



VOLLSTÄNDIGE FESTPLATTENVERSCHLÜSSEL UNG UNTER MANJARO LINUX

Posted on 25. November 2023 by Dirk Wouters



Categories: [Computer](#), [Linux](#)

Tags: [btrfs](#), [grub](#), [luks](#), [luks2](#), [manjaro](#),
[verschlüsselung](#)

[Manjaro](#) Linux bringt mit Calamares einen hervorragenden Installer mit, der aber leider nur eingeschränkte Möglichkeiten bezüglich des Partitionenlayout bietet. Eine komplette Verschlüsselung aller Linux Partitionen, inkl. /boot und dann noch mit [LUKS2](#) ist leider nicht möglich. Auch aufwendigere Partionenlayouts unter BTRFS sind nur mit umwegen mit Calamares möglich. Es gibt zwei Möglichkeiten, entweder geht man den "Arch Linux Weg" und installiert Manjaro komplett manuell, oder man migriert eine vorhandene Installation in ein verschlüsseltes System. Ich beschreibe hier die zweite, etwas einfache Möglichkeit.

Inhalt

- [1. Einleitung](#)
- [2. Ausgangssituation](#)
- [3. Vorbereitungen](#)
 - [3.1 Benötigte Pakete installieren](#)
 - [3.2 Anlegen der Partitionen](#)
 - [3.3 Partition verschlüsseln](#)
 - [3.4 BTRFS Dateisystem erstellen](#)
 - [3.5 BTRFS Subvolumes anlegen](#)
 - [3.6 Subvolumes einhängen](#)
 - [3.7 SWAP Datei anlegen und aktivieren](#)
 - [3.8 FSTAB generieren](#)
 - [3.9 Neuen User anlegen und anmelden](#)
- [4. Übertragen der Daten](#)
- [5. Konfiguration des neuen Systems](#)
 - [5.1 Wurzelverzeichnis virtuell ändern](#)
 - [5.2 Schlüsseldatei anlegen](#)
 - [5.3 Neue initramfs erzeugen](#)
 - [5.4 Modifizierter GRUB installieren](#)
 - [5.4.1 Vorhandenes GRUB Paket deinstallieren](#)
 - [5.4.2 Modifiziertes GRUB Paket installieren](#)
 - [5.4.3 GRUB konfigurieren](#)
 - [5.4.4 GRUB Bootloader installieren](#)
 - [5.5 Rechner neu starten](#)
- [6. Abschließende Maßnahmen](#)
 - [6.1 Temporären Benutzer wieder löschen](#)

- [6.2 LUKS Passwort ändern](#)
 - [6.3 LUKS Header sichern](#)
 - [6.3.1 LUKS Header Backup zurück schreiben](#)
 - [6.4 Recovery Keys generieren](#)
 - [6.4.1 Eigenes Passwort nutzen](#)
 - [6.4.2 Automatisch Recovery-Key generieren lassen](#)
 - [6.5 Alte Manjaro Partition löschen](#)
 - [7. Fazit](#)
 - [Quellen:](#)
-

1. Einleitung

In der heutigen digitalen Ära, in der die Datenintegrität und der Schutz der Privatsphäre von höchster Bedeutung sind, ist die Sicherheit des Betriebssystems ein unverzichtbarer Aspekt für jeden Computerbenutzer. Manjaro Linux, eine auf Arch Linux basierende Distribution, bietet nicht nur eine benutzerfreundliche und leistungsstarke Plattform, sondern auch eine Vielzahl von Sicherheitsfunktionen, darunter die Luks-Verschlüsselung.

Luks (Linux Unified Key Setup) ist ein robustes Verschlüsselungssystem, das auf Linux-Plattformen weit verbreitet ist und eine transparente, benutzerfreundliche Möglichkeit bietet, Daten auf einem System zu verschlüsseln. Diese Technologie ermöglicht es Benutzern, ihre gesamte Festplatte oder bestimmte Partitionen zu verschlüsseln, was eine zusätzliche Sicherheitsebene bietet, die den Zugriff auf sensible Daten selbst im Falle eines physischen Diebstahls oder unbefugten Zugriffs auf das System verhindert.

Möchte man ein höchstes Maß an Sicherheit, sollte das aktuelle LUKS2 eingesetzt werden, mit entsprechend sicheren Verschlüsselungsmechanismen. Zusätzlich wird ein möglichst sicheres Passwort benötigt. Wie man sich recht einfach ein solches selber erzeugen kann, habe ich unter [Sichere und einfache Passwörter einfach würfeln](#) beschrieben.

