

# Linux Security: What Everyone Talks About but Few Know How to Implement

Check GitHub for helpful DevOps tools:

## Michael Robotics

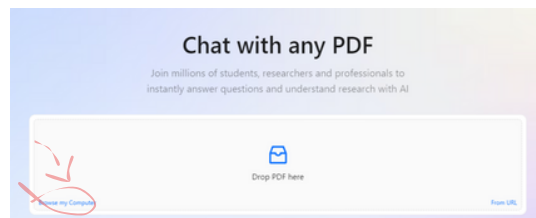
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>



Ask Personal AI Document assistant to learn interactively (FASTER)!

- 1 Download PDF
  - 2 Go to website
  - 3 Browse file
  - 4 Chat with Document
- 1 <https://github.com/MichaelRobotics/DevOpsTools/blob/main/LinuxSecurity.pdf>
- 2 | Click there to go to ChatPdf website
- 3
- 4
- Ask questions about document!



# Completly new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

 <https://www.hackthebox.com/>



## What is Linux Security?

Linux security involves using built-in features and additional configurations to protect systems from unauthorized access, threats, and vulnerabilities. By customizing and tuning Linux's security model, administrators can enhance protection against potential cyberattacks.

## How to implement Linux Security?

Implementing Linux security includes configuring settings such as GRUB passwords, setting up firewalls, and disabling unnecessary services. Regularly updating the system and monitoring network ports also help to minimize vulnerabilities.

# Linux Security: Why and When

Linux security is essential to protect sensitive data and maintain system integrity, especially for publicly accessible servers or production environments.

For instance, securing SSH access by disabling root login and changing the default port prevents unauthorized remote access, reducing the risk of automated attacks.

## System Requirements

- 1 GB RAM (minimum, 2 GB recommended for better performance)
- 10 GB free storage (more may be required depending on the size of the DNS zone files and logs)
- Ubuntu 22.04

## Linux Security: Main components & packages

- SELinux – Enforces mandatory access control policies.
- firewalld / ufw – Manages incoming/outgoing traffic with firewall rules.
- SSH – Secure remote server access with encryption.
- PAM – Manages authentication policies (e.g., passwords).
- AppArmor – Restricts programs with security profiles.
- Fail2ban – Blocks IPs after failed login attempts.
- Snort – Network intrusion detection system (IDS).

# Linux Security: Updates

Patching Vulnerabilities: Vulnerabilities in software are inevitable, and Linux is no exception. Security flaws or "exploits" are frequently discovered in Linux kernels, libraries, or applications. These vulnerabilities, if left unpatched, can be exploited by attackers to gain unauthorized access, execute arbitrary code, or compromise sensitive data

Regularly update your Linux distribution and all installed packages to the latest security patches and bug fixes using your system default package manager, such as apt for Debian-based distributions

Update package list

```
sudo apt-get update
```

Upgrade all installed packages

```
sudo apt-get upgrade -y
```

Clean up unnecessary packages and files

```
sudo apt-get autoremove -y sudo apt-get autoclean
```

# Linux Security: SSH and fail2ban

## 1) Basic secure setup

Secure Shell (SSH) is a widely used protocol that offers a secure way to access and manage your Linux servers. However, to maximize security, there are several best practices you should follow, such as disabling root login, allowing only specific users and changing the default SSH port.

Disabling root login forces users to log in with their user accounts, providing better accountability and reducing the risk of unauthorized access. Enter conf file

```
$ sudo nano /etc/ssh/sshd_config
```

Find the line that says PermitRootLogin and change its value to no:

```
$ PermitRootLogin no
```

Restricting SSH access to specific users adds a layer of security by ensuring that only authorized users can log in.

Add a line at the end of the file to specify the allowed users:

```
$ AllowUsers user1 user2
```

Changing the **default SSH port (22)** to a non-standard port can help reduce the number of automated attacks on your server. Restart the SSH service to apply the changes

```
$ Port 2233
```

## 2) Display SSH Banner Before Login

Displaying a banner message before the SSH login prompt can be a useful way to provide legal notices, warnings, or information to users attempting to access your Linux server.

