


# Linux Hardening

By: Negin Nourollahimoghadam (Nian Nour)

Date: September 2024



## **Why do we need to study Linux security? Isn't Linux already secure?**

There are several threats on Linux server such as:

- Botnet malware
- Ransomware
- Cryptocoin mining software
- Security breaches



# Linux Security Tips

Most users consider Linux an excellent system with a high degree of security.

However, to ensure your servers operate safely and effectively, you still need to follow recommendations.



## 1. Keep the System Up to Date

- Failing to keep your server updated can leave it vulnerable to attacks.
- Consider automating the update process to ensure you don't miss essential security updates.
- *KernelCare Enterprise*, for example, can help you keep your systems continuously patched from the latest vulnerabilities without the need for them to be rebooted.



## 2. Remove Unnecessary Software

- The more packages, software, and third-party repositories you add, the greater the attack surface and the potential for security vulnerabilities.
- Regularly maintaining your software environment reduces the risk of security breaches and ensures optimal server performance.



### 3. Check and Close Open Ports

- Regularly scan for open ports and close any that are not in use.
- Tools like *netstat* can help you identify open ports and promptly secure your server by closing them.



## 4. Regularly Create and Maintain Backups

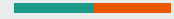
- Backups are essential to the security of any system. With backups, you can restore important data in the event of any server intrusion.
- In Linux, the *Rsync* application is a popular choice for data backups. It includes several options that allow you to create daily backups.



## 5. Create an SSH Key Pair

- Secure Shell (SSH) key pairs offer a more secure means of accessing your servers.
- SSH keys make brute-force attacks significantly more challenging. They rely on cryptographic key pairs, which are much more secure than passwords.





# Linux Security Mechanisms



## 1. POSIX

- POSIX (Portable Operating System Interface) is a standard that defines a file system permission model for Linux.
- POSIX associates three sets of permissions with each file system object: owner, group, and other.
- Each set can contain read, write, and execute permissions.



## 2. SELinux

- It can reduce the potential damage caused by vulnerabilities in applications and system services.
- It isolates and secures system components
- It is useful in environments where applications and data need to be kept separate, such as cloud servers, publicly accessible servers, and government use.



### 3. AppArmor

AppArmor, or Application Armor helps protecting systems by limiting what applications can do.

- AppArmor helps containing security threats by separating applications from each other and the rest of the system.
- AppArmor can prevent attacks even if it's exploiting previously unknown vulnerabilities



## 4. Address Space Layout Randomization

- It is a security feature in Linux that helps protecting against buffer overflow attacks
- ASLR randomizes the location of a process's code and data in virtual memory, making it harder for attackers to predict the next instruction's memory address.



## Securing Users Accounts

- Managing users is one of the most challenging aspects of IT administration.
- *The question is “How do we keep our users accounts safe in Linux server?”.*



## 1. Be careful with the root user

- The all-powerful administrator account on a Linux system is the *root account*.
- Always logging in as the root user can bring lots of security problems.
- It makes it easier for *you or someone else* to accidentally perform an action that causes damage to the system



## 1. Be careful with the root user (cont.)

- What we need is a mechanism that allows users to perform administrative tasks without incurring the risk of having them always log on as the root user.
- This mechanism in Linux is in the form of the *sudo utility*.





## 2. The advantages of using sudo

- Assign full administrative privileges to certain users.
- Assign only necessary privileges to other users.
- Allow users to perform necessary administrative tasks with their normal user passwords.
- Create sudo policies for an entire enterprise network.
- Improve auditing capabilities



## 2.1 Setting up sudo privileges for full administrative users

- Before we look at how to limit what users can do, let's first look at how to allow a user to do everything, including logging in to the root command prompt.



## 2.2 Sudo Group

- In Debian Family, by doing *sudo visudo*, we will open the sudo policy file which contains:

```
# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@include /etc/sudoers.d
```

1- sudo visudo



## 2.2 Sudo Group (cont.)

```
%sudo ALL=(ALL:ALL) ALL
```

- This line means that members of sudo group can perform any command, as any user, on any machine in the network on which this policy is deployed.



## 2.3 Adding an Existing User to the Sudo Group

```
sudo usermod -a -G sudo user1
```

- The `-a` option appends the user to the group without removing them from any other groups they belong to.



## 2.4 Creating a New User and Adding Them to the Sudo Group

- You can also add a user account to the sudo group as you create it.

```
sudo useradd -G sudo createduser1
```

- The line above creates a new user (for example, *createduser1*) and add them to the sudo group at the same time.



### 3. Additional Notes

- When you create a new user you have to set a password and create a Home Directory for them.

1. To create Home Directory use the command below:

```
sudo useradd -G sudo -m -d /home/frank -s /bin/bash frank
```

1. To set password use the command below:

```
sudo passwd createduser1
```



## 4. Delegation of Privileges

- IT security philosophy is to give network users enough privileges so that they can get their jobs done, but no privileges beyond that.
- With sudo, you can delegate these privileges and disallow users from doing any other administrative jobs that don't fit their job description.





