navigation

- Main page
- Table of Contents
- Recent changes
- Random page
- Help

search

Search

Go Search

tools

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link
- Page information

Installing Parabola on Libreboot with full disk encryption (including /boot)

i18n

Dansk – Deutsch – English – Esperanto – Español – Euskera – Français – Galego – Indonesia – Italiano – Lietuviškai – Magyar – Nederlands – Polski – Português – Română – Slovenský – Suomi – Svenska – Türkçe – Česky – Ελληνικά – Български – Русский – Српски – Українська – עברית – العربية – ไทย – 日本語 – 正體中文 – 简体中文 – 한국어

Libreboot on x86 uses the GRUB payload by default, which means that the GRUB configuration file (where your libreboot GRUB menu comes from) is stored directly alongside libreboot and its GRUB payload executable inside the flash chip. This means that both installing and managing GNU/Linux distributions are handled slightly differently compared to traditional BIOS systems.

On most systems, it is necessary to have at least an unencrypted `/boot` partition (while the others, including root, may be encrypted). This is so that GRUB, and therefore the kernel, can be loaded and executed, because the boot firmware itself can't open a LUKS volume. Not so with libreboot! Since GRUB is already included directly in the boot flash even `/boot` can be encrypted. This protects its contents from tampering by someone with physical access to the system.

Note that this guide is for the GRUB payload only. If you use some other payload you can ignore this entirely.

Contents [hide]

1 Booting

Boot Parabola's install environment (see **How to boot a GNU/Linux installer on Libreboot**).

This guide will go through the installation steps taken at the time of writing, which may or may not change due to the volatile nature of Parabola (it changes all the time). This guide has been confirmed to work with the **Main live ISO 2019.03.10 image** but most of it should stay the same over time. If you spot any mistakes, please correct them!

2 Wiping the disk

This section deals with wiping the storage device on which you plan to install Parabola GNU/Linux-libre. Follow these steps, but if you use an SSD, also

- beware that there are issues with TRIM (not enabled through LUKS) and security issues if you do enable it (see **this page** for more info);
- make sure that it's brand-new (or barely used), or, otherwise, be sure that it never previously contained plaintext copies of your data;
- make sure to read **this article**. Edit /etc/fstab later on when chrooted into your install. Also, read the whole article and keep all points in mind, adapting them for this guide.

This article describes how to completely reset the SSD's cells, restoring it to its original write performance.

Wipe the MBR (if you use MBR):

```
# lsblk
```

Your storage is probably /dev/sda, but be very sure to double check this or you WILL lose your data!

```
# dd if=/dev/zero of=/dev/sda bs=446 count=1; sync
```

Securely wipe the drive:

```
# dd if=/dev/urandom of=/dev/sda; sync
```

NOTE: If you have an SSD, only do this the first time. If it was already LUKS-encrypted before, use the info below to wipe the LUKS header. Also, check online for your SSD what the recommended erase block size is. For example if it was 2MiB:

```
# dd if=/dev/urandom of=/dev/sda bs=2M; sync
```

If your drive was already LUKS encrypted (maybe you are re-installing your distro) then it is

already 'wiped', except for the LUKS header. **This article** explains how you can wipe only the LUKS header. It recommends doing the first 3MiB. Now, that guide is recommending putting zero there. We're going to use urandom. Do this:

```
# head -c 3145728 /dev/urandom > /dev/sda; sync
```

Wiping the LUKS header is good because it contains hashed passphrases which could potentially constitute a risk.

3 Change keyboard layout

The Parabola live shell assumes US Qwerty. If you have something different, list the available keymaps and use yours:

```
# localectl list-keymaps
# loadkeys LAYOUT
```

4 Establish an internet connection

See **this guide**. Wired is recommended, but wireless is also explained.

5 Getting started

This section is based on the **Installation Guide** and **Partitioning**.