To set such banners, you need to create a text file that contains the message you want to display.

```
$ sudo nano /etc/ssh/ssh-banner
```

Add your banner message:

```
*****  
WARNING: Unauthorized access to this system is prohibited.  
  
All activities on this system are logged and monitored. By logging in,  
you acknowledge that you are authorized to access this system and agree  
to abide by all relevant policies and regulations.  
  
Unauthorized users will be prosecuted to the fullest extent of the law.  
*****
```

Next, you need to configure the SSH server to display the banner before the login prompt.

```
$ sudo nano /etc/ssh/sshd_config
```

Find and modify the **Banner** directive and After making these changes, restart the SSH service to apply the new configuration.

```
$ sudo systemctl restart sshd
```

Find the line containing PubkeyAuthentication and set it to yes.

```
$ PubkeyAuthentication yes
```

This setting enables public key authentication, which allows users to authenticate using SSH keys instead of passwords. The server checks if the connecting user has the corresponding private key to the authorized public key stored on the server.

If public key and private key is stored on remote server, you can download its private key in .pem format. Store this key in local machine and use, when connection to remote server is needed:

```
$ ssh -i /path/to/your-key.pem username@server-ip
```

Optionally save this ssh configuration in **~/.ssh/config**:

```
Host myserver
  HostName 192.168.1.3
  User user2
  IdentityFile /path/to/your-key.pem
  Port 2222
```

Then you can connect to host using “myserver”

```
$ ssh myserver
```

#### 4) Use Fail2Ban as

Fail2ban is one of the most popular open-source intrusion detection/prevention frameworks written in a python programming language. It operates by scanning log files such as `/var/log/secure`, `/var/log/auth.log`, `/var/log/pwdfail` etc. for too many failed login attempts.

Fail2ban is used to update Netfilter/iptables or TCP Wrapper's `hosts.deny` file, to reject an attacker's IP address for a set amount of time. It also has the ability to unban a blocked IP address for a certain period of time set by administrators. However, a certain minute of unbanning is more than enough to stop such malicious attacks.

All configuration files are in **`/etc/fail2ban`**

Configuration file examples and defaults are in two main files

**`/etc/fail2ban/fail2ban.conf`** and **`/etc/fail2ban/jail.conf`**

Best Config - **`/etc/fail2ban/jail.local`**

```
[DEFAULT]
ignoreip = 127.0.0.1/8 ::1
bantime = 3600
findtime = 600
maxretry = 5
[sshd]
enabled = true
```



Now in more complex service environments add services and programs like **ssh-jail.conf** to the **/etc/fail2ban/jail.d/** directory. Any program that hackers use is typically always under watch, like WordPress installs for example:

**/etc/fail2ban/jail.d/wordpress.conf**

```
[DEFAULT]
ignoreip = 127.0.0.1/8 ::1
bantime = 3600
findtime = 600
maxretry = 5
[sshd]
enabled = true
```

Enabling Fail2Ban

```
sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

# Linux Security: Firewall

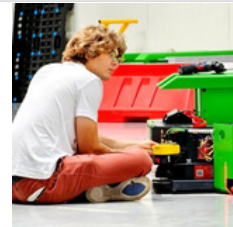
## 1) ufw

Check basic firewall configuration, from my UFW (uncomplicated firewall) PDF!

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>



## 2) Disabling IPv6 Protocol on a Linux Server

If you're not using an IPv6 protocol, then you should disable it because most of the applications or policies do not require IPv6 protocol and currently, it isn't required on the server.

You need to edit the network configuration file to disable IPv6.

