HOST:-

1. Document the host information

Each time you work on a new Linux hardening job, you need to create a new document that has all the checklist items listed in this post, and you need to check off every item you applied on the system. Furthermore, on the top of the document, you need to include the Linux host information:

* Machine name
* IP address
* Mac address
* Name of the person who is doing the hardening (most likely you)
* Date
* Asset Number (If you’re working for a company, then you need to include the asset number that your company uses for tagging hosts.)

2. BIOS protection

You need to protect the BIOS of the host with a password so the end-user won’t be able to change and override the security settings in the BIOS; it’s important to keep this area protected from any changes. Each computer manufacturer has a different set of keys to enter the BIOS mode, then it’s a matter of finding the configuration where you set the administrative password.

Next, you need to disable the booting from external media devices (USB/CD/DVD). If you omit to change this setting, anyone can use a USB stick that contains a bootable OS and can access your OS data.

The latest servers’ motherboards have an internal web server where you can access them remotely. Make sure to change the default password of the admin page or disable it if it’s possible.

3. Hard disk encryption (confidentiality)

Most of the Linux distributions will allow you to encrypt your disks before installation. Disk encryption is important in case of theft because the person who stole your computer won’t be able to read your data if they connect the hard disk to their machine.

In the image below, choose the third option from the list: Guided-use entire disk and set up encrypted LVM (LVM stands for logical volume manager.)



If your Linux distribution doesn’t support encryption, you can go with a software like *TrueCrypt.*



4. Disk protection (availability)

Backups have so many advantages in case of a damaged system, bugs in the OS update. For important servers, the backup needs to be transferred offsite in case of a disaster. Backup needs to be managed as well. For example, how long will you keep the old backups? When do you need to backup your system (every day, every week …)?

Critical systems should be separated into different partitions for:

* */*
* */boot*
* */usr*
* */home*
* */tmp*
* */var*
* */opt*

Portioning disks gives you the opportunity of performance and security in case of a system error. In the picture below, you can see the option of how to separate partitions in Kali Linux during the installation.



5. Lock the boot directory

The boot directory contains important files related to the Linux kernel, so you need to make sure that this directory is locked down to read-only permissions by following the next simple steps. First, open the “fstab” file.



Then, add the last line highlighted at the bottom.



When you finish editing the file, you need to set the owner by executing the following command:

#chown root:root /etc/fstab

Next,

 set few permissions for securing the boot settings:

* Set the owner and group of /etc/grub.conf to the root user:
* *#chown root:root /etc/grub.conf*
* Set permission on the /etc/grub.conf file to read and write for root only:
* *#chmod og-rwx /etc/grub.conf*
* Require authentication for single-user mode:
* *#sed -i "/SINGLE/s/sushell/sulogin/" /etc/sysconfig/init*
* *#sed -i "/PROMPT/s/yes/no/" /etc/sysconfig/init*

6. Disable USB usage

Depending on how critical your system is, sometimes it’s necessary to disable the USB sticks usage on the Linux host. There are multiple ways to deny the usage of USB storage; here’s a popular one:

Open the “blacklist.conf” file using your favorite text editor:

#nano /etc/modprobe.d/blacklist.conf

When the file opens, then add the following line at the end of the file (save and close):

blacklist usb\_storage

After this, open the rc.local file:

#nano /etc/rc.local

Finally, add the following two lines:

modprobe -r usb\_storage

exit 0

SERVER

1. System update

The first thing to do after the first boot is to update the system; this should be an easy step. Generally, you open your terminal window and execute the appropriate commands. In Kali Linux, you achieve this by executing the commands in the picture below:





2. Check the installed packages

List all packages installed on your Linux OS and remove the unnecessary ones. You need to be very strict if the host you’re trying to harden is a server because servers need the least number of applications and services installed on them. Here’s an example of how to list the packages installed on Kali Linux:



Remember that disabling unnecessary services will reduce the attack surface, so it is important to remove the following legacy services if you found them installed on the Linux server:

* Telnet server
* RSH server
* NIS server
* TFTP server
* TALK server

3. Check for open ports

Identifying open connections to the internet is a critical mission. In Kali Linux, I use the following command to spot any hidden open ports:



4. Secure SSH

Yes, indeed SSH is secure, but you need to harden this service as well. First of all, if you can disable SSH, that’s a problem solved. However, if you want to use it, then you have to change the default configuration of SSH. To do it, browse to /etc/ssh and open the “sshd\_config” file using your favorite text editor.



