```
#!/bin/bash


# filename: LAMPinstall_2.1.7

# assignment title: OSYS2022 Project - Build Your Own Script

# author: Christopher Jones

# created: 2024.04.05

# last modified: 2024.04.15


# Summary: This script will install and harden a LAMP server on ubuntu.

# Functionality includes the following, in order:

        # Updating and upgrading ubuntu

        # Installing Apache2

        # Installing MySQL

        # Securing MySQL

        # Installing PHP

        # configuring automatic upgrades

        # Installing SSH

        # Securing SSH

                # Includes: generating a key pair, disabling root login, and disabling password authentication

        # Creating a user account with sudo access

        # Disabling root login

        # Installing and configuring UFW

        # modifying the /etc/apache2/apache2.conf file
```

```
### Check that script is being run with root permission ###

if [ "$EUID" -ne 0 ]; then
        echo "This script must be run as root"
        exit 1
fi
        # This checks if the EUID does not equal 0 and exits if it is not.
                # (0 is the root user's id)




### Install LAMP server ###

echo
echo "* Installing LAMP server *"
echo

# Update apt package manager in preparation for the LAMP install
echo "Updating repo.."
apt update -y
sapt upgrade -y
    # updates and upgrades the repository
    # -y automatically answers yes to all prompts


# install expect package to be used later in the script
echo -e "\n Installing expect package"
        # -e enables interpretation of backslash escapes in the string.
         # Without -e, \n would be treated as literal characters rather than a newline character.
apt install expect -y


# Install Apache web server
echo -e "\n Installing Apache2"
apt install apache2 -y


# Install MySQL database server
echo -e "\n Installing MySQL"
apt install mysql-server -y
        # MySQL installer displays password as it is entered.
        # POSSIBLE BUG: The script hangs here for a bit.
                # Is it just waiting for MySQL to finish installing?
    ### Expect is used here because -y does not work to respond to these prompts
echo -e "\n Running MySQL secure installation..."
/usr/bin/expect -c "
    # -c tells the Expect interpreter to execute commands provided inline as a script
directly from the command line.
spawn mysql_secure_installation
    # spawn is an Expect command used to start a new process
                # y\r\ means to input y and then enter
expect \"Remove anonymous users?\" { send \"y\r\" }
expect \"Disallow root login remotely?\" { send \"y\r\" }
expect \"Remove test database and access to it?\" { send \"y\r\" }
expect \"Reload privilege tables now?\" { send \"y\r\" }
expect {
  \"(Press y|Y for Yes, any other key for No)\" { send \"y\r\"; exp_continue }
  \"Enter current password for root (enter for none):\" { send \"\r\" }
  eof
}
"


# Install PHP and required modules
echo -e "\n Installing PHP"
apt install php libapache2-mod-php php-mysql -y


# Restart Apache web server
echo -e "\n Restaring apache2..."
systemctl restart apache2
```

```
# Install PHPMyAdmin

echo -e "\n Installing phpMyAdmin"

apt install phpmyadmin -y


# Prompt for PHPMyAdmin root password

echo "Enter a password for PHPMyAdmin root user:"

read -s PMA_ROOT_PASS

     # -s makes the input silent so that the characters being inputted are not displayed


# Set PHPMyAdmin root password

smysql -e "ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY
'$PMA_ROOT_PASS';"

                    # mysql -e "GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED
BY '$PMA_ROOT_PASS' WITH GRANT OPTION;"

                    # mysql -e "FLUSH PRIVILEGES;"


# Configure PHPMyAdmin with Apache

echo 'Include /etc/phpmyadmin/apache.conf' | tee -a  /etc/apache2/apache2.conf

          # this appends the echoed string to the /etc/apache2/apache2.conf file.



echo -e "\n Restarting apache2..."

systemctl restart apache2
```

```
### Install unattended-upgrades and apt-listchanges ###

echo "Installing unattended-upgrades..."

apt -y install unattended-upgrades apt-listchanges


# Prompt for user's email address

read -p "Enter your email address for upgrade notifications: " email

# Set up email notifications

echo "Configuring email notifications..."

sed -i "/^\/\/.*Unattended-Upgrade::Mail/s/.*/Unattended-Upgrade::Mail \"$email\";/"
/etc/apt/apt.conf.d/50unattended-upgrades

echo "Email notifications configured."

        # sed performs text substitutions in the specified file.

        # -i instructs sed to edit files in place. It modifies the input file directly, without
creating a new file.

        # The next part of the command is the sed expression, which consists of a pattern
and a replacement.

        # /etc/apt/apt.conf.d/50unattended-upgrades is the path to the file being edited.


# Disable automatic reboots

echo "Disabling automatic reboots..."

sed -i 's/^\/\/\s*Unattended-Upgrade::Automatic-Reboot\s*"false";/Unattended-
Upgrade::Automatic-Reboot "false";/' /etc/apt/apt.conf.d/50unattended-upgrades

echo "Automatic reboots disabled."

echo "Manual reboot required when email notification of automatic upgrade is received"
echo "unattended-upgrade installation complete."





### Install SSH ###


echo

echo "Installing SSH..."

apt install openssh-server -y


echo "Configuring SSH..."

# Create backup of sshd_config

cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original

# Write protect the backup file

chmod a-w /etc/ssh/sshd_config.original

        # chmod is used to change the permissions of a file or directory.

        # a-w specifies the permissions to be modified.

        # a stands for "all users", meaning that the permissions will be modified for all users
who have access to the file.

        # -w means to remove the "write" permission for the specified users. This effectively
makes the file read-only for all users.

# Check the syntax of /etc/ssh/sshd_config
```

```
echo "Checking configuration..."