```
$ sudo nano /etc/sysctl.conf
```

Add the following lines to the end of the file:

```
# Disable IPv6

net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

Next, apply the changes to the running system.

```
$ sudo sysctl -p
```

### 3) Check Listening Network Ports in Linux

Monitoring and managing network ports is a crucial aspect of securing a Linux system and knowing which ports are open and listening can help you identify potential vulnerabilities and ensure that only necessary services are accessible.

```
$ sudo netstat -tuln  
OR  
sudo ss -tuln
```

### 4) Ignoring ICMP or Broadcast Requests in Linux

In Linux, you can configure your system to ignore ICMP (Internet Control Message Protocol) or broadcast requests to enhance security and reduce unwanted network traffic.

Open the `/etc/sysctl.conf` file using a text editor:

```
sudo nano /etc/sysctl.conf
```

Add or modify the following line to set the ICMP echo ignore flag:

```
net.ipv4.icmp_echo_ignore_all=1
```

Apply the changes:

```
sysctl -p
```

# Linux Security: Repos, services & processes

## 1) Create partitions for data security

It's important to have different partitions to obtain higher data security in case any disaster happens. By creating different partitions, data can be separated and grouped.

When an unexpected accident occurs, only data from that partition will be damaged, while the data on other partitions will survive. Make sure you have the following separate partitions and that third-party applications should be installed on separate file systems under /opt.

Most Linux distributions allow you to create and configure partitions during the installation process using the guided or manual partitioning options.

Post-installation, you can use tools like fdisk, parted, or graphical tools like GParted to create and manage partitions.

Example using fdisk:

```
sudo fdisk /dev/sda
```

## 2) Minimize Packages to Minimize Vulnerability: Remove Unwanted Services

One of the key strategies in securing a Linux system is to minimize the number of installed packages and running services. Each package and service can potentially introduce vulnerabilities, so keeping your system lean and efficient is a crucial step in hardening your server.

Start by identifying the packages and services that are not needed for your server's specific function, which can be done using package management tools such as `dpkg` and service management utilities.

```
dpkg --get-selections | grep install  
systemctl list-units --type=service --state=running
```

Once you have identified the unnecessary packages, you can remove them using your package manager such as apt or.

```
sudo apt remove package_name
```

After removing the unwanted packages, the next step is to disable and stop services that are not needed.

```
sudo systemctl stop service_name  
sudo systemctl disable service_name
```

### 3) Repo update priority

Many users add numerous repositories to their Linux installation without careful consideration. While adding repositories is necessary to get specific programs, doing so without setting priorities can lead to security risks or system instability by allowing packages from third-party repos to override those from the base distribution. It's crucial to configure priorities so that these additional repositories are only used when a package isn't available in the base repo. Otherwise, unwanted packages from added repos may overwrite important system packages, causing potential problems

```
sudo apt remove package_name
```

All priority preference files are stored in the **/etc/apt/preferences.d/** directory.

Example: volian.pref

```
Package: *  
Pin: origin deb.volian.org  
Pin-Priority: 100
```

- Set specific packages to only install with the Package: \* Line
- Pin: Origin is the address of the repo
- Pin-Priority is generally 100 which means it will update packages NOT in base repos.
  - 1 = do not auto update
  - 100 = update if not in other repos
  - over 100 = overwrite base repos

# Linux Security: Hardware & General & Backups

## 1) Disable **ctrl+alt+del**

In most Linux distributions, pressing 'CTRL-ALT-DELETE' will take your system to the reboot process. So, it's not a good idea to have this option enabled at least on production servers, if someone mistakenly does this.

To disable Ctrl+Alt+Delete, create or edit the override file for the Ctrl+Alt+Delete key combination.

```
sudo systemctl edit ctrl-alt-del.target
```

Add the following lines to the override file to disable the key combination:

```
sudo systemctl edit ctrl-alt-del.target
```

Add the following lines to the override file to disable the key combination:

```
[Service]
ExecStart=
```

After making changes, reload the systemd configuration and mask the ctrl-alt-del.target ensures that it cannot be triggered:

```
sudo systemctl daemon-reload
sudo systemctl mask ctrl-alt-del.target
```

## 2) Implement backups

Backing up files in Linux using rsync is an efficient and reliable method, as it synchronizes files and directories between two locations, making it perfect for backups.

