How to Harden Ubuntu Server

Quick steps to secure your node.

> Thank you for your support and kind messages! It really energizes us to keep creating the best crypto guides. Use cointr.ee to find our donation addresses and share your message.

Pre-requisites

- Ubuntu Server or Ubuntu Desktop installed
- SSH server installed
- a SSH client or terminal window access

In case you need to install SSH server, refer to:

How to install SSH server on Ubuntu
www.simplified.guide

In case you need a SSH client for your operating system, refer to:

How to Connect to an SSH Server from Windows, macOS, or Linux
An SSH client allows you to connect to a remote computer running an SSH server. The Secure Shell (SSH) protocol is often used for
www.howtogeek.com

Create a non-root user with sudo privileges

Make a habit of logging to your server using a non-root account. This will prevent the accidental deletion of files if you make a mistake. For instance, the command rm can wipe your entire server if run incorrectly using by a root user.

Tip: Do NOT routinely use the root account. Use `su` or `sudo`, always.

SSH to your server

```
ssh username@server.public.ip.address
# example
# ssh myUsername@77.22.161.10
```

Create a new user called cardano

```
useradd -m -s /bin/bash cardano
```

Set the password for cardano user

```
passwd cardano
```

Add cardano to the sudo group

```
usermod -aG sudo cardano
```

Disable SSH password Authentication and Use SSH Keys only

> The basic rules of hardening SSH are:
>
> - No password for SSH access (use private key)
> - Don't allow root to SSH (the appropriate users should SSH in, then `su` or `sudo`)
> - Use `sudo` for users so commands are logged
> - Log unauthorized login attempts (and consider software to block/ban users who try to access your server too many times, like fail2ban)
> - Lock down SSH to only the ip range your require (if you feel like it)

Create a new SSH key pair on your local machine. Run this on your local machine. You will be asked to type a file name in which to save the key. This will be your **keyname**.

Your choice of ED25519 or RSA public key algorithm.

ED25519

```
ssh-keygen -t ed25519
```

RSA

```
ssh-keygen -t rsa -b 4096
```

Transfer the public key to your remote node. Update the **keyname**.

```
ssh-copy-id -i $HOME/.ssh/<keyname>.pub cardano@server.public.ip.address
```

Login with your new cardano user

```
ssh cardano@server.public.ip.address
```

Disable root login and password based login. Edit the `/etc/ssh/sshd_config file`

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

Locate **ChallengeResponseAuthentication** and update to no

```
ChallengeResponseAuthentication no
```

Locate **PasswordAuthentication** update to no

```
PasswordAuthentication no
```

Locate **PermitRootLogin** and update to no

```
PermitRootLogin no
```

Locate **PermitEmptyPasswords** and update to no

```
PermitEmptyPasswords no
```

Optional: Locate **Port** and customize it your **random** port.

> Use a **random** port # from 1024 thru 49141. Check for possible conflicts.

```
Port <port number>
```

Validate the syntax of your new SSH configuration.

```
sudo sshd -t
```

If no errors with the syntax validation, reload the SSH process

```
sudo service sshd reload
```

Verify the login still works

Standard SSH Port 22

```
ssh cardano@server.public.ip.address
```

Custom SSH Port

```
ssh cardano@server.public.ip.address -p <custom port number>
```

> Alternatively, you might need to use the following.
>
> Add the `-p <port#>` flag if you used a custom SSH port.

```
ssh -i <path to your SSH_key_name.pub> cardano@server.public.ip.addr
```

Update your system

> It's critically important to keep your system up-to-date with the latest patches to prevent intruders from accessing your system.

```
sudo apt-get update -y && sudo apt-get upgrade -y
sudo apt-get autoremove
sudo apt-get autoclean
```

Enable automatic updates so you don't have to manually install them.

```
sudo apt-get install unattended-upgrades
sudo dpkg-reconfigure -plow unattended-upgrades
```

