalation
  chmod 02750 /bin/su
  chmod 02750 /bin/sudo
Network settings:
  chmod 02750 /bin/ping
  chmod 02750 /sbin/ifconfig
Users On:
  chmod 02750 /usr/bin/w
  chmod 02750 /usr/bin/who
System Configuration
  chmod 02750 /usr/bin/locate
  chmod 02750 /usr/bin/whereis
```

2. Kernel Modules

Ensure that the files holding the kernel and any kernel modules are owned by root, have group ownership set to group id 0 and permissions that prevent them being written to by any non-root users.

To list current module directory:

```
echo "Modules dir: /lib/modules/$(uname -r) for kernel version $(uname -r)"
```

To list contents/permissions of that directory:

```
ls -l /lib/modules/$(uname -r)
```



Upcoming SANS Training

[Click Here for a full list of all Upcoming SANS Events by Location](#)

SANS Secure Caribbean 2026	Kingston, JM	Feb 16, 2026 - Feb 21, 2026	Live Event
SANS Surge 2026	La Jolla, CAUS	Feb 23, 2026 - Feb 28, 2026	Live Event
SANS Dublin February 2026	Dublin, IE	Feb 23, 2026 - Feb 28, 2026	Live Event
SANS Secure Japan 2026	Tokyo, JP	Mar 02, 2026 - Mar 14, 2026	Live Event
SANS London March 2026	London, GB	Mar 02, 2026 - Mar 07, 2026	Live Event
SANS DC Metro March 2026	Arlington, VAUS	Mar 02, 2026 - Mar 07, 2026	Live Event
SANS Secure Singapore 2026	Singapore, SG	Mar 02, 2026 - Mar 14, 2026	Live Event
SANS Paris March 2026	Paris, FR	Mar 09, 2026 - Mar 14, 2026	Live Event
SANS Open-Source Intelligence Summit 2026	Arlington, VAUS	Mar 16, 2026 - Mar 22, 2026	Live Event
SANS Thailand March 2026	Bangkok, TH	Mar 16, 2026 - Mar 21, 2026	Live Event
SANS Amsterdam March 2026	Amsterdam, NL	Mar 16, 2026 - Mar 21, 2026	Live Event
SANS Melbourne March 2026	Melbourne, VIC, AU	Mar 16, 2026 - Mar 21, 2026	Live Event
SANS Cybersecurity Leadership Summit & Training 2026	Arlington, VAUS	Mar 17, 2026 - Mar 22, 2026	Live Event
SANS 2026	Orlando, FLUS	Mar 29, 2026 - Apr 03, 2026	Live Event
SANS London April 2026	London, GB	Apr 13, 2026 - Apr 18, 2026	Live Event
SANS Rome April 2026	Rome, IT	Apr 13, 2026 - Apr 18, 2026	Live Event
SANS Secure Australia 2026	Canberra, ACT, AU	Apr 13, 2026 - Apr 18, 2026	Live Event
SANS Rocky Mountain 2026	Denver, COUS	Apr 20, 2026 - Apr 25, 2026	Live Event
SANS SEC535 at AI Cybersecurity Summit & Training 2026	Arlington, VAUS	Apr 20, 2026 - Apr 27, 2026	Live Event
SANS AI Cybersecurity Summit & Training 2026	Arlington, VAUS	Apr 20, 2026 - Apr 27, 2026	Live Event
SANS Amsterdam April 2026	Amsterdam, NL	Apr 20, 2026 - Apr 25, 2026	Live Event
SANS ICS Munich 2026	Munich, DE	Apr 20, 2026 - Apr 25, 2026	Live Event
SANS Doha April 2026	Doha, QA	Apr 26, 2026 - Apr 30, 2026	Live Event
SANS Cyber Incident Management 2026	OnlineAU	Feb 09, 2026 - Feb 13, 2026	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced