42 cursus Born To Be Root

This article is a walkthrough of the 42 project Bornt2BeRoot implemented by 42 Yerevan student *Sargis Hovsepyan* (*shovsepy*). This project aims to introduce you to the wonderful world of virtualization. You will create your first machine in VirtualBox under specific instructions. Then, at the end of this project, you will be able to set up your own operating system while implementing strict rules.

Table of Contents

The article contains the solution to the project. It would be highly effective if you read this article after gathering some information about Linux OS and the project in general.

Installation

You must choose as an operating system either the latest stable version of *Debian*, or the latest stable version of *CentOS*. Debian is highly recommended if you are new to system administration.

I created the server using the Debian operating system. At the time of writing, the latest stable version of Debian is *Debian 10 Buster* which can be downloaded from here. You can find the bonus installation walkthrough (no audio) here.

After the installation of the server, just login, get the terminal and let's start configuration of the environment.

Step 1: Installing Sudo

Switch to root and its environment by the help of following command:

```
$ su -
Password:
#
```

Just input the root credentials and switch the user to root. Next, we need to install *sudo* on the server. To do so we need to use the following command:

$ apt install *sudo*

To verify that sudo has been installed successfully you can us the command:

$ dpkg -l | grep *sudo*

If we see any output showing the packages of sudo then we can precede with the next step.

Step 2: Adding user to sudo group

Add user to sudo group via the command:

$ adduser <username> *sudo*

Of course, you should write the name of the user you are trying to add to the sudo group instead of <*username*>.

Alternatively, you can add user to sudo group via the command:

$ usermod -aG *sudo* <*username*>

To Verify whether user was successfully added to sudo group use the command:

$ getent group *sudo*

If you see the username of the user added, then everything is fine. Indeed, you should *reboot* the system for changes to take effect, then log in and verify your sudopowers via the command:

$ sudo -v

Step 3: Running root-privileged commands

From here on out, you can run root-privileged commands via prefix *sudo*. For instance:

$ sudo apt update

Configuring Sudo

In order to configure sudo you need to create a *sudo conig* file in *sudoers.d* directory. In order to work with files, we need a text editor. Indeed, we can use the default editor *vi* in the server but I preferred installing vim and using that. You can install vim via the command:

$ sudo apt install *vim*

Now let's create a sudo config file via the command:

$ sudo vim /etc/sudoers.d/<*filename*>

<*filename*> shall not end in "~" or contain "." (dot).

We need to add some lines to the *sudo config* file in order to have the sudo policy required by the project. You need to

1. Limit authentication using *sudo* to 3 attempts (defaults to 3 anyway) in the event of an incorrect password.
2. Add a custom error message in the event of an incorrect password
3. Log all *sudo* commands to */var/log/sudo/<filename>* (firstly you need to create */var/log/sudo* directory via *sudo mkdir /var/log/sudo*)
4. Require *TTY*
5. Set *sudo* paths to */usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin*

To achieve all of the above, at the end your *sudo config* file should have the following content:

```
Defaults	passwd_tries=3
Defaults	badpass_message="<custom-error-message>"
Defaults	logfile="/var/log/sudo/<filename>"
Defaults	requiretty
Defaults
secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
```

Configuring SSH

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.

Let's configure the ssh connection of our server.

Step 1: Installing ssh

In order to be able to use the ssh connection you have to install *openssh-server*. You can install it with the following command:

$ sudo apt install *openssh-server*

You can verify whether openssh-server was successfully installed via the command:

$ dpkg -l | grep *ssh*

If you see packages of ssh, then everything is fine.

Step 2: Configuring ssh

In order to change the shh port and restrict root access you need to change the *sshd_config* file. Open the SSH configuration file via the command:

$ sudo vim */etc/ssh/sshd_config*

- To set up SSH using Port 4242, replace the line:

  13 #Port 22

  with:

  13 Port 4242

- To disable SSH login as *root* regardless of authentication mechanism, replace the line:

  32 #PermitRootLogin prohibit-password

  with:

  32 PermitRootLogin no

Check ssh status via the command:

$ sudo service *ssh* status

Alternatively, you can check it with the command:

$ systemctl status *ssh*

Configuring UFW

Uncomplicated Firewall is a program for managing a net filter firewall designed to be easy to use. It uses a command-line interface consisting of a small number of simple commands, and uses iptables for configuration. It is the default firewall for *Debian Ubuntu*.