Disable root account

System admins should not frequently log in as root in order to maintain server security. Instead, you can use sudo execute that require low-level privileges.

```
# To disable the root account, simply use the -l option.
sudo passwd -l root
```

```
# If for some valid reason you need to re-enable the account, simply use the
sudo passwd -u root
```

Setup Two Factor Authentication for SSH

> SSH, the secure shell, is often used to access remote Linux systems. Because we often use it to connect with computers containing important data, it's recommended to add another security layer. Here comes the two factor authentication (*2FA*).

```
sudo apt install libpam-google-authenticator -y
```

To make SSH use the Google Authenticator PAM module, edit the `/etc/pam.d/sshd` file:

```
sudo nano /etc/pam.d/sshd
```

Add the follow line:

```
auth required pam_google_authenticator.so
```

Now you need to restart the `sshd` daemon using:

```
sudo systemctl restart sshd.service
```

Modify `/etc/ssh/sshd_config`

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

Locate **ChallengeResponseAuthentication** and update to yes

```
ChallengeResponseAuthentication yes
```

Locate **UsePAM** and update to yes

```
UsePAM yes
```

Save the file and exit.

Run the **google-authenticator** command.

```
google-authenticator
```

It will ask you a series of questions, here is a recommended configuration:

- Make tokens “time-base””: yes
- Update the `.google_authenticator` file: yes
- Disallow multiple uses: yes
- Increase the original generation time limit: no
- Enable rate-limiting: yes

You may have noticed the giant QR code that appeared during the process, underneath are your emergency scratch codes to be used if you don’t have access to your phone: write them down on paper and keep them in a safe place.

Now, open Google Authenticator on your phone and add your secret key to make two factor authentication work.

Note: If you are enabling 2FA on a remote machine that you access over SSH you need to follow **steps 2 and 3** of this tutorial to make 2FA work.

Secure Shared Memory

One of the first things you should do is secure the shared memory used on the system. If you're unaware, shared memory can be used in an attack against a running service. Because of this, secure that portion of system memory.

To learn more about secure shared memory, read this techrepublic.com article.

One exceptional case

There may be a reason for you needing to have that memory space mounted in read/write mode (such as a specific server application like **Chrome** that requires such access to the shared memory or standard applications like Google Chrome). In this case, use the following line for the fstab file with instructions below.

```
none /run/shm tmpfs rw,noexec,nosuid,nodev 0 0
```

The above line will mount the shared memory with read/write access but without permission to execute programs, change the UID of running programs, or to create block or character devices in the namespace. This a net security improvement over default settings.

Use with caution

With some trial and error, you may discover some applications(**like Chrome**) do not work with shared memory in read-only mode. For the highest security and if

> compatible with your applications, it is a worthwhile endeavor to implement this secure shared memory setting.
>
> Source: techrepublic.com

Edit `/etc/fstab`

```
sudo nano /etc/fstab
```

Insert the following line to the bottom of the file and save/close.

```
tmpfs /run/shm  tmpfs ro,noexec,nosuid  0 0
```

Reboot the node in order for changes to take effect.

```
sudo reboot
```

Install Fail2ban

> Fail2ban is an intrusion-prevention system that monitors log files and searches for particular patterns that correspond to a failed login attempt. If a certain number of failed logins are detected from a specific IP address (within a specified amount of time), fail2ban blocks access from that IP address.

```
sudo apt-get install fail2ban -y
```

Edit a config file that monitors SSH logins.

```
sudo nano /etc/fail2ban/jail.local
```

Add the following lines to the bottom of the file.

> **Whitelisting IP address tip**: The `ignoreip` parameter accepts IP addresses, IP ranges or DNS hosts that you can specify to be allowed to connect. This is where you want to specify your local machine, local IP range or local domain, separated by spaces.
>
> ```
> # Exampleignoreip = 192.168.1.0/24 127.0.0.1/8
> ```

```
[sshd]
enabled = true
port = <22 or your random port number>
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
# whitelisted IP addresses
ignoreip = <list of whitelisted IP address, your local daily laptop/pc>
```

Save/close file.

