Linux ADMIN: Have full control over Who can access your system

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

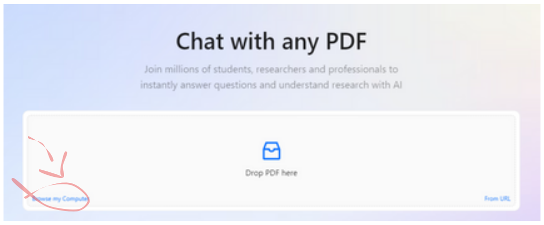
3 Browse file

4 Chat with Document

1 https://github.com/MichaelRobotics/DevOpsTools/blob/main/LinuxADMIN.pdf

Click there to go to ChatPdf website 2

3



Chat with any PDF

Join millions of students, researchers and professionals to instantly answer questions and understand research with AI

Ask questions about document! 4

Complety 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 Administration?

Linux administration involves managing and maintaining Linux-based systems, ensuring their optimal performance, security, and efficiency. It includes tasks such as installing software, managing user accounts, configuring system settings, and monitoring system health.

How Linux Administration works?

Linux administration works through command-line tools and system utilities that allow administrators to configure and control different aspects of the system, from network settings to user permissions. Administrators also use scripting and automation to streamline routine tasks and ensure system integrity.

Linux Administration: Why and When

Linux administration is crucial for managing servers, networks, and cloud platforms, especially in environments requiring high security, scalability, and performance. It is needed whenever there is a need to handle large-scale or mission-critical applications with flexible and open-source technologies.

Example: A company might need Linux administration when deploying a new web server to ensure robust and secure performance.

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

If you want to install it on a different Linux distro, ask in the comments and I will write an Ansible playbook or bash script.

Linux Administration: Main components & packages

- Linux administration is bare linux feature

Linux Administration: User Structure

1) User structure files

In Linux, the ‘useradd‘ command is a low-level utility used for adding or creating user accounts in Linux and other Unix-like operating systems. The ‘adduser‘ command is very similar to the ‘useradd‘ command, as it is just a symbolic link to it.

When we run the ‘useradd‘ command in the Linux terminal, it performs the following major tasks:

- It edits **/etc/passwd**, **/etc/shadow**, **/etc/group**, and **/etc/gshadow** files for the newly created user accounts.
- Creates and populates a home directory for the new user.
- Sets permissions and ownerships to the home directory.

/etc/passwd – User account information.
/etc/shadow – Secure account information.
/etc/group – Group account information.
/etc/gshadow – Secure group account information.
/etc/login.defs – Shadow password suite configuration..

2) User files records

/etc/passwd – User account information.

```
username:x:UID:GID:comment:home_directory:shell
```

- **username**: The login name of the user.
- **x**: Placeholder for the password. In modern systems, passwords are not stored here for security reasons. The actual password is in /etc/shadow.
- **UID (User ID)**: A unique identifier for the user.
- **GID (Group ID)**: The primary group the user belongs to, linked to /etc/group.
- **comment:** A field for additional information like the user's full name (can also be left blank).
- **home_directory:** The user's home directory (e.g., /home/username).
- **shell:** The default shell assigned to the user (e.g., /bin/bash).

/etc/shadow – Secure account information

```
username:encrypted_password:last_password_change:min_age:max_age:warning:
inactive:expiration:
```

username: The login name.

encrypted_password: The encrypted version of the user's password. If the password is locked, this field may contain a ! or * symbol.

last_password_change: The number of days since the last password change.

min_age: The minimum number of days before the password can be changed.

max_age: The maximum number of days the password remains valid before it must be changed.

warning: The number of days before the password expires that a warning will be given.

inactive: Days after a password expires when the account will be disabled.

expiration: The date the account expires (if applicable).

/etc/group – Group account information

```
username:x:UID:GID:comment:home_directory:shell
```

- **group_name:** The name of the group.
- **password_placeholder:** A placeholder for the group password, usually x (not commonly used).
- **GID (Group ID):** The numeric identifier for the group.
- **user_list:** A comma-separated list of users who are members of the group, aside from the group's primary user.

/etc/gshadow – Secure group account information

```
group_name:encrypted_password:group_admins:group_members
```

- **group_name:** The name of the group.
- **encrypted_password:** The encrypted group password (used rarely).
- **group_admins:** A comma-separated list of group administrators who can manage the group.
- **group_members:** A comma-separated list of users who are members of the group.

