How to secure Linux Home Server

There are many reasons to set up a home server. You can use it as a media server, file server or even a local backup server.

There are many reasons to set up a home server. You can use it as a media server, file server or even a local backup server. Basically, any file that doesn't need to be online is a good candidate for home servers. Setting up a home server running Linux is relatively easy, especially in today's time. However, keeping that server safe is a completely different story. Security is a difficult but extremely important task.

Do you know how to secure Linux Home Server?

1. Only install what you really need
2. Configuring sudo
3. SSH configuration
4. Configure the firewall
5. Always pay attention to updates!

Only install what you really need

One of the easiest ways to secure your home server is to focus on security at the beginning. Let's start with the installation. If you are not sure you need an application or service, do not install it. You can always install it later if needed.

If you have installed Linux a few times, this is even easier. Instead of sticking with default options, use modes that allow you to control the most settings. Sometimes these are named **'expert mode'** or something similar

Careful monitoring of installation options can save you time to disable services for security reasons later.

Configuring sudo

Before switching to any other step, you need to configure **sudo.** Why? Because you will login to your server via SSH and you cannot login with root account. To make any further changes to the system, you will need to use **sudo.**

First, check if you can use **sudo** . From the user account, run the following command with an alternate real user name for **USERNAME:**

```
sudo -lU USERNAME
```

If you see a message displayed indicating that the username can be run **'(ALL) ALL'** or something similar, you're ready to continue.

Now, as the root account on the server, run the following command to edit the file **'/ etc / sudoers'.** If you like another editor, use it instead of **nano.**

```
EDITOR=nano visudo
```

Edit the file and include the following items, with alternate real usernames for **USERNAME:**

```
USERNAME ALL=(ALL) ALL
```

SSH configuration

OpenSSH 8.0 released April 18, 2019

Please enable SSH on the home server. In fact, you can do so, because this is often the way you interact with the server.

First, make sure OpenSSH is installed. If you use another distribution, the command will change, but the package name is quite consistent. On Ubuntu, run the following command:

```
sudo apt install openssh-server
```

Using key-based authentication is much safer than password authentication, so we'll set up SSH to work this way. To do this, make sure you are working on the client that you intend to connect to the server, not the server itself. First, you want to make sure you don't have any SSH keys:

```
ls ~/.ssh/
```

If you see **'id_rsa'** and **'id_rsa.pub',** among the file names listed, you already have SSH key. Skip this next step.

```
ssh-keygen -t rsa -b 4096 -C "youremail@domain.com"
```

Now you will copy the SSH key to the server:

```
ssh-copy-id USERNAME@SERVER
```

For home servers, you may be using IP addresses for servers instead of names. If you don't know your server name, use the IP address instead of **SERVER** above.

Now, we will modify the SSH settings for more security. Log in to the server from the client you created the keys for. This will allow you to login again after this step. Execute the following command, replace **nano** with the editor you choose.

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

Edit the file with the following settings. They will be placed in different places in the file. Make sure there are no copies, because only the first version of the installation is tracked.

```
ChallengeResponseAuthentication no PasswordAuthentication no UsePAM no PermitRoo
```

Now, you need to restart the SSH server with one of the following commands.

On Ubuntu, run:

```
sudo systemctl restart ssh
```

Configure the firewall

Depending on the service you are running and the number of servers connected to the Internet, you may want to use a firewall. There are several options for this, but the method that has been tested and works well on Linux is **iptables.**

Iptables setup is beyond the scope of this article, but don't worry!**TipsMake.com** has a complete guide to how to configure iptables on Linux computers.

Another easier way to set up a firewall is to use **ufw.** You can install it with the command:

```
sudo apt install ufw
```

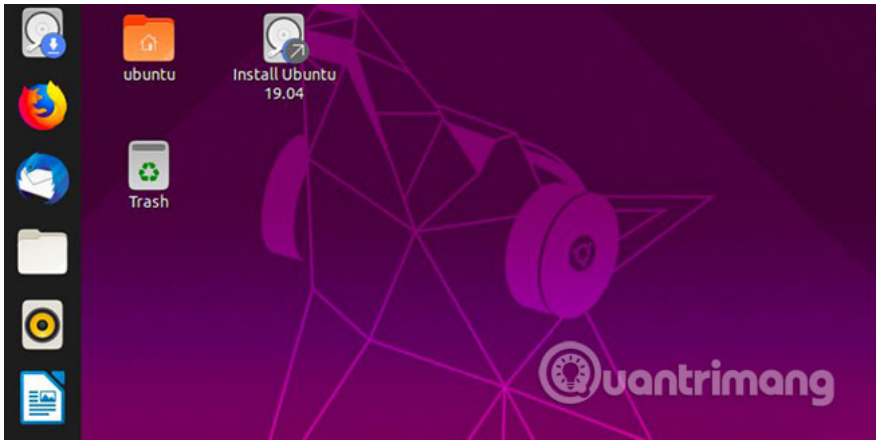
By default, this will block all ports. To enable SSH access and online, run the following **ufw** commands to open ports **80, 443** and **22:**

```
sudo ufw allow 80 sudo ufw allow 443 sudo ufw allow 22
```

And finally, activate **ufw:**

```
sudo ufw enable
```

Always pay attention to updates!



Servers can be easily forgotten, if they still work normally, but this can be very dangerous. Make sure your software is always up to date. You can use unsupervised updates (automatic updates), but nothing can be predicted. The safest way is to schedule weekly or monthly server maintenance to make sure everything is okay.

You now have a good start to keep your server protected from external threats. What if you need to access the server from home? Every door you open is likely to be exploited and exploited by an attacker.

One of the easiest ways to access your home network from outside is to use VPN. See a list of the best available security VPN services available from **TipsMake.com** to know what is the right choice for you!

Hope you are succesful.

You finished reading the article "**How to secure Linux Home Server**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
