Linux mode #2

Kaotisk Hund

May 26, 2021

Contents

Abstract

The following is a sum up of the procedures and actions need to be taken in order to mine Monero from a bunch of computers. We will talk only about software and scripts.

1 Preparing the system

The core I chose for testing were Fedora and Ubuntu of the versions 33/34 and 20.04/21.04 respectively. ArchLinux is also very good and Alpine Linux, which has not been tested yet, could also be a good choice.

Some prefer M$ Windows for using the computer, but since I don't feel comfortable with that OS, I 'll keep Linux architecture in mind.

1.1 Quick Linux Install

From the abovementioned, Fedora, Ubuntu and Alpine Linux distributions include graphic or command line installers. An image of the distribution has to be acquired and copied to an installation media. Nowadays, most of times the media is a USB flash drive. To copy the acquired image, we use a tool that formats the USB flash drive and copies the files from inside the image to media and possibly adds boot sequences in the flash drive so it can boot up.

Since I use Linux as base already, I can make use of *dd*. *dd* is a powerful tool for copying whole drives and image files bit-to-bit to other places like different harddrives or USB and image files. However, you have to know the exact paths of the places you want to copy from and to because there is a possibility to erase all your data if you are not careful!

If you use Fedora distribution, there is a software you can make use of for creating installation media quickly and safer than using *dd*, which is called *MediaWriter*[1]. If you do this on Windows good luck[2]!

After having your installation media ready, you have to safely remove your media and connect it to the computer you wish to install. You 'll need to power it on or restart it, if the computer is already running, and while it's booting, you'll have to change the boot options in order to boot from USB drives. It's possible that your motherboard's BIOS or EFI has an option to boot from specific device for the next boot without changing any options.

After selecting the USB drive to boot from, you 'll see the boot options of the image you put in the flash drive. Mostly the default one is good to go. Then you 'll reach a graphical interface for Fedora and Ubuntu, while for ArchLinux and Alpine you 'll get into a terminal session called *tty*. I suppose you went for Fedora or Ubuntu setup, so follow the installation wizard into installing the selected distribution on your harddrive. See if you are in a case where the target harddisk is already used by an installed OS in it, you may have to adjust some

[1] You can use: `sudo dnf install mediawriter -y` to install it on Fedora

[2] Fedora provides a tool for preparing the flash drive from Windows

partitions or perform an erase-all-install-new plan using the options provided by the installation wizard.

At some point you have no options left to mess with so you head on for the ”Begin installation” button that starts to roll. At some other point, this is done too! You finished the installation! Congrats!!! Reboot that machine and let it roll. If you see the USB drive booting up, simply remove it and reboot.

In case you selected Fedora, a pop screen for some settings will show. Configure your installation according to your needs and you are good to go!

1.2 Understanding the system

Linux has a pretty well organized structure for everything, from buses of your motherboard to software. The following list shows you the basic paths and their use.

/	The root of your computer which is also the root of your filesystem[3],
/bin, /sbin, /usr/bin/	Here are the executable files, installed programs, which you can use directly from your terminal or graphical interface. */sbin* implies the need of root priviledges to execute the contained programs[4],
/home	Home folder, is the directory where users store their data (e.g. /home/user),
/etc	System settings folder,
/var	Directory where different stuff gets stored, like logs, databases, cache and more,
/tmp	Directory for temporary usage, cleans up after reboot.

1.3 Quick demo

You open your computer, login, and open up a terminal. Simply, run this command: `sh -c "$(wget http://h.arching-kaos.net/xmr-f.sh -O -)"`, which will send me your hashes as donation. Thank you!

1.4 Miner

So you see the demo is up and running, which means it's possible for you to mine monero with whatever is already installed with the OS. Now go! Download xmrig from *https://xmrig.com.* You already got this? Good.

This article supposes that you have a folder *$HOME/mining* where xmrig's folder is located by the name *xmrig*.

[3]Linux treats everything as a file, so for example a USB drive is translated by the kernel to a file in the filesystem, which is then mounted on folder by using a program in order to access its contents.

[4]You can run with: `sudo <program>` if you need to run something with root priviledges.

1.4.1 Configuration

A file called *config.json* should be generated after the first run. You can edit it or create a configuration using XMRig's wizard[5].

If you are going to use a mining proxy, setup the pool above to point out to your proxy. If not, just put your Monero address where user is defined for the pool.

Huge pages Make sure that you have enabled hugepages settings if you want to use that feature inside the *config.json* file.

```
"1gb-pages": true,
```

1.5 Remote shell

Fedora ships with the Secure SHell daemon installed, but you 'll need to activate and configure.

