GUIX SYSTEM AND LIBREBOOT

Raghav Gururajan
GUIX SYSTEM AND LIBREBOOT

Guix System with Full Disk Encryption on Libreboot

Raghav Gururajan
INTRODUCTION

Guix System is an exotic distribution of GNU+Linux operating system; with Guix as package+system manager, Linux-Libre as kernel and Shepherd as init system.

Libreboot is a de-blobbed distribution of Coreboot firmware. By default, Libreboot comes with GRUB bootloader as a payload.

The objective of this manual is to provide step-by-step guide for setting up Guix System (stand-alone Guix), with Full Disk Encryption (FDE), on devices powered by Libreboot.

Any users, for their generalized use cases, need not stumble away from this guide to accomplish the setup. Advancers, for deviant use cases, will have to explore outside this guide for customization; although this guide provides information that is of paramount use.

Let us begin!
PREPARATION

In the current GNU+Linux system, open terminal as root user.

Insert USB drive and get the device letter /dev/sdX, where “X” is the device letter.

```
lsblk --list
```

Unmount the device just in case if it is auto-mounted.

```
umount /dev/sdX --verbose
```

Download the Guix System ISO installer package and it’s GPG signature; where “a.b.c” is the version number and “sss” is the system architecture.

```
wget --verbose
```

```
wget --verbose
```

Import the Guix’s public key.

```
gpg --verbose --keyserver pool.sks-keyservers.net --receive-keys
3CE464558A84FDC69DB40CFB090B11993D9AEBB5
```

Verify the GPG signature of the downloaded package.

```
gpg --verbose --verify guix-system-install-a.b.c.sss-linux.iso.xz.sig
```

Extract ISO image from the downloaded package.
xz --verbose --decompress guix-system-install-a.b.c.sss-linux.iso.xz

Write the extracted ISO image to the drive.

dd if=guix-system-install-a.b.c.sss-linux.iso of=/dev/sdX status=progress; sync

Reboot the device.

reboot
On reboot, as soon as the Libreboot's graphic art appears, press "S" or choose “Search for GRUB2 configuration on external media [s]”. Wait for the Guix System from USB drive to load.

Once Guix System installer starts, choose "Install using the shell based process".

Set your keyboard layout, where “lo” is the two-letter keyboard layout code (lower-case).

```bash
loadkeys --verbose lo
```

Unblock network interfaces.

```bash
rfkill unblock all
```

Get the names of network interfaces.

```bash
ifconfig -v -a
```

Bring the desired network interface (wired or wireless) up, where “nwif” is the network interface name.

```bash
ifconfig -v nwif up
```

For wireless connection, follow the wireless setup.

```bash
*** Start of Wireless Setup ***
```

Create a configuration file using text editor, where “fname” is any desired name for file.

```bash
nano fname.conf
```

Choose, type and save ONE of the following snippets, where ‘net’ is the network name, ‘pass’ is the password or passphrase and ‘uid’ is the user identity.
For most private networks:

```plaintext
network={
    ssid="net"
    key_mgmt=WPA-PSK
    psk="pass"
}
```

(or)

For most public networks:

```plaintext
network={
    ssid="net"
    key_mgmt=NONE
}
```

(or)

For most organizational networks:

```plaintext
network={
    ssid="net"
    scan_ssid=1
    key_mgmt=WPA-EAP
    identity="uid"
    password="pass"
    eap=PEAP
    phase1="peaplabel=0"
    phase2="auth=MSCHAPV2"
}
```

Connect to the configured network.

```plaintext
wpa_supplicant -B -c fname.conf -i nwif
```

*** End of Wireless Setup ***
Assign an IP address to the network interface.

```
dhclient -v nwif
```

Obtain the device letter `/dev/sdX` in which you would like to deploy and install Guix System, where “X” is the device letter.

```
lsblk --list
```

Wipe the device (Ignore if the device is new).

```
shred --verbose --random-source=/dev/urandom /dev/sdX
```

Load the device-mapper module in the current kernel.

```
modprobe --verbose dm_mod
```

Partition the device. Follow the prompts. Just do, GPT -> New -> Write -> Quit; defaults will be set.

```
cfdisk /dev/sdX
```

