

Lab Project - 8

Objective: Linux environment management lab

DURATION: 3 - 4.5 Hourse

PRE-REQUISITES:

Oracle VirtualBox or VMWare, Ubuntu installed.

Lab 1: Configuring and Managing User Environments

Objective: Learn to manage and configure user-specific environment settings in Linux.

Task:

1. Set Environment Variables:

Set environment variables like PATH, EDITOR, and JAVA_HOME in /etc/profile, /etc/bash.bashrc, and user-specific files like ~/.bashrc.

```
OP-5K616C3:~$ sudo nano /etc/profile
OP-5K616C3:~$
```

```
#Add the environment variable at the end of the file
export PATH="$HOME/bin:$PATH" export JAVA_HOME="/home/vinu/vinu/to/java" export
EDITOR="nano"
```

```
@DESKTOP-5K616C3:~$ nano ~/.bashrc
@DESKTOP-5K616C3:~$
```

```
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# If set, the pattern "***" used in a pathname expansion context will
[ Read 117 lines ]
```

#Verify the environment variables using echo \$VARIABLE_NAME.

```
vinu@DESKTOP-5K616C3:~$ echo $HOME
/home/vinu
vinu@DESKTOP-5K616C3:~$
```

2. Configure Bash Prompt:

#Modify the PS1 variable to customize the command prompt.

Example Customizations:

1. Basic Customization

bash

```
PS1="\u@\h:\w$ "
```

- `\u` → Username
- `\h` → Hostname
- `\w` → Current working directory
- `$` → Prompt symbol (`#` for root)

2. Colorized Prompt

bash

```
PS1="\[\e[32m\]\u@\h:\[\e[34m\]\w\[\e[0m\]$ "
```

- `\e[32m` → Green (username@hostname) ↓

- `\e[32m` → Green (username@hostname)
- `\e[34m` → Blue (working directory)
- `\e[0m` → Reset color

3. Adding Date & Time

bash

```
PS1="[\d \t] \u@\h:\w$ "
```

- `\d` → Date (e.g., "Fri Feb 23")
- `\t` → Time (HH:MM:SS)

4. Emoji & Custom Symbols

bash

```
PS1="🌟 \u@\h:\w → "
```

e/vinu

```
@DESKTOP-5K616C3:~$ PS1="\u@\h:\w$ "
@DESKTOP-5K616C3:~$
```

#Set a colored prompt and add user-specific information like username, hostname, and current directory.

```
TOP-5K616C3:~$ nano ~/.bashrc
TOP-5K616C3:~$
```

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w$ '
fi
unset color_prompt force_color_prompt
```

3. Create and Manage Aliases:

#Set up aliases for common commands (e.g., `alias ll='ls -l'`).

```
vinu@DESKTOP-5K616C3:~$ alias ll='ls -l  
> _
```

Store aliases in ~/.bashrc and ensure they are loaded at login.

```
OP-5K616C3:~$ ~/.bashrc  
ome/vinu/.bashrc: Permission denied  
OP-5K616C3:~$
```

4. Configure Shell Options:

Enable options such as noclobber (prevent overwriting files) and autocd (auto-change directory) in the shell.

```
SKTOP-5K616C3:~$ set -o noclobber  
SKTOP-5K616C3:~$ _
```

Lab 2: Managing System-Wide Environment Settings

Objective: Learn to configure system-wide environment settings for all users.

Task:

1. Configure Global Environment Variables:

Set global environment variables in /etc/environment, /etc/profile, and /etc/bash.bashrc.

ANS

sudo nano /etc/environment

```
GNU nano 6.2 /etc/environment  
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
```

sudo nano /etc/profile

```
export MY_VARIABLE=value export ANOTHER_VARIABLE=value2
```

source /etc/environment source /etc/profile source /etc/bash.bashrc

```
DESKTOP-5K616C3:~$ source /etc/environment source /etc/profile source /etc/bash.bashrc
DESKTOP-5K616C3:~$
```

2.Configure Shell Initialization Files:

Modify /etc/profile and /etc/bash.bashrc to configure system-wide settings such as umask, PATH, and the default shell.

/etc/environment :

ini

```
MY_VAR="my_value"
```

/etc/profile :

bash

```
export MY_VAR="my_value"
```

/etc/bash.bashrc :

bash

```
export MY_VAR="my_value"
```

3. Control User Environment with PAM:

Modify /etc/pam.d/common-session to ensure user-specific environment settings are correctly applied for each login session.

sudo nano /etc/pam.d/common-session