3) create new user

```
adduser [new_account]
```

When a new user account is added to the system, the following operations are performed.

1. His/her home directory is created (**/home/username** by default).
2. The following hidden files are copied into the user’s home directory, and will be used to provide environment variables for his/her user session.

```
.bash_logout
.bash_profile
.bashrc
```

.bash_logout

- Purpose: This file is executed automatically when a user logs out from a Bash session (a login shell)

.bash_profile

- Purpose: This file is executed for login shells—when a user logs in to a session (via console, SSH, etc.).

.bashrc

- Purpose: This file is executed for non-login shells—usually interactive shells, such as when you open a terminal emulator or run a script.

4) Usefull commands

create user with Specified Home, Shell, Skeleton, and UID

```
useradd -m -d /var/www/user -k /etc/custom.skell -s /bin/bash -c "Some
text" -u 1027 user
```

- A new user named navin will be created.
- The user's home directory will be /var/www/user.
- The files from /etc/custom.skell will be copied to this home directory when it is created.
- The user's default shell will be /bin/bash.
- The comment "Some" will be stored in the GECOS field for the user.
- The UID of the user will be set to 1027.

change shell for user

```
usermod -s /bin/sh user
```

Move home directory to new location

```
usermod -d /var/user2/ -m user2
```

change user primary group

```
usermod -g group user
```

Linux Administration: permission structure

1) chmod

The chmod command is used to change the permissions of files and directories.

Permissions define who can read, write, or execute a file or directory.

Permissions Overview:

In Linux, there are three types of permissions:

- r: Read (4)
- w: Write (2)
- x: Execute (1)

Permissions are applied for three categories of users:

- User (u): The owner of the file.
- Group (g): Members of the file's group.
- Others (o): All other users.

Common chmod Syntax:

```
chmod [permissions] [file/directory]
```

You can set permissions using either:

- Symbolic mode (e.g., u+r, g-w), or
- Numeric mode (e.g., 755, 644).

Example and Best pracice

1. Principle of Least Privilege

What It Means: Give users or processes the minimum permissions they need to perform their tasks.

Example: For sensitive files like configuration files (config.cfg), use restrictive permissions like (only owner can read and write):

```
chmod 600 config.cfg
```

Use 755 for Publicly Accessible Directories

755: Owner can read, write, and execute; group and others can only read and execute.

```
chmod 755 /var/www/html
```

Be Cautious with the 777 Permission

What It Means: Avoid using 777 permissions (read, write, execute for everyone) unless absolutely necessary.

Set 700 for Private Directories and Files, 600 for files ensures only the owner can read and write or even 400 to ensure it cannot be modified (only read)

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/id_rsa
chmod 400 ~/.ssh/id_rsa
```

Avoid Recursive chmod on Large Directories without Testing

- What It Means: Be careful when using the -R (recursive) flag with chmod on directories.
- Why: A recursive chmod can change permissions on all files and subdirectories, which can lead to unintended results.
- Best Practice: Test changes on individual files or smaller subsets first, and ensure that the permissions being applied are suitable for both directories and files.

Use Symbolic Mode for Clarity in Small Changes

What It Means: Use symbolic permissions (u, g, o) to adjust individual permission bits clearly, rather than resetting all permissions.

Adding execute permission only for the owner:
bash

```
chmod u+x script.sh
```

u+x: Adds execute permission for the owner without affecting other permissions.

```
chmod a+r file.txt       # Add read permission for everyone
chmod a-wx file.txt      # Remove write and execute permission for everyone
chmod a=rx file.txt      # Set read and execute permissions for everyone
```

2) Setgid, stickybit, chattr

What is Setgid?
The setgid (Set Group ID) bit on a directory allows all files created within that directory to inherit the group ownership of the directory, rather than the primary group of the user who created the file.

```
chmod g+s [filename]
```

or prepend 2

```
chmod 2755 [directory]
```

What is Sticky Bit?

The sticky bit is a permission bit used on directories that prevents users from deleting files owned by other users in that directory. Only the owner of the file, the owner of the directory, or the root user can delete or rename the file.