2. Ausgangssituation

- Ein Rechner mit einer oder mehreren Festplatten, bzw. SSD Laufwerken.
- Mindestens Manjaro ist schon installiert und entsprechend eingerichtet, weitere Betriebssysteme, wie zum Beispiel Windows sind kein Problem
- Ausreichend Platz für eine weitere Installation oder ein zusätzliches leeres Laufwerk

Sofern Manjaro noch nicht installiert sein sollte, kann dieses irgendwo auf einer freien kleinen Partition mit Hilfe der [Live CD](#) eingerichtet werden. Die weiteren Arbeiten, werden aus dieser Installation, bzw. aus der schon vorher bestehenden Partition ausgeführt. Es wird also davon ausgegangen, dass das Manjaro Linux gebootet ist.

3. Vorbereitungen

3.1 Benötigte Pakete installieren

Sofern noch nicht vorhanden, müssen einige Pakete installiert werden:

```
$ pamac install base-devel git patch gdisk gparted yay
```

Die restlichen benötigten Pakete sollten schon durch die Standard Manjaro Installation vorhanden sein.

Im weiteren Verlauf wird ein Konsolenfenster, bzw. Terminal mit Root Rechten benötigt:

```
$ sudo su -
```

3.2 Anlegen der Partitionen

In meinem Beispiel habe ich ein NVMe SSD Laufwerk unter `/dev/nvme0`, welches schon folgende Partitionen enthält:

- `/dev/nvme0n1p1` --> `/efi`
- `/dev/nvme0n1p1` --> `/`

Diese Partitionen wurden durch den Manjaro Installer installiert. Während `/efi` im FAT32 Format vorliegt, ist die Root Partition in `btrfs` formatiert. Es kann aber auch eine EXT4 Partition oder sonstiges sein, das spielt keine Rolle. Auf der Festplatte selber ist noch genug Platz, so dass ich eine 3. Partition anlegen kann. Dazu wird das Partitionierungstool `gdisk` aufgerufen mit Angabe der entsprechenden Festplatte:

```
# gdisk /dev/nvme0n1
```

Dort erstellt man mit folgender Tastenfolge eine weitere Partition:

```
n
```

```
8309
```

```
W
```

```
Y
```

Die Eingaben sind dabei wie folgt:

- `n` --> Neue Partition anlegen
- --> Es wird die nächst freie Partitionnummer (hier dann die 3) vorgeschlagen, die übernommen werden kann
- --> Start der neuen Partition, welcher ebenfalls vorgeschlagen wird

- --> Ende der neuen Partition. Hier ist der Rest des Laufwerks vorgegeben. Falls nicht gewünscht, kann zum Beispiel mit +300G, anstatt eine Partition mit der Größe von 300 Gigabyte angelegt werden und der Rest bleibt frei.
- W --> Schreiben des Partitionslayout
- Y --> Änderungen, bzw. das Schreiben bestätigen

3.3 Partition verschlüsseln

Die freigemachte Partition `/dev/nvme0n1p3` wird nun mit folgenden Parametern Verschlüsselt:

- Verschlüsselungs-Typ: `aes-xts-plain64`
- Schlüssellänge: 512 Byte
- Streuwertfunktion: `whirlpool`
- Luksversion: 2
- Passwortbasierte Schlüsselableitungs-Funktion: `argon2id`

Sofern ein sicheres Passwort verwendet wird, sollte dies für die nächsten Jahrtausende eigentlich reichen, wenn man von heutiger Rechenleistung ausgeht:

```
# cryptsetup luksFormat --use-random --cipher=aes-xts-plain64 --key-size=512  
--hash=whirlpool --type=luks2 --pbkdf=argon2id /dev/nvme0n1p3
```

Es wird im Anschluss nach einem Passwort gefragt. Ich empfehle während der Installation erst einmal ein einfaches Kennwort zu verwenden. Am Schluss wird dann das Kennwort auf eine sichere Variante geändert. Natürlich kann an dieser Stelle auch schon das gewünschte Kennwort eingegeben werden.

Danach wird die verschlüsselte Partition geöffnet, damit mit dieser weiter gearbeitet werden kann:

```
# cryptsetup luksOpen /dev/nvme0n1p3 cryptbtrfs
```

3.4 BTRFS Dateisystem erstellen

Da mit BTRFS Subvolumes gearbeitet wird, werden keine weiteren Partitionen benötigt. BTRFS bringt seinen eigenen "Volumemanager" mit. Daher kann die gesamte Partition, die eben angelegt worden ist, mit dem BTRFS Dateisystem formatiert werden. Da sich das Dateisystem innerhalb des verschlüsselten LUKS2 Containers angelegt wird, darf nicht das Laufwerk selber (also `/dev/nvme1p3`) angegeben werden, sondern die verschlüsselte Partition, die unter `/dev/mapper/cryptbtrfs` zu finden ist:

```
# mkfs.btrfs -L manjaro /dev/mapper/cryptbtrfs
```

Und nun das Laufwerk noch unterhalb von `/mnt` einhängen:

```
# mount /dev/mapper/cryptbtrfs /mnt
```

3.5 BTRFS Subvolumes anlegen

Ich habe mich für folgendes Layout entschieden, welches natürlich den eigenen Vorlieben angepasst werden kann:

- `/@` --> `/`, bzw. Root Verzeichnis
- `/@swap` --> `/swap`, Swap wird als Datei angelegt, welcher somit automatisch auch verschlüsselt ist
- `/@opt` --> `/opt` Verzeichnis

- /@home --> /home Verzeichnis
- @log --> /var/log Verzeichnis
- @cache --> /var/cache Verzeichnis
- @snapshots --> /.snapshots

Für folgende Verzeichnisse, in denen sehr viel geschrieben und geändert wird, wird das Copy-on-Write (CoW) noch deaktiviert:

- /swap
- /var/log
- /var/cache

Als erstes die Subvolumes anlegen:

```
# btrfs subvolume create /mnt/@
# btrfs subvolume create /mnt/@swap
# btrfs subvolume create /mnt/@opt
# btrfs subvolume create /mnt/@home
# btrfs subvolume create /mnt/@log
# btrfs subvolume create /mnt/@cache
# btrfs subvolume create /mnt/@snapshots
```

und anschließend das CoW auf den oben genannten Subvolumes deaktivieren und für Swap die Rechte nur für Root setzen:

```
# chattr +C /mnt/@swap
# chattr +C /mnt/@log
# chattr +C /mnt/@cache
# chmod 700 /mnt/@swap
```

Danach das BTRFS Dateisystem wieder aushängen, damit anschließend die einzelnen Subvolumes gemountet werden können:

```
# umount /mnt
```

3.6 Subvolumes einhängen

Um sich ein wenig Tipparbeit zu sparen, wird als erste die variable `mount_opt` entsprechend gesetzt mit den notwendigen Mount Parametern:

```
# export mount_opt="defaults,noatime,compress=zstd:1"
```

Danach wird als erstes das Root Subvolume eingehangen:

```
# mount -o ${mount_opt},subvol=@ /dev/mapper/cryptbtrfs /mnt
```

Damit die weiteren Subvolumes eingehangen werden können, müssen zuerst die entsprechenden Verzeichnisse im Root Subvolume angelegt:

```
# mkdir -p /mnt/{swap,opt,home,.snapshots,var/cache,var/log}
```

Und nun können die restlichen Subvolumes gemountet werden:

```
# mount -o ${mount_opt},subvol=@swap /dev/mapper/cryptbtrfs /mnt/swap
# mount -o ${mount_opt},subvol=@opt /dev/mapper/cryptbtrfs /mnt/opt
# mount -o ${mount_opt},subvol=@home /dev/mapper/cryptbtrfs /mnt/home
# mount -o ${mount_opt},subvol=@log /dev/mapper/cryptbtrfs /mnt/var/log
# mount -o ${mount_opt},subvol=@cache /dev/mapper/cryptbtrfs /mnt/var/cache
# mount -o ${mount_opt},subvol=@snapshots /dev/mapper/cryptbtrfs
/mnt/.snapshots
```

Und das schon vorhandene `/efi` Verzeichniss wird ebenfalls noch eingehangen:

```
# mount --mkdir /dev/nvme0n1p1 /mnt/efi
```

3.7 SWAP Datei anlegen und aktivieren

Da die Swap Datei auch für das `/tmp` Verzeichnis genutzt werden soll, lege ich mit 32GB eine etwas großzügigere Datei an. Dies kann den eigenen Ansprüchen entsprechend angepasst werden:

```
# btrfs filesystem mkswapfile --size 32G /mnt/swap/swapfile  
# swapon /mnt/swap/swapfile
```

3.8 FSTAB generieren

Da nun alle Dateisysteme eingehangen sind, kann mit dem Befehl `fstabgen` recht einfach die `/etc/fstab` Datei generiert werden:

```
# mkdir /mnt/etc  
# fstabgen -U /mnt >> /mnt/etc/fstab
```

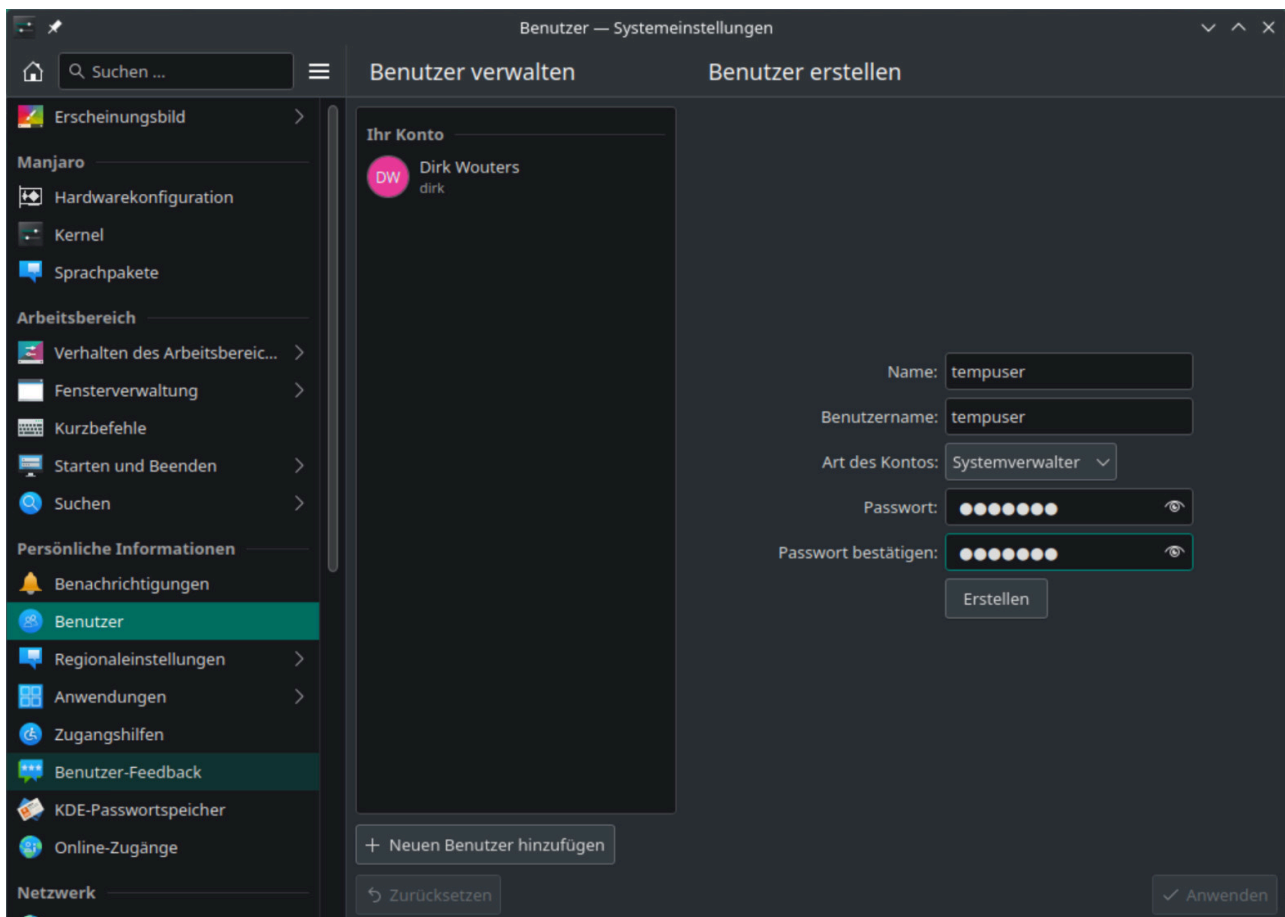
Diese Datei muss nun erst einmal in ein Verzeichnis der bestehenden alten Manjaro Datei weg gesichert werden, da später ja noch alles überschrieben wird. Mit Hilfe dieser Datei kann später die `/etc/fstab` einfacher konfiguriert werden, ohne dass alle Einträge dann manuell erstellt werden müssen:

```
# cp /mnt/etc/fstab /etc/fstab.backup.new
```

Damit wird die Datei in die bestehende Manjaro Installation ins `/etc` Verzeichnis kopiert, welches ja in einem der nächsten Schritte wieder auf die verschlüsselte Partition kopiert wird, zusammen mit allen anderen Daten.