Let's install and configure the ufw for the server we created.

Step 1: Installing ufw

First, we need to install the ufw via the command:

$ sudo apt install *ufw*

You can verify whether *ufw* was successfully installed via the command:

$ dpkg -l | grep *ufw*

If you see packages of ufw, then everything is fine.

Step 2: Configuring ufw

Now we need to enable the firewall and to do so use the command:

$ sudo ufw *enable*

In order to be able to connect to the server via ssh the firewall need to allow the port we set up for ssh. Allow incoming connections using Port 4242 via the command:

$ sudo ufw *allow* 4242

Check UFW status with the command:

$ sudo ufw *status*

As we installed and configured both the firewall and ssh server, we can use ssh to connect to our server remotely.

Connecting to server via ssh

Now you can use another device (for example your host machine) to connect to our *Debian server*. In order to ssh to your server you need your machine's *username* and *ip address* (the command *hostname -I* will give you the ip address). You can SSH into your virtual machine using Port 4242 by opening a terminal and typing the following command:

$ ssh <*username*>@<*ip-address*> -p 4242

After this you will be able to login to your server and control it via ssh session.
In order to end the session type *logout* or *exit* in the terminal.

User management

We need to set up some configuration for password policy, users and groups. To set up everything according to the requirements of the project, let's follow the steps below.

Step 1: Setting Up a Strong Password Policy

We have some policy about the Password expiration and warning age. According to the project we need to:

1. Set password to expire every 30 days
2. Set minimum number of days between password changes to 2 days
3. Send user a warning message 7 days (defaults to 7 anyway) before password expiry

To do so, go ahead and open the login.defs file in text editor via the command:

$ sudo vim */etc/login.defs*

We need to change some lines in the file opened. Mainly we need to replace the lines:

160 PASS_MAX_DAYS 99999
161 PASS_MIN_DAYS 0
162 PASS_WARN_AGE 7

with:

160 PASS_MAX_DAYS 30
161 PASS_MIN_DAYS 2
162 PASS_WARN_AGE 7

After that just save the file and exit.

Step 2: Increasing Password Strength

Secondly, to set up policies in relation to password strength, install the *libpam-pwquality* package.

$ sudo apt install *libpam-pwquality*

To strengthen the password policy, you need to add some lines in the common-password file in *pam.d* directory. Go ahead and open the */etc/pam.d/* common-password file in text editor via the command:

$ sudo vim */etc/pam.d/common-password*

Now navigate to line 25 and make it look like the following:

password requisite pam_pwquality.so retry=3 minlen=10 ucredit=-1 dcredit=-1 maxrepeat=3 reject_username difok=7 enforce_for_root

Step 3: Creating a new user

Create new user via the command:

$ sudo adduser <*username*>

Verify whether user was successfully created via the command:

$ getent passwd <*username*>

Verify newly-created user's password expiry information via the command:

$ sudo chage -l <*username*>

After that you will see information about the user’s password.

Step 4: Creating a new group

Create new user42 group via the command:

$ sudo addgroup *user42*

Now, you can add user to user42 group via the command:

$ sudo usermod -aG *user42 <username>*

Verify whether user was successfully added to user42 group via the command:

$ getent group *user42*

Setting up a cron job

You need to write a shell script which should display some info about the system according to the project. Once you are done with the script you need to make it run every 10 minutes and you need to do that with the help of crontab (Make sure to see my script in the GitHub).

Configure cron as *root* via the command:

$ sudo crontab -u *root* -e

It will open a file and to schedule a shell script to run every 10 minutes, we need to replace the line:

23 # m h dom mon dow command

with:

*23 */10 * * * * sh /path/to/script | wall*

Check root's scheduled cron jobs via the command:

$ sudo crontab -u *root* -l

After this do not forget to add the *reboot* line to the same file to make it run at *reboot*.

Bonus Part

In this part your partitions are different from the mandatory part, hence you need to install with bonus partitions if you need to do the bonus part (watch the bonus installation video mentioned above).

We need to create LLMP Stack (Linux Lighttpd MariaDB PHP).

1. Lighttpd is an open-source web server optimized for speed-critical environments. You will use it as an HTTP server.
2. MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system.
3. PHP is a general-purpose scripting language especially suited to web development. It will be used to maintain our Wordpress website.