## 5. Set up User Permissions Step By Step

1. **Open *visudo*** and add command lines based on your needs.
2. **Define user alias** for different groups of people such as BackupAdmins, WebAdmins and etc.

The command is:

*User\_Alias SOFTWAREADMINS = user1,user2*



## 5. Set up User Permissions Step By Step (cont.)

**3. Define command alias** for software management commands. For instance:

The command is:

*Cmnd\_Alias SOFTWARE = /bin/rpm, /usr/bin/up2date, /usr/bin/yum*



## 5. Set up User Permissions Step By Step (cont.)

### 4. **Assign** command alias to the user alias

The command is:

```
SOFTWAREADMINS ALL=(ALL) NOPASSWD: SOFTWARE
```



## 5. Set up User Permissions Step By Step (cont.)

- User1 and user2, are both members of the *SOFTWAREADMINS* user alias, can now run the rpm, up2date, and yum commands with root privileges, on all servers on which this policy is installed.



## 5. Command Exceptions

- Some commands cover a lot of different functions, therefore you have to be careful with them. You have 2 choice:
  1. Eliminate all of the subcommands and let users use all the functions.  
For example:

*Cmnd\_Alias SERVICES = /usr/bin/systemctl*

*\*\* (systemctl covers a lot of functions)\*\**



## 5. Command exceptions (cont.)

2. Add a wildcard to each of their subcommands. For example:

```
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig,  
/usr/bin/systemctl start *, /usr/bin/systemctl stop *, /usr/bin/systemctl  
reload *, /usr/bin/systemctl restart *, /usr/bin/systemctl status *,  
/usr/bin/systemctl enable *, /usr/bin/systemctl disable *
```

- Now, users can perform any of the systemctl functions that are listed in this command alias, for any service.



## 5. Set up User Permissions Step By Step (cont.)

- In the end, when a user types a command into a server on the network, sudo will first look at the hostname of that server. If the user is authorized to perform that command on that server, then sudo allows it. Otherwise, sudo denies it.



## 6. Enforcing strong password criteria

- Passwords nowadays can be :
  - passphrases that are long, yet easy to remember.
  - changed only if they've been breached.
- However, you may need to follow some principles for setting passwords in a huge company.



## 6. Enforcing strong password criteria (cont.)

First, install *libpam-pwquality*.

Then open  
*/etc/security/pwquality.conf*

2-  
*/etc/security/pwquality.conf*

```
# Configuration for systemwide password quality limits
# Defaults:
#
# Number of characters in the new password that must not be present in the
# old password.
# difok = 1
#
# Minimum acceptable size for the new password (plus one if
# credits are not disabled which is the default). (See pam_cracklib manual.)
# Cannot be set to lower value than 6.
# minlen = 8
#
# The maximum credit for having digits in the new password. If less than 0
# it is the minimum number of digits in the new password.
# dcredit = 0
#
# The maximum credit for having uppercase characters in the new password.
# If less than 0 it is the minimum number of uppercase characters in the new
# password.
# ucredit = 0
#
# The maximum credit for having lowercase characters in the new password.
# If less than 0 it is the minimum number of lowercase characters in the new
# password.
# lcredit = 0
#
# The maximum credit for having other characters in the new password.
# If less than 0 it is the minimum number of other characters in the new
# password.
```



## 6. Enforcing strong password criteria (cont.)

- You can make any new principle for passwords by uncommenting the lines and changing their values.



## 7. Setting and enforcing password and account expiration

- When someone's *password expires*, he or she can change it, and everything will be all good.
- If somebody's *account expires*, only someone with the proper admin privileges can unlock it.

## 7. Setting and enforcing password and account expiration (cont.)

- By using *chage -l*, every user can take a look at their expiry data.

```
(nian@kali) [ ~/Desktop ]  
$ chage -l nian  
Last password change           : Jul 07, 2024  
Password expires               : never  
Password inactive              : never  
Account expires                : never  
Minimum number of days between password change : 0  
Maximum number of days between password change : 99999  
Number of days of warning before password expires : 7
```

3- chage -l information



## 7. Setting expiry data on a per-account basis with chage

- You can use *chage* for setting either an account expiration or a password expiration.