3.9 Neuen User anlegen und anmelden

Damit der bestehende Benutzer keine offenen Dateien hat, empfehle ich einen "Wegwerfaccount" zu erstellen. Also einen Benutzer mit Adminrechten, die nach Abschluß der Maßnahmen wieder gelöscht werden wird. Dies kann einfach über die **Systemeinstellungen** --> **Benutzer** --> **Neuen Benutzer anlegen** erfolgen. Wichtig ist darauf zu achten, dass diesem bei **Art des Kontos** die Systemverwalter Rechte gegeben werden.



Anschließend alle Programme beenden, abmelden und mit dem gerade neu erstellten Benutzer wieder anmelden.

4. Übertragen der Daten

Nun ist es an der Zeit, die Daten auf das verschlüsselte Laufwerk zu übertragen. Dies geht mit dem Befehl `rsync` am einfachsten. Dazu wird wieder ein Terminalfenster geöffnet und mit Root Rechten gearbeitet:

```
$ sudo su -  
# rsync -xaAXv --  
exclude={"/dev/*","/proc/*","/sys/*","/tmp/*","/run/*","/mnt/*","/media/*","/  
lost+found"} / /mnt/
```

Damit werden alle Verzeichnisse und Daten kopiert, mit Ausnahme der virtuellen Laufwerke, wie `/dev`, `/proc`, `/sys` und so weiter. Dieser werden später beim ersten Neustart des Rechner automatisch wieder hergestellt.

Und das ist eigentlich schon alles, was die Übernahme der Daten angeht. Als nächstes müssen jetzt noch einige Anpassungen vorgenommen werden, sowie ein modifizierter GRUB Bootlader installiert werden.

5. Konfiguration des neuen Systems

5.1 Wurzelverzeichnis virtuell ändern

Für die weiteren Schritte wird das jetzt noch gemountete `/mnt/...` direkt nach `/...` virtuell geändert. Dies ist nur so lange gültig, bis der Befehl `exit` abgesetzt wird, oder das Terminalfenster geschlossen wird:

```
# manjaro-chroot /mnt /bin/bash
```

5.2 Schlüsseldatei anlegen

Damit später nicht bei einem Neustart jedesmal das Passwort zweimal eingegeben werden muss, also einmal für den GRUB Bootlader und einmal dann für das System selber, wird eine Schlüsseldatei angelegt, mit dieser die Partition ebenfalls entschlüsselt werden kann. Diese verschlüsselte Datei wird dann in das `initramfs` geschrieben, welche aber ebenfalls im verschlüsselten Bereich liegt. Mit dem ersten Passwort kann der GRUB geladen werden, welcher dann `initramfs` startet und diese kann dann durch den hinterlegten Schlüssel auf den Rest zugreifen, um es mal vereinfacht auszudrücken. Da das `/boot` Verzeichnis ja verschlüsselt ist, gibt es hier keine Sicherheitsbedenken, das ohne das Kennwort nicht an die `initramfs` heran gekommen werden kann:

```
dd bs=512 count=4 iflag=fullblock if=/dev/random of=/root/crypt_keyfile.bin
openssl genrsa -out /root/crypt_keyfile.bin 4096
chmod 600 /root/crypt_keyfile.bin
cryptsetup -v luksAddKey --cipher=aes-xts-plain64 --key-size=512 --
hash=whirlpool --type=luks2 --pbkdf=argon2id /dev/nvme0n1p3
/root/crypt_keyfile.bin
```

Im ersten Schritt wird ein Platzhalter unterhalb von `/root` in Form der Datei `crypt_keyfile.bin` angelegt. Danach wird über `openssl` ein komplexer Schlüssel generiert und in diese Datei gespeichert. Im 3. Schritt werden die Rechte so gesetzt, dass nur der Root User Zugriff darauf hat. Und im letzten Schritt wird dieser Schlüssel, ähnlich einem Passwort, dem LUKS2 Container hinzugefügt.