* Change the default port number 22 to something else e.g. 99.
* Make sure that root cannot login remotely through SSH:
* *PermitRootLogin no*
* Allow some specific users:
* *AllowUsers [username]*

The list can go on and on, but these should be enough to start with. For example, some companies add banners to deter attackers and discourage them from continuing further. I encourage you to check the manual of the SSH to understand all the configurations in this file, or you can visit [this site](https://www.ssh.com/ssh/sshd_config/) for more information.

Here are some additional options that you need to make sure exist in the “sshd\_config” file:

* Protocol2
* IgnoreRhosts to yes
* HostbasedAuthentication no
* PermitEmptyPasswords no
* X11Forwarding no
* MaxAuthTries 5
* Ciphers aes128-ctr,aes192-ctr,aes256-ctr
* ClientAliveInterval 900
* ClientAliveCountMax 0
* UsePAM yes

Finally, set the permissions on the sshd\_config file so that only root users can change its contents:

#chown root:root /etc/ssh/sshd\_config

#chmod 600 /etc/ssh/sshd\_config

5. Enable SELinux

Security Enhanced Linux is a Kernel security mechanism for supporting access control security policy. The SELinux has three configuration modes:

* Disabled: Turned-off
* Permissive: Prints warnings
* Enforcing: Policy is enforced

Using a text editor, open the config file:

#nano /etc/selinux/config

And make sure that the policy is enforced:

SELINUX=enforcing



6. Network parameters

Securing your Linux host network activities is an essential task. Don’t always assume that your firewall will take care of everything. Here are some important features to consider for securing your host network:

* **Disable the IP Forwarding** by setting the net.ipv4.ip\_forward parameter to 0 in *“/etc/sysctl.conf”*
* **Disable the Send Packet Redirects** by setting  the net.ipv4.conf.all.send\_redirects and net.ipv4.conf.default.send\_redirects parameters to 0 in*“/etc/sysctl.conf”*
* **Disable ICMP Redirect Acceptance** by setting the net.ipv4.conf.all.accept\_redirects and net.ipv4.conf.default.accept\_redirects parameters to 0 in *“/etc/sysctl.conf”*
* **Enable Bad Error Message Protection** by setting the net.ipv4.icmp\_ignore\_bogus\_error\_responses parameter to 1 in *“/etc/sysctl.conf”*

I strongly recommend using the Linux Firewall by applying the iptable rules and filtering all the incoming, outgoing and forwarded packets. Configuring your iptables rules will take some time, but it’s worth the pain.

7. Password policies

People often [reuse their passwords](https://www.pluralsight.com/blog/it-ops/world-password-day), which is a bad security practice. The old passwords are stored in the file “/etc/security/opasswd”. We are going to use the PAM module to manage the security policies of the Linux host. Under a debian distro, open the file “/etc/pam.d/common-password” using a text editor and add the following two lines:

auth sufficient pam\_unix.so likeauth nullok

password sufficient pam\_unix.so remember=4

(Will not allow users to reuse the last four passwords.)

Another password policy that should be forced is strong passwords. The PAM module offers a pam\_cracklib that protects your server from dictionary and brute-force attacks. To accomplish this task, open the file /etc/pam.d/system-auth using any text editor and add the following line:

/lib/security/$ISA/pam\_cracklib.so retry=3 minlen=8 lcredit=-1 ucredit=-2 dcredit=-2 ocredit=-1

Linux will hash the password to avoid saving it in cleartext so, you need to make sure to define a secure password hashing algorithm SHA512.