To back up files on the local system, use the following command:

```
rsync -av --delete /source/directory/ /backup/directory/
```

To back up files on the remote system, use the following command:

```
rsync -avz -e ssh /source/directory/ user@remote_host:/backup/directory/
```

example local backup:

```
0 2 * * * rsync -av --delete /home/user/data/ /home/user/backup/
```

example remote backup

```
0 2 * * * rsync -avz -e ssh /home/user/data/ user@remote_host:/remote/backup/
```



### 3) Keep /boot as Read-Only

The /boot directory in Linux contains essential files needed to boot the operating system, such as the kernel, initial ramdisk (initrd), and bootloader configuration files. Ensuring that /boot is mounted as read-only can enhance system security and integrity by preventing unauthorized modifications to these critical files. To do this, open “**/etc/fstab**” file.

```
nano /etc/fstab
```

Add the following line at the bottom, save, and close it.

```
LABEL=/boot /boot ext4 defaults,ro 1 2
```

Please note that you need to reset the change to read-write if you need to upgrade the kernel in the future.

### 4) Disable USB Storage Detection

Many times it happens that we want to restrict users from using USB stick in systems to protect and secure data from stealing.

Create a file ‘**nano /etc/modprobe.d/no-usb.conf**’ and adding the below line will not detect USB storage.

```
blacklist usb_storage
```

After creating the blacklist file, update the initramfs (initial RAM filesystem) to ensure the blacklisted module is not loaded during the boot process:

```
sudo update-initramfs -u
```

Reboot your system for the changes to take effect.

## 5) Physical System Security – Setting a GRUB Password

One effective way to enhance security is by setting a GRUB password, which is a boot loader used by most Linux distributions to load the operating system when the computer starts up.

By setting a GRUB password, you add an extra layer of defense against unauthorized users who might attempt to tamper with or gain unauthorized access to your system. Log into your Linux server and open the GRUB configuration file, which is located in **/etc/default/grub**

Next, use a text editor to edit this file with root privileges (sudo).

Look for the line that starts with GRUB\_CMDLINE\_LINUX or similar and append GRUB\_PASSWORD=<password> to the end of this line.

```
GRUB_CMDLINE_LINUX="quiet splash"  
GRUB_PASSWORD=password123
```

After editing the configuration file, update GRUB to apply the changes.

```
sudo update-grub
```

Restart your server to apply the GRUB password.

```
sudo reboot
```

## 6) Enforcing Stronger Passwords

Many users use soft or weak passwords and their passwords might be hacked with dictionary-based or brute-force attacks.

The 'pam\_cracklib' module is available in the PAM (Pluggable Authentication Modules) module stack which will force users to set strong passwords.

Open the following file with an editor.

```
nano /etc/pam.d/system-auth
```

And add a line using credit parameters such as (**lcredit**, **ucredit**, **dcredit**, and/or **ocredit** respectively lower-case, upper-case, digit, and other)

```
/lib/security/$ISA/pam_cracklib.so retry=3 minlen=8 lcredit=-1 ucredit=-2  
dcredit=-2 ocredit=-1
```

## 7) Restricting Users from Using Old Passwords on Linux

This is very useful if you want to disallow users to use the same old passwords. The old password file is located at `/etc/security/opasswd`. This can be achieved by using the **PAM** module.

Open the '**/etc/pam.d/common-password**' file

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

Add the following line to the 'auth' section.

```
auth    sufficient    pam_unix.so likeauth nullok
```

Add the following line to the 'password' section to disallow a user from re-using the last 5 passwords

```
password    sufficient    pam_unix.so nullok use_authtok md5 shadow remember=5
```

Only the last 5 passwords are remembered by the server. If you try to use any of the last 5 old passwords, you will get an error like.

```
Password has been already used. Choose another.
```

## 8) Managing Cron Job Permissions

Cron is a powerful utility in Unix-like operating systems that allows users to schedule jobs to run at specific intervals.