5.1 dm-mod

`device-mapper` will be used -- a lot. Make sure that the kernel module is loaded:

```
# modprobe dm-mod
```

5.2 Create LUKS partition

If you are fine with MBR partitioning, then you can use cfdisk:

```
# cfdisk /dev/sda
```

Create a single large `sda1` that fills the whole drive and leave it as the default type 'Linux' (83).

Read the article on **Partitioning** and then **Dm-crypt/Device encryption**. Follow the good advice and run:

```
# cryptsetup benchmark (to make sure that the list below is populated)
```

Then:

```
# cat /proc/crypto
```

This gives us the crypto options that can be used. It also provides a representation of the best way to set up LUKS. In our case, security is a priority and speed a distant second. To gain a better understanding, also read:

```
# man cryptsetup
```

Following **Dm-crypt/Device encryption**, considering the above requirements, we do the following based on **Encryption options for LUKS mode**. Reading through, it seems like

Serpent (encryption) and Whirlpool (hash) is the best option.

Initialize LUKS with the following command (as of version 2.1 cryptsetup changed the default type to LUKS2 which does not work with libreboot's grub payload. Hence it is necessary to explicitly change the type to LUKS1):

```
# cryptsetup -v --cipher serpent-xts-plain64 --key-size 512 --hash whirlpool --use-random
--verify-passphrase luksFormat --type luks1 /dev/sda1
```

Choose a secure passphrase here. You should ideally use a long series of mixed lowercase and uppercase letters, numbers, symbols, etc. The password length should be as long as you are able to handle without writing it down or storing it anywhere.

5.3 Create LVM

See **LVM**.

Open the LUKS partition:

```
# cryptsetup luksOpen /dev/sda1 lvm
```

(it will be available at `/dev/mapper/lvm`)

Create an LVM partition:

```
# pvcreate /dev/mapper/lvm
```

Show that you just created it:

```
# pvdisplay
```

Create the volume group, inside of which the logical volumes will be created:

```
# vgcreate matrix /dev/mapper/lvm
```

(volume group name is 'matrix' -- choose a different name, if you like) Show that you created it:

```
# vgdisplay
```

Now create the logical volumes:

```
# lvcreate -L 4G matrix -n swapvol (4 GB swap partition, named swapvol)
```

Again, choose a different name if you like. Also, make sure to choose a swap size of your own needs. It basically depends on how much RAM you have installed. See this **this article**.

This creates a single large partition in the remaining space, named rootvol:

```
# lvcreate -l +100%FREE matrix -n rootvol
```

You can also be flexible here and, for example, specify a `/boot`, `/`, `/home`, `/var`, `/usr`, etc. If you will be running a web/mail server then you want `/var` in its own partition (so that if it fills up with logs, it won't crash your system). For a home/laptop system a root and a swap will be just fine (really).

Verify that the logical volumes were created, using the following command:

```
# lvdisplay
```

5.4 Create / and swap partitions, and mount

For the swapvol LV:

```
# mkswap /dev/mapper/matrix-swapvol
```

Activate swap:

```
# swapon /dev/matrix/swapvol
```

For the rootvol LV:

```
# mkfs.ext4 /dev/mapper/matrix-rootvol
```

Mount the root (/) partition:

```
# mount /dev/matrix/rootvol /mnt
```

6 Continue with Parabola installation

Now, let's follow the rest of the **Installation Guide**.

Create `/home` and `/boot` on the rootvol mountpoint:

```
# mkdir -p /mnt/home
# mkdir -p /mnt/boot
```

Once all the remaining partitions, if any, have been mounted, the devices are ready to have Parabola installed. Proceed with **Verification of package signatures** and further with generic Parabola installation.

One special consideration is initramfs generation. Edit the configuration file `/etc/mkinitcpio.conf`. For our specific use case we'll do the following modifications:

```
 MODULES="i915"
```

This forces the Intel graphics driver to load earlier, so that the console font isn't wiped out after getting to login).

```
HOOKS="base udev autodetect modconf block keyboard keymap consolefont encrypt lvm2
filesystems fsck shutdown"
```

`keymap` adds to initramfs the keymap that you specified in /etc/vconsole.conf; `consolefont` the font specified in /etc/vconsole.conf; `encrypt` to unlock your LUKS encrypted disks at boot time; `lvm2` to mount the LVM partitions at boot time; and `shutdown` to properly unmount devices such as LUKS/LVM when you want to power down. Please also note that the order of the above hooks matter.