```
# here are the per-package modules (the "Primary" block)
session [default=1]                                pam_permit.so
# here's the fallback if no module succeeds
session requisite                                   pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
session required                                   pam_permit.so
# The pam_umask module will set the umask according to the system default in
# /etc/login.defs and user settings, solving the problem of different
# umask settings with different shells, display managers, remote sessions etc.
# See "man pam_umask".
session optional                                   pam_umask.so
```

Verify the ~/.pam_environment file exists: Each user can have a .pam_environment home directory where user-specific environment variables can be set.

For example, a .pam_environment file might look like this:

```
bash

VAR_NAME VALUE PATH DEFAULT
```

You can create this file if it doesn't exist:

```
bash

touch ~/.pam_environment
```

Reload or restart session: After making these changes, restart the login session for effect, or log out and log back in.

4. Set System-Wide Aliases:

Create aliases in /etc/bash.bashrc for commonly used system commands (e.g., alias rm='rm -i' to prompt before deleting files).

sudo nano /etc/bash.bashrc

```
# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# set variable identifying the chroot you work in (used in the prompt below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, overwrite the one in /etc/profile)
# but only if not SUDOing and have SUDO_PS1 set; then assume smart user.
if ! [ -n "${SUDO_USER}" -a -n "${SUDO_PS1}" ]; then
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi

# Commented out, don't overwrite xterm -T "title" -n "icontitle" by default.
# If this is an xterm set the title to user@host:dir
#case "$TERM" in
```

```
alias rm='rm -i' # Prompt before removing files alias ll='ls -l' # Use 'll' for long
directory listing alias la='ls -A' # List all files, including hidden files alias
l='ls -CF' # Simple directory listing alias grep='grep --color=auto' # Colorize grep
output
```

```
source /etc/bash.bashrc
```

```
u@DESKTOP-5K616C3:~$ source /etc/bash.bashrc
u@DESKTOP-5K616C3:~$
```

```
5.Test User Sessions:

    #Test login with multiple users to ensure that system-wide
    configurations are applied.
```

1. **Switch Users:** Use `su` or `sudo` to switch to a different user and verify that `tl` variables are set correctly.

- ```
bash

su - username
```

```
su - username
```

- ```
bash
env
```

env

Objective: Learn how to manage the system's time zone and locale settings.

Task:

1. Configure Time Zone:

Use the `timedatectl` command to set the system time zone (e.g., `timedatectl set-timezone America/New_York`).

- ```
Use the timedatectl command to set the system time zone
(e.g., timedatectl set-timezone America/New York).
```

```
SKTOP-5K616C3:~$ printenv
/bin/bash
I_APPS_ENABLED=1
PRO_NAME=Ubuntu-22.04
SKTOP-5K616C3
me/vinu
=vinu
ome/vinu
JTF-8
EROP=/run/WSL/2274_interop
RS=rs=0;di=01;34:ln=01;36:mh=00;pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=4
```



```
sudo timedatectl set-timezone America/New_York
```

```
K616C3:~$ sudo timedatectl set-timezone America/New_York
K616C3:~$
```

```
timedatectl
```

```
K616C3:~$ timedatectl
Local time: Fri 2025-02-28 07:16:44 EST
Universal time: Fri 2025-02-28 12:16:44 UTC
RTC time: Fri 2025-02-28 07:16:48
Time zone: America/New_York (EST, -0500)
Synchronized: no
NTP service: active
In local TZ: no
K616C3:~$
```

## 2. Synchronize Time with NTP:

#Configure NTP (Network Time Protocol) for time synchronization using systemctl enable ntp and verify synchronization with timedatectl.

```
K616C3:~$ timedatectl
Local time: Fri 2025-02-28 07:16:44 EST
Universal time: Fri 2025-02-28 12:16:44 UTC
RTC time: Fri 2025-02-28 07:16:48
Time zone: America/New_York (EST, -0500)
Synchronized: no
NTP service: active
In local TZ: no
K616C3:~$
```

### 1. Enable and start the NTP service:

Use the `systemctl` command to enable the NTP service to start on boot and then

```
bash

sudo systemctl enable ntp sudo systemctl start ntp
```

### 2. Verify the synchronization:

You can check if the time synchronization is working by using the `timedatectl`

```
bash

timedatectl status
```

In the output, check the "NTP synchronized" line. It should say "yes" if the system is synchronized with an NTP server.

### 3. (Optional) Verify the NTP synchronization with `ntpq` :

You can use the `ntpq` command to query the NTP server and get detailed synchronization information:

```
bash

ntpq -p
```

This will show the NTP server peers and the synchronization status.

### 3. Set Locale:

```
#Configure system locale using locale and localectl (e.g.,
localectl set-locale LANG=en_US.UTF-8).
```

```
DESKTOP-5K616C3:~$ sudo localectl set-locale LANG=en_US.UTF-8
DESKTOP-5K616C3:~$ _
```

```
#Test the locale setting with locale and configure the
keyboard layout if needed.
```

locale

```
vinu@DESKTOP-5K616C3:~$ locale
LANG=C.UTF-8
LANGUAGE=
LC_CTYPE="C.UTF-8"
LC_NUMERIC="C.UTF-8"
LC_TIME="C.UTF-8"
LC_COLLATE="C.UTF-8"
LC_MONETARY="C.UTF-8"
LC_MESSAGES="C.UTF-8"
LC_PAPER="C.UTF-8"
LC_NAME="C.UTF-8"
LC_ADDRESS="C.UTF-8"
LC_TELEPHONE="C.UTF-8"
LC_MEASUREMENT="C.UTF-8"
LC_IDENTIFICATION="C.UTF-8"
LC_ALL=
```

4.Change Date and Time Manually:

```
Use date to set the current date and time manually (useful
for debugging)
```

## On Linux/macOS (Terminal):

1. Set the date using `date` :

```
bash
```

```
sudo date MMDDhhmm[[CC]YY][.ss]
```

- MM = month (01–12)
- DD = day of the month (01–31)
- hh = hour (00–23)
- mm = minute (00–59)
- CC = century (optional, 00–99)
- YY = year (two digits)
- .ss = seconds (optional)

Example:

```
bash
```

```
sudo date 022312302025
```

### On Windows (Command Prompt):

1. Set the date with `date` :

```
cmd

date MM-DD-YY
```

2. Set the time with `time` :

```
cmd

time HH:MM:SS
```

Example:

```
cmd

date 02-23-2025 time 15:30:00
```

### On Windows (Command Prompt):

1. Set the date with `date` :

```
cmd

date MM-DD-YY
```

2. Set the time with `time` :

```
cmd

time HH:MM:SS
```

Example:

```
cmd

date 02-23-2025 time 15:30:00
```

## 5. Verify Changes:

# Ensure that the time zone and locale settings are applied by checking /etc/localtime and environment variables like LANG.

```
vinu@DESKTOP-5K616C3:~$ ls -l /etc/localtime
-rwxrwx 1 root root 36 Feb 28 02:15 /etc/localtime -> /usr/share/zoneinfo/America/New_York
vinu@DESKTOP-5K616C3:~$
```

## Lab 4: Configuring System PATH and Executable Search Order

**Objective: Understand how to manage the system's executable search path and control command execution order.**

### Task:

1. View the Current PATH:

# Use echo \$PATH to view the current directories listed in the system PATH.

```
vinu@DESKTOP-5K616C3:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

2. Modify the PATH:

# Add directories to the PATH in /etc/profile, /etc/bash.bashrc, and ~/.bashrc to include custom executable directories.

```
ord for vinu:
-5K616C3:~$ sudo nano /etc/profile
-5K616C3:~$
```

#Test the new directory by placing an executable in a new directory and running it directly.

```
_export PATH=$PATH:/home/user/custom_bin
```

```
^G Help ^O Write Out ^W Where Is ^K Cut
^Y Exit ^R Read File ^\ Replace ^U Paste
```

### 3. Configure Local User PATH:

# Modify ~/.bash\_profile or ~/.bashrc to append directories to the user-specific PATH.

```
vinu@DESKTOP-5K616C3:~$ nano ~/.bashrc
vinu@DESKTOP-5K616C3:~$ _
don't put duplicate lines or lines starting with space in the history.
See bash(1) for more options
HISTCONTROL=ignoreboth

append to the history file, don't overwrite it
shopt -s histappend

for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

check the window size after each command and, if necessary,
update the values of LINES and COLUMNS.
shopt -s checkwinsize
```

### 4. Ensure Proper Order of PATH:

# Ensure that custom directories are searched before system directories by placing them at the beginning of the PATH.

```
DESKTOP-5K616C3:~$ nano ~/.bashrc
DESKTOP-5K616C3:~$ export PATH=/path/to/custom/directory:$PATH
DESKTOP-5K616C3:~$
```

## For permanent changes:

To make this change persistent, add the `export` command to your shell's configuration file:

- For **bash** (common default shell), add it to `~/.bashrc` or `~/.bash_profile`.
- For **zsh**, add it to `~/.zshrc`.
- For **fish**, add it to `~/.config/fish/config.fish`.

For example, in `~/.bashrc` (or `~/.bash_profile`), you would add:

```
bash
```



```
export PATH=/path/to/custom/directory:$PATH
```

Then, apply the changes by sourcing the file:

```
bash
```



```
source ~/.bashrc # Or source ~/.bash_profile
```

This ensures that the custom directory is always prioritized over system directories when looking for executables.

## 5. Test Command Execution Order:

#Test the execution order of commands by creating two executables with the same name in different directories.

```
[2]~ Stopped vim path_basic.sh
vinu@DESKTOP-5K616C3:~$ export PATH=~/.dir1:$PATH
vinu@DESKTOP-5K616C3:~$
```



## Lab 5: Configuring and Managing User Groups and Permissions

**Objective:** Learn how to manage user groups and file permissions to secure the system.

### Task:

1. Create User Groups:

# Create user groups with groupadd (e.g., groupadd developers).

sudo groupadd developers

```
vinu@DESKTOP-5K616C3:~$ sudo groupadd developers
do] password for vinu:
vinu@DESKTOP-5K616C3:~$
```

getent group developers

```
vinu@DESKTOP-5K616C3:~$ getent group developers
developers:x:1009:
```

cat /etc/group | grep developers

```
vinu@DESKTOP-5K616C3:~$ cat /etc/group | grep developers
developers:x:1009:
```

sudo groupadd groupname1 sudo groupadd groupname2

```
vinu@DESKTOP-5K616C3:~$ sudo groupadd groupname1 sudo groupadd groupname2
Usage: groupadd [options] GROUP

Options:
 -f, --force exit successfully if the group already exists
 and cancel -g if the GID is already used
 -g, --gid GID use GID for the new group
 -h, --help display this help message and exit
 -K, --key KEY=VALUE override /etc/login.defs defaults
 -o, --non-unique allow to create groups with duplicate
 (non-unique) GID
 -p, --password PASSWORD use this encrypted password for the new group
 -r, --system create a system account
 -R, --root CHROOT_DIR directory to chroot into
 -P, --prefix PREFIX_DIR directory prefix
 --extrausers Use the extra users database
```

## 2. Add Users to Groups:

# Use `usermod -aG groupname username` to add users to a group.

```
u. Cannot lock /etc/passwd, try again later.
ESKTOP-5K616C3:~$ sudo usermod -aG vj vinu
ESKTOP-5K616C3:~$
```

## 3. Set File Permissions:

# Use `chmod`, `chown`, and `chgrp` to configure file ownership and permissions for directories and files.

`chmod`

`chmod 700 mydir/`

```
vinu@DESKTOP-5K616C3:~$ chmod 700 vinu/file.txt
vinu@DESKTOP-5K616C3:~$
```

`Chown`

### 1. Change Ownership with `chown`

- The `chown` command is used to change the ownership of a file or directory.
- Syntax: `chown [owner][:group] file/directory`
  - To change the owner of a file: `chown user filename`
  - To change both the owner and the group: `chown user:group filename`

Example:

```
bash
```

```
chown john:staff myfile.txt
```

This will set `john` as the owner and `staff` as the group for `myfile.txt`.

```
DESKTOP-5K616C3:~$ sudo chown vinu:vj file.txt
DESKTOP-5K616C3:~$
```

## 2. Change Group Ownership with `chgrp`

- The `chgrp` command is used to change the group ownership of a file or directory.
- Syntax: `chgrp group file/directory`

Example:

```
bash
```



```
chgrp admins myfile.txt
```

This changes the group ownership of `myfile.txt` to `admins`.

```
DESKTOP-5K616C3:~$ chgrp vj file.txt
DESKTOP-5K616C3:~$
```

#Set directory and file permissions for different users (e.g., read, write, execute) and test access.

```
DESKTOP-5K616C3:~$ chmod 644 file.txt
DESKTOP-5K616C3:~$
```

### 3. Change Permissions with `chmod`

- The `chmod` command is used to change the permissions of a file or directory.
- Syntax: `chmod permissions file/directory`
  - **Read** = `r` (4)
  - **Write** = `w` (2)
  - **Execute** = `x` (1)
  - You can combine permissions by adding the values:
    - `rwX` =  $4 + 2 + 1 = 7$
    - `rw-` =  $4 + 2 = 6$
    - `r--` =  $4$
- Permissions can be set for the **user (owner)**, **group**, and **others**.