Obtain the partition number from the device, where “Y” is the partition number.

```
lsblk --list
```

Encrypt the partition. Follow the prompts.

```
cryptsetup --verbose --hash whirlpool --cipher serpent-xts-plain64 --verify-passphrase --use-random --key-size 512 --iter-time 500 luksFormat /dev/sdXY
```

Obtain and note down the UUID of the LUKS partition.

```
cryptsetup --verbose luksUUID /dev/sdXY
```
Open the encrypted partition, where “luks-uuid” is the LUKS UUID and “partname” is any desired name for partition.

cryptsetup --verbose luksOpen UUID=luks-uuid partname

Create a physical volume in the partition.

pvcreate /dev/mapper/partname --verbose

Create a volume group in the physical volume, where “vgname” is any desired name for volume group.

vgcreate vgname /dev/mapper/partname --verbose

Create logical volumes in the volume group; where “num” is the number for space in GB, and “lvnameroot” and “lvnamehome” are any desired names for root and home volumes respectively.

lvcreate --extents 25%VG vgname --name lvnameroot --verbose

lvcreate --extents 100%FREE vgname --name lvnamehome --verbose

Create filesystems on the logical-volumes, where “fsnameroot” and “fsnamehome” are any desired names for root and home filesystems respectively.

mkfs.btrfs --metadata dup --label fsnameroot /dev/vgname/lvnameroot

mkfs.btrfs --metadata dup --label fsnamehome /dev/vgname/lvnamehome

Mount the filesystems under the current system.

mount --label fsnameroot --target /mnt --types btrfs --verbose
mkdir --verbose /mnt/home && mount --label fsnamehome --target /mnt/home --types btrfs --verbose

Create a swap file.

dd bs=1MiB count=1GiB if=/dev/zero of=/mnt/swapfile status=progress

mkswap --verbose /mnt/swapfile

Make the swap file readable and writable only by root account.

chmod --verbose 600 /mnt/swapfile

Activate the swap file.

swapon --verbose /mnt/swapfile

Make the installation packages to be written on the mounted root filesystem.

herd start cow-store /mnt

Create the system-wide configuration files directory.

mkdir --verbose /mnt/etc

Create, edit and save the system configuration file by typing the following code snippet. WATCH-OUT for variables in the code snippet and replace them with the relevant values.

nano /mnt/etc/config.scm

Snippet:

(use-modules
(gnu)
(gnu system nss))
(use-package-modules
certs
gnome
linux)
(use-service-modules
desktop
xorg)
(operating-system
(kernel linux-libre-lts)
(kernel-arguments
(append
(list
  "iomem=relaxed")
%default-kernel-arguments))
(bootloader
(bootloader-configuration
(bootloader
 (inherit grub-bootloader)
 (installer #~(const #t)))
 (keyboard-layout keyboard-layout)))
(keyboard-layout
(keyboard-layout
 "xy"
 "altgr-intl")
(host-name "hostname")
(mapped-devices
(list
 (mapped-device
 (source
 (uuid "luks-uuid"))
 (target "partname")
 (type luks-device-mapping))
(mapped-device
 (source "vgname")
 (targets
 (list
  "vgname-lvnameroot"
  "vgname-lvnamehome")
 (type lvm-device-mapping)))
(file-systems
(append
(list
 (file-system
 (type "btrfs")
 (mount-point "/")
 (device "/dev/mapper/vgname-lvnameroot")

Initialize new Guix System.

```
guix system init /mnt/etc/config.scm /mnt
```

Reboot the device.
reboot
COMPLETION

On reboot, as soon as the Libreboot graphic art appears, press “C” to enter the command-line.

Enter the following commands and respond to first command with the LUKS Key.

```
cryptomount -u luks-uuid
```

```
set root=(lvm/vgname-lvnameroot)
```

Upon Guix’s GRUB menu, go with the default option.

Enter the LUKS Key again, for kernel, as prompted.

Upon login screen, login as "root" with password field empty.

Open terminal.