sshd -t -f /etc/ssh/sshd_config

        # -t tests the syntax of the SSH server configuration file without actually starting the SSH server.

                #It checks the configuration for errors and prints any syntax errors or warnings to the terminal.

        # -f /etc/ssh/sshd_config specifies the path to the SSH server configuration file that should be tested.

echo "Restarting SSH..."

systemctl restart sshd.service


# Generate an SSH key

echo "Generating SSH key pair..."

yes "" | ssh-keygen -t rsa -b 4096

        # -t rsa specifies the type of key to create.

        # -b 4096 specifies the number of bits in the key.

echo "SSH key pair generated successfully."


# Modify the sshd_config file to disable root login

echo "Disabling root login via SSH..."

sed -i 's/^#PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config

    # sed is a stream editor for filtering and transforming text.

    # -i tells sed to edit files in-place. It modifies the input file directly, without creating a new file.

    # s/^#PermitRootLogin yes/PermitRootLogin no/  is the sed substitution command.

    # s indicates that sed should perform a substitution.

    # ^#PermitRootLogin yes is the pattern to search for.
```

```
        # It matches lines that begin with #PermitRootLogin yes.

        #The ^ character denotes the start of the line, and # is a literal character in this
context.

    # PermitRootLogin no: This is the replacement text. It replaces the matched pattern
with PermitRootLogin no.

    # /: This character separates the pattern and the replacement text in the sed
substitution command.

    #/etc/ssh/sshd_config: This is the file to be edited: the OpenSSH server configuration
file located at /etc/ssh/sshd_config.


# Restart the SSH service

echo "Restarting SSH..."

systemctl restart sshd.service


# Print a message indicating successful execution

echo

echo "Root login via SSH has been disabled."









### create user account ###


# Prompt the user to enter a username

read -p "Enter username for the new user account: " username
```

```
        # read is a built-in command in Bash used to read input from the user or from a file.
        # -p indicates that the message will be displayed as a user prompt

# Prompt the user to enter a password
read -s -p "Enter password for the new account: " password
        # -s makes the password input silent
echo

# Create a new user account with the specified username
echo "Creating user account '$username'..."
useradd -m -s /bin/bash "$username"
        # useradd creates the user
        # -m creates the user's home directory
        # -s /bin/bash: This option is used to specify the login shell for the new user.

# Set the password for the new user account
echo "Setting password for user '$username'..."
        # The variable $username is incorperated into the message to display the username
for which the password is being set.
echo "${username}:${password}" | chpasswd
        # updates the password database
echo "User account '$username' created and password set successfully."
```

```
### Set up SUDO Access ###

echo
echo "Setting up SUDO access..."

usermod -aG sudo "$username"
        # usermod adds account priviliges
        # -a appends the user to the specified group,
        # -G specifies the group to add the user to.
        # sudo is the name of the group.
echo "User account '$username' has sudo permission."







### Disable root logins ###

echo
echo "Disabling root login..."

# Change the “root” Default Shell
echo "Changing the root user's shell to /usr/sbin/nologin..."
echo "(The new shell in intentionally invalid)"

chsh -s /usr/sbin/nologin root

            #  changes the root user’s default shell from /bin/bash or /bin/sh to
/usr/sbin/nologin, which is invalid.

echo "Root user's shell has been changed to /usr/sbin/nologin."


# Lock the root password

echo

echo "Locking the root password..."

passwd -l root

            # -l locks the password mof the specified account

            # it can be undone with sudo passwd -u root








### Install & Configure UFW ###


# Check the status of UFW

echo

echo "Checking the status of UFW..."

ufw status


# Set rules allowing Apache and SSH traffic
```

```
echo "Setting rules allowing Apache and SSH traffic..."

ufw allow in ssh

ufw allow in 80/tcp

ufw allow in 443/tcp


# Enable logging

echo "Enabling logging..."

ufw logging on


# Enable UFW to start immediately on startup

echo "Enabling UFW..."

ufw enable


# Check the status of UFW after enabling

echo "Checking the status of UFW after enabling..."

ufw status


# Enable UFW to run at boot

systemctl enable ufw








### Set TraceEnable to off and hide ServerTokens ###
```

```
# Define the path to the apache2.conf file
APACHE2_CONF_PATH="/etc/apache2/apache2.conf"

# Check if the apache2.conf file exists
if [ ! -f "$APACHE2_CONF_PATH" ]; then
  echo "Error: apache2.conf file not found at $APACHE2_CONF_PATH"
  exit 1
fi

# Set TraceEnable to off
echo
echo "Setting TraceEnable to off in $APACHE2_CONF_PATH..."
sed -i 's/^TraceEnable.*/TraceEnable Off/' "$APACHE2_CONF_PATH"

# Hide ServerTokens
echo
echo "Hiding ServerTokens in $APACHE2_CONF_PATH..."
sed -i 's/^ServerTokens.*/ServerTokens Prod/' "$APACHE2_CONF_PATH"

# Hide ServerSignature
echo
echo "Hiding ServerSignature in $APACHE2_CONF_PATH..."
sed -i 's/^ServerSignature.*/ServerSignature Off/' "$APACHE2_CONF_PATH"

echo
```

```
echo "TraceEnable set to off, ServerTokens hidden, and ServerSignature hidden in
$APACHE2_CONF_PATH"


# Restart Apache to apply the changes

echo

echo "Restarting Apache to apply the changes..."

systemctl restart apache2


echo

echo "Apache restarted successfully"






echo

echo

echo " ***** CONGRATULATIONS ***** "

echo

echo "You have successfully installed and hardened your LAMP server"

echo "Enjoy! And have a nice day!"

echo
```