#### Examples:

- To set `rw-` for owner, `r--` for group, and `r--` for others:

```
bash
chmod 644 myfile.txt
```

- To give full permissions (read, write, execute) to the owner and read and execute group and others:

```
bash
chmod 755 myfile.txt
```

- To make a directory executable (so it can be entered):

```
bash
chmod 700 mydir/
```

#### 4. Test Permissions:

#Log in as a user from different groups and test the permissions and file access to verify proper configuration.

```
u@DESKTOP-5K616C3:~$ ls -l file.txt
-r--r-- 1 vinu vj 0 Feb 28 08:38 file.txt
u@DESKTOP-5K616C3:~$
```

#### 5. Set Up Sudo Access:

#Add a user to the sudoers file to allow elevated permissions using visudo.

### Step 1: Open the sudoers file safely using visudo

```
bash

sudo visudo
```

This prevents syntax errors that could lock you out of your system.

### Step 2: Add the user to the sudoers file

Scroll down to find a section that looks like this:

```
plaintext

root ALL=(ALL:ALL) ALL
```

Add a new line below it with your username. Replace `<username>` with the actual

```
User privilege specification
root ALL=(ALL:ALL) ALL
Members of the admin group may gain root privileges
```

## Lab 6: Automating Environment Setup with Scripts

**Objective:** Automate environment configuration and settings using shell scripts.

### Task:

1. Create a User Environment Setup Script:

# Write a script that sets up environment variables, custom aliases, and modifies the prompt.

```
vinu@DESKTOP-5K616C3:~$ cat setup.sh
Set environment variables
export EDITOR=nano
export HISTSIZE=5000
export HISTFILESIZE=10000
export PATH="$HOME/bin:$PATH"
export PS1='\[\e[32m\]\u@\h:\w\$ \[\e[m\]'

Custom Aliases
alias ll='ls -lah'
alias gs='git status'
alias gp='git pull'
alias venv='source venv/bin/activate'
alias pyserve='python3 -m http.server'

Custom Functions
mkcd() {
 mkdir -p "$1" && cd "$1"
}

Apply changes
if [["$SHELL" == *"zsh"*]]; then
 source ~/.zshrc
elif [["$SHELL" == *"bash"*]]; then
 source ~/.bashrc
fi

echo "Environment setup complete!"
vinu@DESKTOP-5K616C3:~$
```

#The script should add settings to ~/.bashrc or ~/.bash\_profile and apply them to the user's session.

```
u@DESKTOP-5K616C3:~$ sudo visudo
```

```
directly modifying this file.
#
See the man page for details on how to write a sudoers file.
#
Defaults env_reset
Defaults mail_badpass
Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
Defaults use_pty

This preserves proxy settings from user environments of root
equivalent users (group sudo)
#Defaults:%sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy no_proxy"

This allows running arbitrary commands, but so does ALL, and it means
different sudoers have their choice of editor respected.
#Defaults:%sudo env_keep += "EDITOR"

Completely harmless preservation of a user preference.
#Defaults:%sudo env_keep += "GREP_COLOR"
```

```
vinu@DESKTOP-5K616C3:~$ bash startup.sh
startup.sh: line 1: ettings: command not found
Settings added to /home/vinu/.bashrc
Settings applied to the current session.
```

```
plaintext
```

[Copy](#)

```
<username> ALL=(ALL:ALL) ALL
```

This grants full sudo access.

### Step 3: Save and Exit

- Press `Ctrl + X` to exit.
- Press `Y` to save changes.
- Press `Enter` to confirm.

### Alternative: Add User to the sudo Group

On many Linux distributions, adding a user to the `sudo` group automatically grants them sudo

```
bash
```

[Copy](#)

```
sudo usermod -aG sudo <username>
```

## 2. Automate Software Installation:

```
#Write a script to install commonly used packages and software
(e.g., vim, git, curl).
```

```
#Use package managers like apt, yum, or dnf to automate
installation.
```

```
#!/bin/bash
```

```
Define a list of packages to install
```

```
packages=(vim git curl)
```

```
Detect the package manager
```



```
if command -v apt >/dev/null; then
 pkg_manager="apt"
elif command -v yum >/dev/null; then
 pkg_manager="yum"
elif command -v dnf >/dev/null; then
 pkg_manager="dnf"
else
 echo "Unsupported package manager!"
 exit 1
fi

Update package lists
if [["$pkg_manager" == "apt"]]; then
 sudo apt update -y
fi

Install the packages
for package in "${packages[@]"; do
 echo "Installing $package..."
 sudo $pkg_manager install -y $package
done

echo "Installation completed."
```

```

vinu@DESKTOP-5K616C3:~$ bash programm.sh
programm.sh: line 1: list: command not found
[sudo] password for vinu:
Ign:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Ign:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Ign:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Ign:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Ign:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease

```

- Defines a list of packages to install.
- Detects the appropriate package manager (apt, yum, or dnf).
- Updates package lists (for apt).
- Installs each package in the list.