Now, using mkinitcpio, you can create the kernel and RAM disk for booting the system:

```
# mkinitcpio -p linux-libre
```

6.1 MacBook2,1 users

To get a working keyboard after booting into the newly installed system you will first need to a) add the required kernel modules to the initramfs (including `hid` and `hid-apple`, but

apparently these two are not enough) **or** b) put a keyfile in there.

7 Unmount, reboot!

Exit from chroot:

```
# exit
```

Unmount:

```
# umount -R /mnt
# swapoff -a
```

Deactivate the LVM volumes:

```
# lvchange -an /dev/matrix/rootvol
# lvchange -an /dev/matrix/swapvol
```

Lock the encrypted partition (close it):

```
# cryptsetup luksClose lvm
```

Shut down:

```
# poweroff
```

Remove the installation medium, then boot up again.

8 Booting from GRUB

Initially you will have to boot manually. Press 'c' to open the GRUB command line.

```
grub> cryptomount -a
grub> insmod ext2
grub> insmod lvm
grub> set root='lvm/matrix-rootvol'
grub> linux /boot/vmlinuz-linux-libre root=/dev/matrix/rootvol cryptdevice=/dev/sda1:root
grub> initrd /boot/initramfs-linux-libre.img
grub> boot
```

The `-a` option above instructs GRUB to decrypt all LUKS volumes that it can find. If you know the exact reference to your device, e.g., `ahci0,msdos1`, then use that instead (i.e. `cryptomount ahci0,msdos1`). You could also make it load `linux-libre-lts` or `linux-libre-hardened` by appending `-lts` or `-hardened` respectively to `vmlinuz-linux-libre` **and** `initramfs-linux-libre` above.

9 Follow-up tutorial: configuring Parabola

We will modify `grub.cfg` inside the ROM and do all kinds of fun stuff, but you first might want to transform the current bare-bones Parabola install into a more useable system. Doing so will make the upcoming ROM modifications easier to perform and less risky. See **Leah Rowe's post-installation notes**. You can also cherry pick useful notes and come up with your own system. Parabola is user-centric, which means that you are in control.

10 Modify grub.cfg inside the ROM

To boot Parabola without first having to type in a bunch of commands, you need to modify the GRUB menu that exists inside the flash chip. **This page** shows you how. Follow that

guide while using the configuration details below. If you go for option 2 (re-flash), then make sure to perform this on `grubtest.cfg` first! We can't emphasise this enough. This is to reduce the possibility of bricking your device!

We'll explain the re-flash option here.

First you need to install flashrom:

```
# pacman -S flashrom
```

Now, go to the `libreboot_util/cbfstool/{armv7l i686 x86_64}` directory. Dump the current firmware as below (`libreboot.rom` is just an example name).

```
# flashrom -p internal -r libreboot.rom
```

If flashrom complains about multiple flash chips detected, add the `-c` option at the end, with the name of your chosen chip within double quotes.

You can check if everything is in there (`grub.cfg` and `grubtest.cfg` would be really nice):

```
$ ./cbfstool libreboot.rom print
```

Extract `grubtest.cfg`:

```
$ ./cbfstool libreboot.rom extract -n grubtest.cfg -f grubtest.cfg
```

Edit `grubtest.cfg`. Inside the 'Load Operating System' menu entry, change the contents to something like this:

```
cryptomount -a
set root='lvm/matrix-rootvol'
linux /boot/vmlinuz-linux-libre root=/dev/matrix/rootvol cryptdevice=/dev/sda1:root
initrd /boot/initramfs-linux-libre.img
```

Again, without specifying a device, the (`-a` option tries to unlock all detected LUKS volumes, so if you know the exact reference to your device, e.g., `ahci0,msdos1`, then use that instead (or, alternatively, `-u` UUID).