Another interesting functionality is to lock the account after five failed attempts. To make this happen, you need to open the file “/etc/pam.d/password-auth” and add the following lines:

auth required pam\_env.so

auth required pam\_faillock.so preauth audit silent deny=5 unlock\_time=604800

auth [success=1 default=bad] pam\_unix.so

auth [default=die] pam\_faillock.so authfail audit deny=5 unlock\_time=604800

auth sufficient pam\_faillock.so authsucc audit deny=5 unlock\_time=604800

auth required pam\_deny.so

We’re not done yet; one additional step is needed. Open the file “/etc/pam.d/system-auth” and make sure you have the following lines added:

auth required pam\_env.so

auth required pam\_faillock.so preauth audit silent deny=5 unlock\_time=604800

auth [success=1 default=bad] pam\_unix.so

auth [default=die] pam\_faillock.so authfail audit deny=5 unlock\_time=604800

auth sufficient pam\_faillock.so authsucc audit deny=5 unlock\_time=604800

auth required pam\_deny.so

After five failed attempts, only an administrator can unlock the account by using the following command:

# /usr/sbin/faillock --user <userlocked> --reset

Also, another good practice is to set the password to expire after 90 days, to accomplish this task you need to:

* Set the PASS\_MAX\_DAYS parameter to 90 in “/etc/login.defs”
* Change the active user by executing the following command :
* *#chage --maxdays 90 <user>*

The next tip for enhancing the passwords policies is to restrict access to the su command by setting the pam\_wheel.so parameters in “/etc/pam.d/su”:

auth required pam\_wheel.so use\_uid

The final tip for passwords policy is to disable the system accounts for non-root users by using the following bash script:

#!/bin/bash

for user in `awk -F: '($3 < 500) {print $1 }' /etc/passwd`; do

if [ $user != "root" ]

then

/usr/sbin/usermod -L $user

if [ $user != "sync" ] && [ $user != "shutdown" ] && [ $user != "halt" ]

then /usr/sbin/usermod -s /sbin/nologin $user

fi

fi

done

8. Permissions and verifications

Prepare yourself mentally because this is going to be a long list. But, permissions is one of the most important and critical tasks to achieve the security goal on a Linux host.

Set User/Group Owner and Permission on “/etc/anacrontab”, “/etc/crontab” and “/etc/cron.\*” by executing the following commands:

#chown root:root /etc/anacrontab

#chmod og-rwx /etc/anacrontab

#chown root:root /etc/crontab

#chmod og-rwx /etc/crontab

#chown root:root /etc/cron.hourly

#chmod og-rwx /etc/cron.hourly

#chown root:root /etc/cron.daily

#chmod og-rwx /etc/cron.daily

#chown root:root /etc/cron.weekly

#chmod og-rwx /etc/cron.weekly

#chown root:root /etc/cron.monthly

#chmod og-rwx /etc/cron.monthly

#chown root:root /etc/cron.d

#chmod og-rwx /etc/cron.d

Set the right and permissions on “/var/spool/cron” for “root crontab”

#chown root:root <crontabfile>

#chmod og-rwx <crontabfile>

Set User/Group Owner and Permission on “passwd” file

#chmod 644 /etc/passwd

#chown root:root /etc/passwd

Set User/Group Owner and Permission on the “group” file

#chmod 644 /etc/group

#chown root:root /etc/group

Set User/Group Owner and Permission on the “shadow” file

#chmod 600 /etc/shadow

#chown root:root /etc/shadow

Set User/Group Owner and Permission on the “gshadow” file

#chmod 600 /etc/gshadow

#chown root:root /etc/gshadow

9. Additional process hardening

For this last item in the list, I’m including some additional tips that should be considered when hardening a Linux host.

First, Restrict Core Dumps by:

* Adding hard core 0 to the “/etc/security/limits.conf” file
* Adding fs.suid\_dumpable = 0 to the “/etc/sysctl.conf” file

Second, configure Exec Shield by:

* Adding kernel.exec-shield = 1 to the “/etc/sysctl.conf” file

Third, enable randomized Virtual Memory Region Placement by:

* Adding kernel.randomize\_va\_space = 2 to the “/etc/sysctl.conf” file