```
chmod o+t [directory]
```

or prepend 1

```
sudo chmod 1777 /shared
```

- All users can create files in the **/shared** directory.
- Only the owner of a file can delete or rename their own files, preventing other users from deleting important files in the shared directory.

chattr is a command used to modify file attributes at a low level. These attributes can provide extra control over how files are accessed and modified. For example, you can make a file immutable, meaning it cannot be deleted, modified, or renamed, even by the root user (until the immutable flag is removed)

The immutable attribute prevents any changes to **/etc/passwd**, providing an extra layer of protection against accidental or malicious modifications to crucial system files.

```
sudo chattr +i /etc/passwd
```

Append data (allowed): Appending data (e.g., logging more information) works as usual, but no one can modify or delete existing data.

```
sudo chattr +a /etc/passwd
```

3) chown

The chown command in Linux is used to change the ownership of files and directories. Proper use of chown is important for maintaining security, ensuring that users have appropriate permissions, and managing system files. Here are some best practices for using chown effectively:

Understand Ownership and Permissions

- Owner: The user who owns the file or directory.
- Group: The group associated with the file or directory.
- Familiarize yourself with the difference between ownership and permission settings.

```
chown [OPTION]... [OWNER][:[GROUP]] FILE...
```

Avoid Changing System Files

- Do not change ownership of system files unless you are certain of the consequences.
- Be cautious when changing ownership of files in directories like /etc, /usr, and /var.

Use Groups Effectively

- When changing ownership, consider whether you need to change the group as well.
- Use groups to manage access for multiple users effectively.

Change Ownership Recursively with Caution

- The -R (recursive) option allows you to change the ownership of all files and directories within a directory.

Linux Administration: visudo

To grant access to sudo, the system administrator must edit the **/etc/sudoers** file. It is recommended that this file is edited using the visudo command instead of opening it directly with a text editor.

```
visudo
```

These are the most relevant lines.

```
Defaults    secure_path="/usr/sbin:/usr/bin:/sbin"
root        ALL=(ALL) ALL
user     ALL=/bin/yum update
user2    ALL=NOPASSWD:/bin/updatedb
%admin      ALL=(ALL) ALL
```

This line lets you specify the directories that will be used for sudo(PATH for binaries), and is used to prevent using user-specific directories, which can harm the system.

```
Defaults    secure_path="/usr/sbin:/usr/bin:/sbin:/usr/local/bin"
```

The next lines are used to specify permissions.

1. The first ALL keyword indicates that this rule applies to all hosts.
2. The second ALL indicates that the user in the first column can run commands with the privileges of any user.
3. The third ALL means any command can be run.

```
root        ALL=(ALL) ALL\
```

If no user is specified after the = sign, sudo assumes the root user. In this case, user tecmint will be able to run yum update as root.

```
user     ALL=/bin/yum update
```

The NOPASSWD directive allows user user2 to run /bin/updatedb without needing to enter his password.

```
user2    ALL=NOPASSWD:/bin/updatedb
```

The % sign indicates that this line applies to a group called “admin”. The meaning of the rest of the line is identical to that of an regular user. This means that members of the group “admin” can run all commands as any user on all hosts.

```
%admin      ALL=(ALL) ALL
```

Common troubleshooting

1) Accidental Ownership or Permission Changes on Critical Directories

Problem: Using chown -R or chmod -R without caution can accidentally change ownership or permissions on critical directories like /etc, /usr, or /var. This can lead to system instability, security issues, or even render the system unusable.

2) Breaking System Services by Changing Permissions

Problem: Incorrect recursive changes to directories like /var/www (web files), /var/lib/mysql (database files), or /etc (configuration files) can prevent system services from working because the services may no longer have the correct access permissions.

3) Slow Speeds or High Latency

Description: VPN connection is slow or has high ping times.
Causes: Network congestion, poor server location, or using a heavy encryption method that may slow down the connection.

4) **Just learn more about Linux administration best practices**

5) **If everything is a complete mess (in case of wrong user configuration)**

Remove the user and revert the configuration to its previous state.

How to remove user

1) Remove

Afer running script, choose Remove VPN option

```
$ sudo userdel -r username
```

remove all related files

```
sudo find / -user username -exec rm -rf {} \;
```

Learn more about Linux administration

Check RedHat, they have great docs

RedHat

Red Hat is named a Leader for the 2nd consecutive year in the 2024 Gartner

https://www.redhat.com/en

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!