Step1: Install lighttpd

Install lighttpd via the command:

$ sudo apt install *lighttpd*

Allow incoming connections using Port 80 via the command:

$ sudo ufw *allow* 80

Step2: Install and configure MariaDB

Install mariadb-server via the command:

$ sudo apt install *mariadb-server*

Start interactive script to remove insecure default settings via the command:

$ sudo *mysql_secure_installation*

Enter current password for root (enter for none): #Just press Enter (do not confuse database root with system root)
Set root password? [Y/n] n
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y

Now you have the MariaDB and we are going to create a database, and a user for our Wordpress server. Log in to the MariaDB console via the command:

$ sudo mariadb

MariaDB [(none)]>

Create new database via the command:

MariaDB [(none)]> CREATE DATABASE <*database-name*>;

Create new database user and grant them full privileges on the newly-created database via the command:

MariaDB [(none)]> GRANT ALL ON <*database-name*>.* TO '<*username-2*>'@'*localhost*' IDENTIFIED BY '<*password-2*>' WITH GRANT OPTION;

Flush the privileges via the command:

MariaDB [(none)]> FLUSH PRIVILEGES;

You can exit the database by typing the word *exit* in the terminal.

Verify whether database user was successfully created by logging in to the MariaDB console via the command:

$ mariadb -u <*username-2*> -p

Enter password: <*password-2*>

MariaDB [(none)]>

Now you can Confirm whether database user has access to the database via the command:

MariaDB [(none)]> SHOW DATABASES;

Step3: Install PHP

Install php-cgi & php-mysql via the command:

$ sudo apt install *php-cgi php-mysql*

Step4: Install and configure Wordpress

To retrieve content from web servers we need to install wget. Install wget via the command:

$ sudo apt install *wget*

Download WordPress to */var/www/html* via the command:

$ sudo wget *http://wordpress.org/latest.tar.gz* -P */var/www/html*

Extract downloaded content via the command:

$ sudo tar -xzvf */var/www/html/latest.tar.gz*

After you extract the wordpress files you no more need the latest.tar.gz file. Remove tarball via the command:

$ sudo rm */var/www/html/latest.tar.gz*

Copy content of */var/www/html/wordpress* to */var/www/html* via the command:

$ sudo cp -r */var/www/html/wordpress/** */var/www/html*

Remove */var/www/html/wordpress* via the command:

$ sudo rm -rf */var/www/html/wordpress*

Create WordPress configuration file from its sample via the command:

$ sudo cp */var/www/html/wp-config-sample.php /var/www/html/wp-config.php*

Configure WordPress to reference previously-created MariaDB database & user via the command:

$ sudo vim */var/www/html/wp-config.php*

After the file opens, replace the lines below:

23 define('DB_NAME', 'database_name_here');^M
26 define('DB_USER', 'username_here');^M
29 define('DB_PASSWORD', 'password_here');^M

with:

*23 define('DB_NAME', '<**database-name**>');^M*
*26 define('DB_USER', '<**username-2**>');^M*
*29 define('DB_PASSWORD', '<**password-2**>');^M*

Step5: Configure Lighttpd

Enable some modules via the commands:

$ sudo *lighty-enable-mod fastcgi*

$ sudo *lighty-enable-mod fastcgi-php*

$ sudo service *lighttpd* force-reload

Step6: Install and Configure FTP

Install FTP via the command:

$ sudo apt install *vsftpd*

Allow incoming connections using Port 21 via the command:

$ sudo ufw *allow* 21

Configure vsftpd via the command:

$ sudo vim */etc/vsftpd.conf*

To enable any form of FTP write command, uncomment below line:

31 #write_enable=YES

To prevent user from accessing files or using commands outside the directory tree, uncomment below line:

114 #chroot_local_user=YES

To set *root* folder for FTP-connected user to */home/<username>/ftp*, execute below lines:

$ sudo mkdir */home/<username>/ftp*

$ sudo mkdir */home/<username>/ftp/files*

$ sudo chown nobody:nogroup */home/<username>/ftp*

$ sudo chmod a-w */home/<username>/ftp*

To whitelist FTP, execute below lines:

$ sudo vim */etc/vsftpd.userlist*

$ echo *<username>* | sudo tee -a */etc/vsftpd.userlist*

Step6: Connecting to server via FTP

FTP into your virtual machine via the command:

$ ftp *<ip-address>*

Terminate FTP session at any time via *CTRL + D*

This is the end of the installation and configuration of the project ***Born2BeRoot****.*