### 3.Configure Environment Based on User Input:

#Modify the script to configure different environments based on user input, such as custom editor settings or shell options.

```

PS C:\Users\mlr> ./script.sh
./script.sh : The term './script.sh' is not recognized as the name of a cmdlet, function, script file, or operable
program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ ./script.sh
+ ~~~~~
+ CategoryInfo : ObjectNotFound: (./script.sh:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\mlr> python script.py
C:\Users\mlr\AppData\Local\Programs\Python\Python312\python.exe: can't open file 'C:\\Users\\mlr\\script.py': [Errno 2]
No such file or directory
PS C:\Users\mlr>

```

## Lab 7: Configuring System-Wide Security Settings

**Objective: Configure system-wide security settings to harden the environment and secure user access.**

### Task:

#### 1. Set Up Password Policies:

#Edit /etc/login.defs to enforce password length, expiration, and complexity rules.

```
[sudo] password for vinu:
vinu@DESKTOP-5K616C3:~$ PASS_MAX_DAYS 90
+ PASS_MAX_DAYS 90
+ '[' -x /usr/lib/command-not-found ']'
+ /usr/lib/command-not-found -- PASS_MAX_DAYS
```

#Use chage to configure password aging for users.

```
+ return 127
vinu@DESKTOP-5K616C3:~$ sudo chage -M 90 vinu
+ sudo chage -M 90 vinu
vinu@DESKTOP-5K616C3:~$
```

#### 2. Limit User Logins:

#Configure the /etc/security/limits.conf file to set resource limits for users and groups (e.g., maximum number of simultaneous logins).

```
sudo nano /etc/security/limits.conf
```

```
/etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain> <type> <item> <value>
#
#Where:
#<domain> can be:
- a user name
- a group name, with @group syntax
- the wildcard *, for default entry
- the wildcard %, can be also used with %group syntax,
for maxlogin limit
- NOTE: group and wildcard limits are not applied to root.
To apply a limit to the root user, <domain> must be
the literal username root.
#
#<type> can have the two values:
```

## Steps to Set Resource Limits in `/etc/security/limits.conf`

1. Open the file in a text editor with root privileges:

```
bash
```

[Copy](#) [Edit](#)

```
sudo nano /etc/security/limits.conf
```

2. Add or modify the appropriate limits for users or groups. The syntax follows this format:

```
php-template
```

[Copy](#) [Edit](#)

```
<domain> <type> <item> <value>
```

- **<domain>**: Can be a username, group (prefixed with `@`), or wildcard (`*` for all users).
- **<type>**: Either **soft** (enforceable but can be increased up to **hard** limit) or **hard** (absolute maximum).
- **<item>**: The resource to limit.



- `<value>`: The limit value.

3. Example entries:

```
pgsql

Limit user "john" to 3 simultaneous logins
john hard maxlogins 3

Limit all users to 10 simultaneous logins
* hard maxlogins 10

Limit group "developers" to 5 simultaneous logins
@developers hard maxlogins 5

Limit user "jane" to 1024 open files
jane soft nofile 1024
jane hard nofile 2048
```

4. Save and exit (for nano, press `CTRL+X`, then `Y`, and `Enter`).

5. Ensure the PAM module is enabled to enforce limits: Open `/etc/pam.d/common-session` or `/etc/pam.d/sshd` (for SSH) and check that the following line exists:

```
swift

session required pam_limits.so
```

 Copy

6. **Apply the changes:** Log out and log back in, or restart the system.

3. Enable Firewall:

# Configure ufw or iptables to restrict access to the system based on IP addresses, ports, or protocols.

## Using UFW (Uncomplicated Firewall)

UFW is a user-friendly interface for managing firewall rules.

### 1. Install UFW (if not installed)

```
bash

sudo apt update && sudo apt install ufw -y
```

### 2. Enable UFW

```
bash

sudo ufw enable
```

### 3. Allow Essential Services (e.g., SSH, HTTP, HTTPS)

```
bash

sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
```



---

#### 4. Restrict Access Based on IP Address

Allow SSH access only from a specific IP (e.g., 192.168.1.100):