To boot the latest LTS or Hardened kernel instead, replace the two instances of `linux-libre` with `linux-libre-lts` or `linux-libre-hardened` above. You could also copy the menu entry and have, for example, one default `linux-libre-hardened` option named "Load Parabola GNU/Linux-libre (Hardened)", one `linux-libre` ("Load Parabola GNU/Linux-libre (latest stable)") and one `linux-libre-lts` (Load Parabola GNU/Linux-libre (LTS)"). The first entry will load by default.

Now, to protect your system from an attacker simply booting a live USB distro and re-flashing the boot firmware, we are going to add a password for GRUB.

Connect to the Internet, e.g. by starting `dhcp` on ethernet:

```
# systemctl start dhcpcd.service
```

AGAIN, MAKE SURE TO DO THIS WHOLE SECTION ON `grubtest.cfg` *BEFORE* DOING IT ON `grub.cfg`. (When we get there, upon reboot, select the menu entry that says "Switch to grubtest.cfg" and test that it works. Only once you are satisfied, copy that to `grub.cfg`. Only a few steps to go, though.) WHY? BECAUSE AN INCORRECTLY SET PASSWORD CONFIG MEANS YOU CAN'T AUTHENTICATE, WHICH MEANS 'BRICK'.

We need a utility that comes with GRUB, so we will download it temporarily. (Remember that GRUB doesn't need to be installed on the storage device, since it's already included in the boot flash.) Also, we will use flashrom, and (if you want) dmidecode. You only need base-devel (compilers and so on) to build and use cbfstool. It was already installed if you followed this tutorial, but here it is:

```
# pacman -S grub flashrom dmidecode base-devel
```

Next, run this command:

```
# grub-mkpasswd-pbkdf2
```

Enter your chosen password at the prompt and your hash will be shown. Copy this string -- you will add it to your `grubtest.cfg`.

The password below (it's password, by the way) after 'password_pbkdf2 root' should be changed to your own. Make sure to specify a password that is different from both your LUKS *and* your root/user password. Obviously, do not simply copy and paste the examples shown here...

Next, back in `grubtest.cfg`, above the first 'Load Operating System' menu entry, you should now add your GRUB password as shown below. Replace 'root' with your own name (if you want) and insert the password hash which you copied:

```
set superusers="root"
password_pbkdf2 root
grub.pbkdf2.sha512.10000.711F186347156BC105CD83A2ED7AF1EB971AA2B1EB2640172F34B0DEFFC97E654AF
```

To read more about GRUB security, see **Authentication and authorisation** in the GRUB manual.

Save your changes in `grubtest.cfg`, then delete the unmodified config from the ROM image:

```
$ ./cbfstool libreboot.rom remove -n grubtest.cfg
```

and insert the modified `grubtest.cfg`:

```
$ ./cbfstool libreboot.rom add -n grubtest.cfg -f grubtest.cfg -t raw
```

Now, read **How to update or install libreboot (if you are already running libreboot or coreboot)**. Go (up) to the `libreboot_util` directory and update the flash chip contents:

```
# ./flash update libreboot.rom
```

Occasionally, coreboot changes the name of a given board. If flashrom complains about a board mismatch, but you are sure that you chose the correct ROM image, then run this alternative command:

```
# ./flash forceupdate libreboot.rom
```

You should see "Verifying flash... VERIFIED." written at the end of the flashrom output.

With this new configuration, Parabola can boot automatically but you will need to enter a password at boot time, in GRUB, before being able to use any of the menu entries or switch to the terminal. Let's test it out: reboot and select grubtest.cfg from the GRUB menu by using the arrow keys on your keyboard. Enter the name you chose, the GRUB password, your

LUKS passphrase and then log in as root or your user. All went well? Great!

If it does not work like you want it to, if you are unsure or sceptical in any way, don't despair; you have been wise and did not brick your device! Reboot and log back in the default way, and then modify your grubtest.cfg again until you get it right! Do *not* proceed past this point unless you are absolutely certain that your new configuration is safe (or desirable) to use.