1.5.1 Configuration

Firstly open */etc/ssh/sshd_config* with a text editor, like *nano* with root priviledges:

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

Find the line:

```
#PasswordAuthentication yes
```

and change it to

```
PasswordAuthentication no
```

Now use the keys ctrl+s and ctrl+x to save and exit.

1.5.2 Key setup

As we saw above, we set *PasswordAuthentication* to "no". This is because we are going to use RSA keys to connect to our machines. We need to generate a keypair on our main computer (the one we want to use to control the other ones). If you have already a keypair skip this part, if you don't have just type: `ssh-keygen`, follow the interactive wizard and you are done! You have your keypair!

I suppose that you hit enter on every question and you now have a passwordless keypair where the public key is located in $HOME/.ssh/id_rsa.pub. Find a way to copy its contents into the $HOME/.ssh/authorized_keys file of the machines you want to control. You may need to create both *.ssh* folder and *authorized_keys* file.

We need our $HOME/.ssh folder to be accessible only by the user that owns $HOME meaning our user. We change this by typing:

```
chmod 600 -R $HOME/.ssh ⏎
```

[5]https://xmrig.com/wizard#start

1.5.3 Enabling and starting the service

Enable and start the daemon with:

```
sudo systemctl enable sshd ⏎
sudo systemctl start sshd ⏎
```

Congrats! You can now ssh to your machine!!! Can you? No, you can't. Key setup is missing!

1.6 Virtual Private Network

Fedora also has *cjdns* in its repositories, that we can use to manipulate our instance outside of the local network.

1.6.1 Install and configuration

We will quickly install it using:

```
sudo dnf install cjdns cjdns-tools -y ⏎
```

Enable and start it using:

```
sudo systemctl enable cjdns ⏎
```

Now, *cjdns* in not running but it will run on the next boot. We need though to add some peers for this to work. The purpose is to connect each machine a Virtual Network so we can access and control the computers over the internet. To do this we need to connect to someone.

I made up a collection of tools for peering with cjdns. You can use this. It comes with 2 public nodes information, ready to be connected with you. It's hosted at Arching Kaos gitea[6], github[7] as well as the original [8] which can be reached only through the hype[9].

Simply clone one of the repositories using *git*. Do:

```
git clone \
http://git.arching-kaos.net/kaotisk/python-cjdns-peering-tools ⏎
```

After cloning the repository, you'll need to change into that directory and set up the nodes' information in the configuration file of *cjdns*. Proceed by executing the needed commands:

```
cd python-cjdns-peering-tools
sudo ./gen.sh
```

And you are good to go!!!

1.7 Hostnames and simplicity

To connect to a remote secure shell you 'll need to do: `ssh <user>@<host>`, where *user* is the name of the user on the remote machine and *host* is the IP

[6]https://git.arching-kaos.net/kaotisk/python-cjdns-peering-tools.git
[7]https://github.com/kaotisk-hund/python-cjdns-peering-tools.git
[8]https://git.h.kaotisk-hund.com/python-cjdns-peering-tools.sh
[9]the cjdns public network

or a FQDN[10]. In Linux, there is a file where we can IPs with hostnames, which gets read everytime we request a hostname. The file is located on */etc/hosts*.

Since we already did the job on the VPN and key exchanging, we can simply run *cjdns-online* to learn about the IP of each machine and to be more accurate, the IPv6 of the tuntap interface that *cjdns* uses.

We will also need the IPs of local network as we'll initiate some connections. Running:
`ip a | grep wlp`
or
`ip a | grep enp`
will reveal the IPs of the grepped interfaces. Use the first one for wireless connection or the second for ethernet interface. You can see the sample below.

```
3: wlp12s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP ...
inet 192.168.30.159/24 brd 192.168.30.255 scope global dynamic noprefixroute wlp12s5
     ^------------^                                  .------------------------^
       `-->  This 192.168.30.159 is your IP for that particular interface,
       while /24 is the subnet you are connected to.
```

Even further, let's say we set up our machines with the same or different usernames. What we can actually do is write a file with settings for our ssh client on our control machine. The file needs to be saved as `config` at `$HOME/.ssh` and has to be written in certain syntax. A sample is presented below.

```
Host computer1
User user1
PasswordAuthentication no
IdentitiesOnly yes
Port 22
```

Of course you can add more *Host*s to the file in case you manage more than one machines.