```
bash

sudo ufw allow from 192.168.1.100 to any port 22
```

Deny access from a specific IP:

```
bash

sudo ufw deny from 203.0.113.50
```

#### 5. Block All Incoming Traffic Except Allowed Rules

```
bash

sudo ufw default deny incoming
sudo ufw default allow outgoing
```

```
vinu@DESKTOP-5K616C3:~$ sudo apt update && sudo apt install ufw
+ sudo apt update
Reading package lists... Done
E: Could not get lock /var/lib/apt/lists/lock. It is held by process 2932 (apt)
N: Be aware that removing the lock file is not a solution and may break your system.
E: Unable to lock directory /var/lib/apt/lists/
```

```
vinu@DESKTOP-5K616C3:~$ sudo ufw enable
+ sudo ufw enable
Firewall is active and enabled on system startup
vinu@DESKTOP-5K616C3:~$
```

#### 4. Configure SSH Settings:

#Edit /etc/ssh/sshd\_config to disable root login, set strong encryption, and limit SSH access to specific users or groups.

#### 3. Allow Essential Services (e.g., SSH, HTTP, HTTPS)

```
bash

sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
```

```
vinu@DESKTOP-5K616C3:~$ sudo ufw allow ssh
+ sudo ufw allow ssh
Skipping adding existing rule
```

```
vinu@DESKTOP-5K616C3:~$ sudo ufw allow http
+ sudo ufw allow http
```

```
vinu@DESKTOP-5K616C3:~$ sudo ufw allow https
+ sudo ufw allow https
```

#### 5. Audit and Monitor User Access:

#Install and configure auditd for auditing user activities and access.

```
sudo apt install auditd audispd-plugins -y
```

```
vinu@DESKTOP-5K616C3:~$ sudo apt install auditd audispd-plugins -y
+ sudo apt install auditd audispd-plugins -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
 libauparse0
The following NEW packages will be installed:
 audispd-plugins auditd libauparse0
0 upgraded, 3 newly installed, 0 to remove and 13 not upgraded.
```

```
vinu@DESKTOP-5K616C3:~$ sudo systemctl enable --now auditd
+ sudo systemctl enable --now auditd
Synchronizing state of auditd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable auditd
Job for auditd.service failed because the control process exited with error code.
See "systemctl status auditd.service" and "journalctl -xeu auditd.service" for details.
```



#Review logs in /var/log/auth.log and /var/log/audit/audit.log.

## Step 1: Install `auditd`

On a Debian/Ubuntu system:

```
bash

sudo apt update
sudo apt install auditd audispd-plugins -y
```

On a RHEL/CentOS system:

```
bash

sudo yum install audit -y
```

---

## Step 2: Start and Enable `auditd`

To ensure `auditd` is running and starts on boot:

```
bash

sudo systemctl enable --now auditd
sudo systemctl status auditd
```

---

## Step 3: Configure Audit Rules

Audit rules are defined in `/etc/audit/rules.d/audit.rules`. Modify this file to add rules for tracking user activities and access.

### Example Audit Rules

#### 1. Monitor User Login and Logout:

```
bash Copy Edit

echo "-w /var/log/auth.log -p wa -k auth_log" | sudo tee -a /etc/audit/rules.d/audit.rules
```

#### 2. Monitor Changes to `/etc/passwd` and `/etc/shadow`:

```
bash Copy Edit

echo "-w /etc/passwd -p wa -k passwd_changes" | sudo tee -a /etc/audit/rules.d/audit.rules
echo "-w /etc/shadow -p wa -k shadow_changes" | sudo tee -a /etc/audit/rules.d/audit.rules
```

#### 3. Audit All Commands Run as Root:

```
bash Copy Edit

echo "-a always,exit -F arch=b64 -F euid=0 -S execve -k root_commands" | sudo tee -a /etc/audit/rules.d/audit.rules
```

After adding the rules, restart `auditd`:

```
bash Copy Edit

sudo systemctl restart auditd
```

## Step 4: Review Logs

- Check Authentication Logs ( `auth.log` ):

```
bash

sudo cat /var/log/auth.log | less
```

- Look for failed logins, sudo usage, and SSH attempts.

- Check Audit Logs ( `audit.log` ):

```
bash

sudo cat /var/log/audit/audit.log | less
```

- You can filter specific events using:

```
bash

sudo ausearch -k auth_log
sudo ausearch -k root_commands
```

- Generate human-readable reports:

```
bash

sudo aureport -au # User authentication report
sudo aureport -l # Login attempts
sudo aureport -f # File access attempts
```

## Lab 8: Managing System Resources and Limits

### Objective:

Configure system resource limits for users and processes.

### Task:

1. Set Resource Limits:

#Use ulimit to set process limits for CPU time, file size, number of open files, etc.