Now, we can easily and safely create a copy of `grubtest.cfg`, called `grub.cfg`. This will be the same except for one difference: the menu entry 'Switch to grub.cfg' is changed to 'Switch to grubtest.cfg' and, inside it, all instances of grub.cfg to grubtest.cfg. This is so that the main config still links (in the menu) to grubtest.cfg, so that you don't have to manually switch to it, in case you ever want to follow this guide again in the future (modifying the already modified config). Inside `libreboot_util/cbfstool/{armv7l i686 x86_64}`, we can do this with the following command:

```
$ sed -e 's:(cbfsdisk)/grub.cfg:(cbfsdisk)/grubtest.cfg:g' -e 's:Switch to
grub.cfg:Switch to grubtest.cfg:g' < grubtest.cfg > grub.cfg
```

Delete the grub.cfg that remained inside the ROM:

```
$ ./cbfstool libreboot.rom remove -n grub.cfg
```

Add the modified version that you just made:

```
$ ./cbfstool libreboot.rom add -n grub.cfg -f grub.cfg -t raw
```

Now you have a modified ROM. Once more, read **How to update or install libreboot (if you are already running libreboot or coreboot)**, go to the `libreboot_util` directory and update the flash chip contents:

```
# ./flash update libreboot.rom
```

...and wait for "Verifying flash... VERIFIED." Once you have done that, shut down and then boot up with your new configuration.

When you are done, delete GRUB (remember, we only needed it for the grub-mkpasswd-pbkdf2 utility; GRUB is already part of libreboot, flashed alongside it as a payload):

```
# pacman -R grub
```

If you followed all of the above correctly, you should now have a fully encrypted Parabola installation.

11 Bonus: Using a key file to unlock /boot/

By default, you will have to enter your LUKS passphrase twice; once in GRUB, and once when booting the kernel. GRUB unlocks the encrypted partition and then loads the kernel, but the kernel is not aware of the fact that it is being loaded from an encrypted volume. Therefore, you will be asked to enter your passphrase a second time. A workaround is to put a key file inside initramfs, with instructions for the kernel to use it when booting. This is safe, because `/boot/` is encrypted (otherwise, putting a key file inside initramfs would be a bad idea).

Boot up and log in as root or your user. Then generate the key file:

```
# dd bs=512 count=4 if=/dev/urandom of=/etc/mykeyfile iflag=fullblock
```

Insert it into the LUKS volume:

```
# cryptsetup luksAddKey /dev/sdX /etc/mykeyfile
```

…and enter your LUKS passphrase when prompted. Edit `/etc/mkinitcpio.conf` to include the key in the FILES array, e.g.

```
FILES="/etc/mykeyfile"
```

Create the initramfs image from scratch:

```
# mkinitcpio -p linux-libre
# mkinitcpio -p linux-libre-lts
# mkinitcpio -p linux-libre-hardened
```

Add the following to the kernel (linux) line in your grub.cfg -- which you now know to do, see above! --, e.g. after root=/dev/matrix/rootvol if you've followed this guide:

```
# cryptkey=rootfs:/etc/mykeyfile
```

Finally, deny read access to the key file, even by root:

```
# chmod 000 /etc/mykeyfile
```

12 Further security tips

- **Security**
- **Xtreme**
- **arch:Security**
- **User:GNUtoo/laptop**

13 Troubleshooting

A libreboot user reported issues when booting with a docking station attached on an X200, when decrypting the disk in GRUB. The error AHCI transfer timed out was observed. The workaround was to remove the docking station.