## 8. Setting expiry data on a per-account basis with chage

- Here is a guideline for using chage:

| Option | Explanation   |
|--------|---|
| -d     | if you use the -d 0 option on someone's account, you'll force the user to change his or her password on their next login. |
| -E     | It sets the expiration date for the user account.   |
| -I     | It sets the number of days before an account with an expired password will be locked out                                  |



## 8. Setting expiry data on a per-account basis with chage (cont.)

| Option | Explanation  |
|--------|--|
| -m     | This sets the minimum number of days between password changes. In other words, if Charlie changes his password today, the -m 5 option will force him to wait five days before he can change his password again.                              |
| -W     | This will set the number of warning days for passwords that are about to expire.   |
| -M     | This sets the maximum number of days before a password expires. (Be aware, though, that if Charlie last set his password 89 days ago, using a -M 90 option on his account will cause his password to expire tomorrow, not 90 days from now.) |



## 8. Setting expiry data on a per-account basis with chage (cont.)

For example:

```
sudo chage -E 2021-02-28 -I 4 -m 3 -M 90 -W 4 charlie
donnie@ubuntu-steemnode:~$ sudo chage -l charlie
Last password change : Oct 06, 2019
Password expires : Jan 04, 2020
Password inactive : Jan 08, 2020
Account expires : Feb 28, 2021
Minimum number of days between password change : 3
Maximum number of days between password change : 90
Number of days of warning before password expires : 4
donnie@ubuntu-steemnode:~$
```





## 9. Preventing brute-force password attacks

- Setting a lockout value of three failed login attempts will do three things:
  - It will unnecessarily frustrate users.
  - It will cause extra work for help desk personnel.
  - If an account really is under attack, it will lock the account before you've had a chance to gather information about the attacker.



## 9. Preventing brute-force password attacks (cont.)

- Setting the lockout value to something more realistic, such as 100 failed login attempts, will still provide good security, while still giving you enough time to gather information about the attackers.
- To set this value we use *pam\_tally2* module.



## 9. Preventing brute-force password attacks (cont.)

- You can add the command below as an example to change the lockout value of failed login */etc/pam.d/login* file.

*auth required pam\_tally2.so deny=4 even\_deny\_root unlock\_time=1200*



## 10. Locking user accounts

- There are two utilities that you can use to temporarily lock a user account:
  - Using *usermod* to lock a user account
  - Using *passwd* to lock user accounts



## 10.1 Using usermod to lock a user account

- To lock accounts:

*sudo usermod -L user1*

- To unlock accounts:

*Sudo usermod -U user1*



## 10.2 Using passwd to lock user accounts

- To lock accounts:

```
sudo passwd -l user1
```

- To unlock accounts:

```
sudo passwd -u user1
```



## Linux Firewall

- The name of the Linux firewall is netfilter, and every Linux distro has it built in.
- However, we use *iptables* which is just one of several command-line utilities to manage netfilter.



## 1. Iptables

- Iptables consists of five tables of rules, each table consists of chains of rules:
  - Filter table
  - Network Address Translation (NAT) table
  - Mangle table
  - Raw table
  - Security table





## 1.1 Filter table

- The filter table consists of three chains:
  1. INPUT
  2. FORWARD
  3. OUTPUT



## 1.1 Filter table (cont.)

- First, look at your current configuration by using the *sudo iptables -L* command:

```
Chain INPUT (policy ACCEPT)
target port opt source destination
Chain FORWARD (policy ACCEPT)
target port opt source destination
Chain OUTPUT (policy ACCEPT)
target port opt source destination
```

4- Parts of filter  
table



## 1.1 Filter table (cont.)

- As you may have seen, there is no rule for filter table in Ubuntu/Debian systems. You have to configure them yourself.



## 1.2 Filter table commands

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

- This command sets up a rule in the firewall that allows incoming packets from servers that your host has already connected to or that are related to those connections.



## 1.2 Filter table commands (cont.)

```
sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

- The entire command effectively allows incoming TCP connections on port 22 (SSH) to your server. By running this command, you're telling iptables to accept any incoming SSH traffic, which is essential for remote management and access.



## 1.2 Filter table commands (cont.)

```
sudo iptables -A INPUT -p tcp --dport 53 -j ACCEPT
```

```
sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
```

- Together, these commands allow incoming DNS traffic over both TCP and UDP protocols on port 53.



## 1.2 Filter table commands (cont.)

```
sudo iptables -I INPUT 1 -i lo -j ACCEPT
```

- The command allows all incoming traffic on the loopback interface, ensuring that local applications can communicate without being hindered by firewall rules.

## 1.2 Filter table commands (cont.)

In the end our ruleset for INPUT chain will look like this:

```
Chain INPUT (policy ACCEPT)
target    prot opt source                destination
ACCEPT    all  --  anywhere              anywhere
ACCEPT    all  --  anywhere              anywhere    ctstate
RELATED,ESTABLISHED
ACCEPT    tcp  --  anywhere              anywhere    tcp dpt:ssh
ACCEPT    tcp  --  anywhere              anywhere    tcp
dpt:domain
ACCEPT    udp  --  anywhere              anywhere    udp
dpt:domain
```

5- Input ruleset





## Resources:

- Mastering Linux Security and Hardening by Donald A. Tevault