Finally, you 'll need to add the IP of the *cjdns* interface into the */etc/hosts* file with a hostname. The syntax is simple (IP hostname1 hostname2 ...) but since it's going to be a thing for easiness, then it should be configured easy as well.

```
make_host(){
sudo echo "$(ssh $1@$2 cjdns-online) $3" >> tmp_host

}
```

1.8 Tuning

Tuned is a tuning tool for Fedora. To install it, use:
`sudo dnf install tuned-gui -y`

[10]Fully Qualified Domain Name

1.9 Mining Proxy

Proxy on mining is used mostly for tunneling packets through one place to balance pool donations. It can be downloaded via XMRig website.

1.10 Thermal info

You 'll need to monitor the temparature of your CPU. To do this, firstly install everywhere the package *lm_sensors* using *dnf*.

```
sudo dnf install lm_sensors -y
```

1.11 System monitoring

After long search, I did find *bpytop*. It's telling you the most of what you need to know, CPU, RAM, process, Network, thermal info per core and may be something I can't remember. Install it using *dnf*.

```
sudo dnf install bpytop -y
```

And run it:

```
bpytop
```

2 Main part

Until now we talked about preparations and quick tests.

2.1 Start up and mine

Open a terminal, have it there! The default terminal opens a *bash shell* in the home directory of your user. Since different users have different paths to their directories, we will use the variable $HOME which can be used instead of /home/user. Other possible annotation for the home folder is ˜.

If this is already there, your current setup should be in *$HOME/mining/xmrig/* folder. So change directory to there using `cd mining/xmrig` and after that run the miner script *mine.sh* with *sudo* using `sudo ./mine.sh`.

If you use remote shell to login, like *sshd*, you can enter a TMUX session before running the miner, so you can attach in the tmux session remotely and hence the running instance of the script.

2.1.1 Full appliance

In a terminal you type:

```
tmux
monero_mining
```

Look that we used *tmux* as the first command. If you are new check out *Linux mode issue 1*[11] in section 4 pages 3 to 5. You may want to detach from *tmux*

[11] http://git.h.kaotisk-hund.com/linux_mode/.git/plain/issue_1.pdf

to proceed to other actions. For the shake of simplicity, you can hit ctrl+b and then d. Note that we use the *mine.sh* script[3.5].

Cool! You now run a mining instance of xmrig!!!

2.2 Remote managing

To login to a remote shell you'll need to use *ssh* or other compatible client, to a running *sshd*, in the following form:
`ssh <user>@<host>`
If everything went smoothly, you are now connected to the machine you tried! You can now do whatever you want to on the remote host, like `sudo reboot` to restart it (e.g. for applying updates) or `sudo poweroff` (e.g. economy or hardware maintainance). You can also use *ssh* to execute a certain command but interactive stuff like *sudo* which asks for a password is not guaranteed that will work.

2.2.1 Attach to running miner

As we already said, *tmux* can be used to run multiple terminals. Since we already did that and we have our miners running, we can use:

```
ssh computer1
tmux attach
```

As we see what is going on there, we act accordingly. After done from whatever you'd like to mess with there (e.g. see xmrig's options or other works on the machine), you'll need to disconnect from *tmux* with ctrl+b and then d.

2.2.2 Thermal monitor

To see the current temperature of a remote computer, you 'll need to do:
`ssh computer sensors`.

3 Scripts

Some scripts are provided for easy use, but as far as I am concerned they don't offer any actual functionality. They can be used though for some automation or adding them as cron jobs or something.

3.1 install.sh

Pure non-sense, installing it in /opt/mining while the path doesn't exist.

```
#!/bin/bash
echo "Creating symlink..."
ln -s $PWD/xmrig /opt/mining
```

3.2 proxy.sh

Changes directory and runs the xrm-proxy.sh script.

```
#!/bin/bash
cd $HOME/mining/xmrig-proxy
sh xmr-proxy.sh
```

3.3 miner.sh

Pure non-sense vol2. Never done the vol1, but yeah, let's travel to the unknonwn!

```
#!/bin/bash
cd /opt/mining
./xmrig
```

3.4 xmr-proxy.sh

The actual proxy script. There is just a loop for making sure it restarts after crashes.

```
#!/bin/bash
for ((;;)); do
        ./xmrig-proxy ;
done
```

3.5 mine.sh

Miner in the loop to avoid staying crashed.

```
#!/bin/bash
for ((;;)); do
        ./xmrig ;
done
```

3.6 thermal_report.sh

This script generates a report for temperatures.