Restart fail2ban for settings to take effect.

```
sudo systemctl restart fail2ban
```

Configure your Firewall

The standard UFW firewall can be used to control network access to your node.

With any new installation, ufw is disabled by default. Enable it with the following settings.

- Port 22 (or your random port #) TCP for SSH connection
- Port 6000 TCP for p2p traffic
- Port 3000 TCP for Grafana web server (if hosted on current node)
- Port 9090 tcp for Prometheus export data (optional, if hosted on current node)

```
ufw allow <22 or your random port number>/tcp
ufw allow 6000/tcp
ufw allow 3000/tcp
ufw enable
ufw status numbered
```

> ⚠ Do not expose Grafana (port 3000) and Prometheus endpoint (port 9090) to the public internet as this invites a new attack surface! A secure solution would be to access Grafana through a ssh tunnel with Wireguard.

Only open the following ports on nodes behind a network firewall.

> **It is dangerous to open these ports on a VPS/cloud node.**

```
sudo ufw allow 3000/tcp
sudo ufw allow 9090/tcp
```

Confirm the settings are in effect.

```
     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 3000/tcp                   ALLOW IN    Anywhere
[ 3] 6000/tcp                   ALLOW IN    Anywhere
[ 4] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 5] 3000/tcp (v6)              ALLOW IN    Anywhere (v6)
[ 6] 6000/tcp (v6)              ALLOW IN    Anywhere (v6)
```

[Optional but recommended] Whitelisting (or permitting connections from a specific IP) can be setup via the following command.

```
sudo ufw allow from <your local daily laptop/pc>
# Example
# sudo ufw allow from 192.168.50.22
```

Port Forwarding Tip: You'll need to forward and open ports to your validator. Verify it's working with https://www.yougetsignal.com/tools/open-ports/ or https://canyouseeme.org/ .

Verify Listening Ports

If you want to maintain a secure server, you should validate the listening network ports every once in a while. This will provide you essential information about your network.

```
netstat -tulpn
```

```
ss -tulpn
```

Congrats on completing the guide.

Did you find our guide useful? Send us a signal with a tip and we'll keep updating it.

It really energizes us to keep creating the best crypto guides.

Use cointr.ee to find our donation addresses.

> Any feedback and all pull requests much appreciated.
>
> Hang out and chat with fellow stake pool operators on Discord @
>
> https://discord.gg/w8Bx8W2HPW
>
> Hang out and chat with our stake pool community on Telegram @ https://t.me/coincashew

References

How to Harden your Ubuntu 18.04 Server

SSH is essential to server management. And although Linux is considered to be secure out of the box, it is advisable to follow

medium.com

Ubuntu system hardening guide for desktops and servers

Step by step guide to secure any Ubuntu desktop or server. Harden your Ubuntu during installation and afterwards with the available

linux-audit.com

How To Harden OpenSSH on Ubuntu 18.04 | DigitalOcean

Linux servers are often administered remotely using SSH by connecting to an OpenSSH server, which is the default SSH server

www.digitalocean.com

Configure SSH to use two-factor authentication | Ubuntu

Ubuntu is an open source software operating system that runs from the desktop, to the cloud, to all your internet connected

ubuntu.com

https://gist.github.com/lokhman/cc716d2e2d373dd696b2d9264c0287a3#file-ubuntu-hardening-md

How to Harden Ubuntu Server 18.04 in 5 Easy Steps

Don't assume your Ubuntu Server platform is perfectly secure, out of the box. With these five easy steps, gain a serious bump in

www.lifewire.com

The 50 Best Linux Hardening Security Tips: A Comprehensive Checklist

Although Linux is much more secure when compared to home operating systems, admins still need to maintain a set of Linux

www.ubuntupit.com