However, there might be situations where you need to control who can or cannot create and run cron jobs on your system. Cron has built-in features to manage these permissions using the **/etc/cron.allow** and **/etc/cron.deny** files.

To allow specific users to use cron, edit **/etc/cron.allow** file and the usernames of the users you want to deny, one per line.

```
user1  
user2
```

To deny specific users to use cron, edit **/etc/cron.deny** file and the usernames of the users you want to allow, one per line.

```
user3  
user4
```

# Linux Security: Monitor logs

Reviewing logs on a regular basis is an important part of managing and securing a Linux system, as logs provide detailed records of system events, user activities, and potential security incidents.

- **/var/log/message** – Where whole system logs or current activity logs are available.
- **/var/log/auth.log** – Authentication logs.
- **/var/log/kern.log** – Kernel logs.
- **/var/log/cron.log** – Crond logs (cron job).
- **/var/log/maillog** – Mail server logs.
- **/var/log/boot.log** – System boot log.
- **/var/log/mysqld.log** – MySQL database server log file.
- **/var/log/secure** – Authentication log.
- **/var/log/utmp** or **/var/log/wtmp** : Login records file.

Use Zabbix. Its an Open Source, high-level enterprise software designed to monitor and keep track of networks, servers, and applications in real-time. Build in a server-client model, Zabbix can collect different types of data that are used to create historical graphics and output performance or load trends of the monitored targets.

**Check PDF I've made about Monitoring & ZABBIX on Linux!**

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>



# Linux Security: Selinux & Apparmor

Many Linux distributions come with security tools like AppArmor or SELinux pre-installed by default. AppArmor and SELinux provide a significant layer of security by sandboxing and restricting programs, even when they have elevated privileges. Without these tools, systems are more vulnerable, as they limit program access by default to areas like the home folder and hardware access unless customized otherwise.

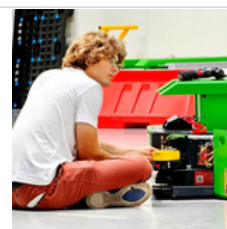
When installed, these tools typically operate in "complain" mode for AppArmor or "permissive" mode for SELinux, which allows them to log potential security issues without actively preventing harm. While this setup offers basic protection, it does not fully stop malicious programs from causing damage. Moving these tools into "enforce" mode—where they actively block unauthorized access—requires user expertise in creating or modifying security profiles for individual programs. This process can be complex and is often neglected in most Linux desktop environments, leaving users without optimal protection unless profiles are properly configured.

## Check PDF I've made about Apparmor & SELinux!

### Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>



# Linux Security: Use IPS

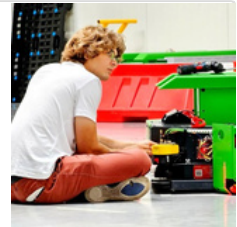
To enhance network security, install and configure an Intrusion Detection System (IDS) or Intrusion Prevention System (IPS) to monitor traffic and detect potential attacks. IDS options like Snort and Suricata analyze network packets for suspicious activity and provide alerts. Snort offers flexible, rule-based detection, while Suricata supports multi-threaded processing and advanced protocols.

**Check PDF I've made about Snort!**

## Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>




## Learn more about Linux Security

**Check TecMint and Chris Linux Tech, they have great docs!**

The 3 Biggest Security Mistakes Linux Users Make


Security is a journey, not a destination

 <https://christitus.com/linux-security-mistakes/>



26 Security Hardening Tips for Modern Linux Servers

Everybody says that Linux is secure by default, and to some extent

 <https://www.tecmint.com/linux-server-hardening-security-tips/>



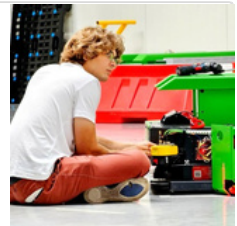
## Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

**Check my GitHub**

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

<https://github.com/MichaelRobotics>



*PS.*

*If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!*