```
vinu@DESKTOP-5K616C3:~$ ulimit -a
+ ulimit -a
real-time non-blocking time (microseconds, -R) unlimited
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 11627
max locked memory (kbytes, -l) 65536
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) 8192
cpu time (seconds, -t) unlimited
max user processes (-u) 11627
virtual memory (kbytes, -v) unlimited
```

## Common Options

| Option | Description                             |
|--------|-----------------------------------------|
| -t     | CPU time (in seconds)                   |
| -f     | Maximum file size (in blocks)           |
| -n     | Maximum number of open file descriptors |
| -u     | Maximum number of processes per user    |
| -v     | Virtual memory size (KB)                |
| -m     | Physical memory size (KB)               |
| -l     | Locked memory (KB)                      |
| -s     | Stack size (KB)                         |
| -c     | Core dump file size (blocks)            |
| -a     | Show all limits                         |

## Setting Limits

- To set a limit on CPU time (e.g., 60 seconds):

```
sh
ulimit -t 60
```

- To restrict the maximum number of open files (e.g., 1024):

```
sh
ulimit -n 1024
```

- To allow unlimited file size:

```
sh
ulimit -f unlimited
```

## Hard vs. Soft Limits

- Soft limits can be changed by the user (within the hard limit).
- Hard limits can only be changed by root.

To set a soft limit:

```
sh

ulimit -Sn 2048
```

To set a hard limit:

```
sh

ulimit -Hn 4096
```

## Permanent Changes

For permanent changes, edit `/etc/security/limits.conf` or add limits in `/etc/security/limits.d/custom.conf`:

```
yaml

username hard nproc 1000
username soft nofile 2048
```

Alternatively, update `/etc/systemd/system.conf` or `/etc/systemd/user.conf` for system

#Modify /etc/security/limits.conf to apply limits for specific users or groups.

sudo nano /etc/security/limits.conf

```
/etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain> <type> <item> <value>
#
#Where:
#<domain> can be:
- a user name
- a group name, with @group syntax
- the wildcard *, for default entry
- the wildcard %, can be also used with %group syntax,
for maxlogin limit
- NOTE: group and wildcard limits are not applied to root.
To apply a limit to the root user, <domain> must be
the literal username root.
#
```

## Add or Modify User/Group Limits

The syntax for entries is:

php-template

CopyEdit

<domain> <type> <item> <value>

- **<domain>**: Username, group (preceded by @), or wildcard (\* for all users).
- **<type>**: soft (warning before enforcing) or hard (absolute limit).
- **<item>**: Resource limit type (e.g., nofile, nproc, memlock).
- **<value>**: The numerical limit

### 3. Monitor Resource Usage:

#Use tools like top, htop, and dstat to monitor resource consumption and identify processes that exceed their limits.

top

```
top - 06:34:34 up 12:56, 2 users, load average: 0.00, 0.01, 0.00
tasks: 68 total, 1 running, 51 sleeping, 16 stopped, 0 zombie
Cpu(s): 0.1 us, 0.1 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
Mem : 2913.6 total, 2019.3 free, 513.5 used, 380.8 buff/cache
```

. htop

```
vinu@DESKTOP-5K616C3: ~
0[0.0%] Tasks: 69, 42 thr; 1 running
1[0.0%] Load average: 0.04 0.02 0.00
2[0.0%] Uptime: 12:56:57
3[0.0%]
Mem[|||||] 517M/2.85G
Swp[] 0K/1.00G

 PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
 4005 vinu 20 0 8656 4252 3348 R 1.3 0.1 0:00.11 htop
 102 root 20 0 22096 5968 4552 S 0.7 0.2 0:05.45 /lib/systemd/systemd-udev
 103 root 20 0 163M 13000 8376 S 0.0 0.4 0:11.44 /bin/init
```

dstat -c -m -d --top-cpu

```
vinu@DESKTOP-5K616C3:~$ dstat -c -m -d --top-cpu
+ dstat -c -m -d --top-cpu
+ '[' -x /usr/lib/command-not-found ']'
+ /usr/lib/command-not-found -- dstat
Command 'dstat' not found, but can be installed with
sudo apt install dstat # version 0.7.4-6.1, or
sudo apt install pcpt # version 5.3.6-1build1
```