Further investigation revealed that it was the DVD drive that was causing problems. Removing it worked around the issue.

```
"sudo wodim -prcap" shows information about the drive:
Device was not specified. Trying to find an appropriate drive...
Detected CD-R drive: /dev/sr0
Using /dev/cdrom of unknown capabilities
Device type    : Removable CD-ROM
Version        : 5
Response Format: 2
Capabilities   :
Vendor_info    : 'HL-DT-ST'
Identification : 'DVDRAM GU10N    '
Revision       : 'MX05'
Device seems to be: Generic mmc2 DVD-R/DVD-RW.
```

Drive capabilities, per MMC-3 page 2A:

```
 Does read CD-R media
 Does write CD-R media
 Does read CD-RW media
 Does write CD-RW media
 Does read DVD-ROM media
 Does read DVD-R media
 Does write DVD-R media
 Does read DVD-RAM media
 Does write DVD-RAM media
```

```
Does support test writing
```

```
Does read Mode 2 Form 1 blocks
Does read Mode 2 Form 2 blocks
Does read digital audio blocks
Does restart non-streamed digital audio reads accurately
Does support Buffer-Underrun-Free recording
Does read multi-session CDs
Does read fixed-packet CD media using Method 2
Does not read CD bar code
Does not read R-W subcode information
Does read raw P-W subcode data from lead in
Does return CD media catalog number
Does return CD ISRC information
Does support C2 error pointers
Does not deliver composite A/V data
```

```
Does play audio CDs
Number of volume control levels: 256
Does support individual volume control setting for each channel
Does support independent mute setting for each channel
Does not support digital output on port 1
Does not support digital output on port 2
```

```
Loading mechanism type: tray
Does support ejection of CD via START/STOP command
Does not lock media on power up via prevent jumper
Does allow media to be locked in the drive via PREVENT/ALLOW command
Is not currently in a media-locked state
Does not support changing side of disk
Does not have load-empty-slot-in-changer feature
Does not support Individual Disk Present feature
```

```
Maximum read  speed:  4234 kB/s (CD  24x, DVD  3x)
Current read  speed:  4234 kB/s (CD  24x, DVD  3x)
Maximum write speed:  4234 kB/s (CD  24x, DVD  3x)
Current write speed:  4234 kB/s (CD  24x, DVD  3x)
Rotational control selected: CLV/PCAV
Buffer size in KB: 1024
Copy management revision supported: 1
Number of supported write speeds: 4
Write speed # 0:  4234 kB/s CLV/PCAV (CD  24x, DVD  3x)
Write speed # 1:  2822 kB/s CLV/PCAV (CD  16x, DVD  2x)
Write speed # 2:  1764 kB/s CLV/PCAV (CD  10x, DVD  1x)
Write speed # 3:   706 kB/s CLV/PCAV (CD   4x, DVD  0x)
```

Supported CD-RW media types according to MMC-4 feature 0x37:

```
Does write multi speed       CD-RW media
Does write high  speed       CD-RW media
Does write ultra high speed  CD-RW media
Does not write ultra high speed+ CD-RW media
```

14 Warranty disclaimer

UNLESS OTHERWISE SEPARATELY UNDERTAKEN BY THE LICENSOR, TO THE EXTENT POSSIBLE, THE LICENSOR OFFERS THE LICENSED MATERIAL AS-IS AND AS-AVAILABLE, AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE LICENSED MATERIAL, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHER. THIS INCLUDES, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OR ABSENCE OF ERRORS, WHETHER OR NOT KNOWN OR DISCOVERABLE. WHERE DISCLAIMERS OF WARRANTIES ARE NOT ALLOWED IN FULL OR IN PART, THIS DISCLAIMER MAY NOT APPLY TO YOU.

TO THE EXTENT POSSIBLE, IN NO EVENT WILL THE LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE) OR OTHERWISE FOR ANY DIRECT, SPECIAL, INDIRECT, INCIDENTAL, CONSEQUENTIAL,

PUNITIVE, EXEMPLARY, OR OTHER LOSSES, COSTS, EXPENSES, OR DAMAGES ARISING OUT OF THIS PUBLIC LICENSE OR USE OF THE LICENSED MATERIAL, EVEN IF THE LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES, COSTS, EXPENSES, OR DAMAGES. WHERE A LIMITATION OF LIABILITY IS NOT ALLOWED IN FULL OR IN PART, THIS LIMITATION MAY NOT APPLY TO YOU.

The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Categories: Installation | Software | Security