Set passkey for the "root" user. Follow the prompts.

```
passwd root
```

Set passkey for the “username” user. Follow the prompts.

```
passwd username
```

Install flashrom and wget.

```
guix package --install flashrom wget
```

Obtain the ROM chip’s model and size. Look for the output line “Found [...] flash chip [...]”.

```
flashrom --verbose --programmer internal
```
Download Libreboot ROM and utilities, where "YYYYMMDD" is the release date, "devmod" is the device model and "N" is the ROM chip size.

```
```

```
wget --verbose https://rsync.libreboot.org/stable/YYYYMMDD/libreboot_rYYYYMMDD_util.tar.xz
```

Extract the downloaded files.

```
tar --extract --file=libreboot_rYYYYMMDD_grub_devmod_Nmb.tar.xz --verbose
```

```
tar --extract --file=libreboot_rYYYYMMDD_util.tar.xz --verbose
```

Rename the directories of extracted files.

```
mv --verbose "libreboot_rYYYYMMDD_grub_devmod_Nmb.tar.xz" "libreboot_rom"
```

```
mv --verbose "libreboot_rYYYYMMDD_util" "libreboot_util"
```

Copy the ROM image to the directory of cbfstool, where "kbdlo" is the keyboard layout and "arch" is the system architecture.

```
cp libreboot_rom/devmod_Nmb_kbdlo_vesafb.rom libreboot_util/cbfstool/arch/libreboot.rom
```

Change directory to the directory of cbfstool.

```
CD libreboot_util/cbfstool/arch/
```

Extract the GRUB configuration file from the image.
./cbfstool libreboot.rom extract -n grub.cfg -f grub.cfg

Edit the GRUB configuration file and insert the following code snippet above the line "menuentry 'Load Operating System [o]' --hotkey='o' --unrestricted { [...]}".

nano grub.cfg

Snippet:

```
menuentry 'Guix System (An advanced distribution of the GNU operating system) [g]' --hotkey='g' --unrestricted
{
cryptomount -u luks-uuid
set root=(lvm/vgname-lvnameroot)
configfile /boot/grub/grub.cfg
}
```

Remove the old GRUB configuration file from the ROM image.

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

Insert the new GRUB configuration file into the ROM image.

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

Move the ROM image to the directory of ich9gen.

mv libreboot.rom ~/libreboot_util/ich9deblob/arch/libreboot.rom

Change directory to the directory of ich9gen.

cd ~/libreboot_util/ich9deblob/arch/

Generate descriptor+GbE images with the MAC address, where "mac-addr" is the MAC address of the machine.
ich9gen --macaddress mac-addr

Insert the descriptor+GbE image into the ROM image, where "N" is the ROM chip size.

dd bs=12k conv=notrunc count=1 if=ich9fdgbe_Nm.bin of=libreboot.rom status=progress

Move the ROM image to the directory of flash.

mv libreboot.rom ~/libreboot_util/libreboot.rom

Change directory to the directory of flash.

cd ~/libreboot_util

Modify the shebang of flash script, from `#!/bin/bash` to `#!/bin/sh`.

nano flash

Flash the ROM with the new image.

./flash update libreboot.rom

(or)

./flash forceupdate libreboot.rom

Reboot the device.

reboot
CONCLUSION

Everything should be stream-lined from now. Upon Libreboot’s GRUB menu, you can either press “G” or choose “Guix System (An advanced distribution of the GNU operating system) [g]”.

During the boot process, as prompted, you have to type LUKS key twice; once for Libreboot’s GRUB and once more for Linux-Libre kernel.

Generally, you will be using Libreboot’s initial/default grub.cfg, whose Guix menu-entry invokes Guix’s grub.cfg located at "/boot/grub/". For trouble-shooting, you can also use Libreboot’s grubtest.cfg, which hasn’t been modified.

That is it! You have now setup Guix System with Full Disk Encryption on your device powered by Libreboot. Enjoy!
REFERENCES


ACKNOWLEDGEMENTS

[1] Thanks to Guix developer, Clement Lassieur (clement@lassieur.org), for helping me with the Scheme code for the bootloader configuration.

[2] Thanks to Libreboot founder and developer, Leah Rowe (leah@libreboot.org), for helping me with the understanding of Libreboot’s functionalities.
LICENSE

This work by Raghav Gururajan is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.

To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/