5.3 Neue `initramfs` erzeugen

Zunächst muss die Datei `/etc/mkinitcpio.conf` angepasst werden. Dort finden sich verschiedene Abschnitte. Eventuell stehen dort schon diverse Werte drin. Bei **MODULES**, **BINARIES** und **FILES** werden diese dann erweitert, während **HOOKS** ausgetauscht wird. Also am besten den bestehenden Eintrag mit `#` auskommentieren:

```
MODULES=() --> MODULES=(btrfs)
BINARIES=() --> BINARIES=(btrfsck)
FILES=() --> FILES=(/root/crypt_keyfile.bin)
```

```
HOOKS=(...) --> HOOKS=(base udev kms keyboard autodetect keymap consolefont  
modconf block encrypt btrfs filesystems fsck)
```

Fertig sieht meine Datei wie folgt aus (Kann je nach Gegebenheiten abweichen und soll nur als Beispiel dienen):

```
/etc/mkinitcpio.conf
```

```
# vim:set ft=sh
# MODULES
# The following modules are loaded before any boot hooks are
# run.  Advanced users may wish to specify all system modules
# in this array.  For instance:
MODULES=(btrfs)

# BINARIES
# This setting includes any additional binaries a given user may
# wish into the CPIO image.  This is run last, so it may be used to
# override the actual binaries included by a given hook
# BINARIES are dependency parsed, so you may safely ignore libraries
BINARIES=(btrfsck)

# FILES
# This setting is similar to BINARIES above, however, files are added
# as-is and are not parsed in any way.  This is useful for config files.
FILES=(/root/crypt_keyfile.bin)

# HOOKS
# This is the most important setting in this file.  The HOOKS control the
# modules and scripts added to the image, and what happens at boot time.
# Order is important, and it is recommended that you do not change the
# order in which HOOKS are added.  Run 'mkinitcpio -H <hook name>' for
# help on a given hook.
# 'base' is required unless you know precisely what you are doing.
# 'udev' is required in order to automatically load modules
# 'filesystems' is required unless you specify your fs modules in MODULES
# Examples:
## This setup specifies all modules in the MODULES setting above.
```

```
## No RAID, lvm2, or encrypted root is needed.
# HOOKS=(base)
#
## This setup will autodetect all modules for your system and should
## work as a sane default
# HOOKS=(base udev autodetect modconf block filesystems fsck)
#
## This setup will generate a 'full' image which supports most systems.
## No autodetection is done.
# HOOKS=(base udev modconf block filesystems fsck)
#
## This setup assembles a mdadm array with an encrypted root file system.
## Note: See 'mkinitcpio -H mdadm_udev' for more information on RAID
devices.
# HOOKS=(base udev modconf keyboard keymap consolefont block mdadm_udev
encrypt filesystems fsck)
#
## This setup loads an lvm2 volume group.
# HOOKS=(base udev modconf block lvm2 filesystems fsck)
#
## NOTE: If you have /usr on a separate partition, you MUST include the
# usr and fsck hooks.
#HOOKS=(base udev autodetect kms modconf block keyboard keymap consolefont
plymouth filesystems)
HOOKS=(base udev kms keyboard autodetect keymap consolefont modconf block
encrypt btrfs filesystems fsck)

# COMPRESSION
# Use this to compress the initramfs image. By default, gzip compression
# is used. Use 'cat' to create an uncompressed image.
#COMPRESSION="gzip"
#COMPRESSION="bzip2"
#COMPRESSION="lzma"
#COMPRESSION="xz"
#COMPRESSION="lzop"
#COMPRESSION="lz4"
#COMPRESSION="zstd"

# COMPRESSION_OPTIONS
```

```
# Additional options for the compressor
#COMPRESSION_OPTIONS=()

# MODULES_DECOMPRESS
# Decompress kernel modules during initramfs creation.
# Enable to speedup boot process, disable to save RAM
# during early userspace. Switch (yes/no).
#MODULES_DECOMPRESS="yes"
```

Wenn die Änderungen durchgeführt wurden, kann die initramfs neu erzeugt werden:

```
# mkinitcpio -P linux
```

5.4 Modifizierter GRUB installieren

5.4.1 Vorhandenes GRUB Paket deinstallieren

Der aktuelle GRUB Bootloader hat leider immer noch nur eine sehr rudimentäre Unterstützung für LUKS. Es gibt aber im [AUR](#) ein entsprechendes Paket, welches auch unter Manjaro problemlos eingesetzt werden kann. Dazu wird der Quelltext heruntergeladen, automatisch gepatcht und dann installiert. Da dieser mit dem schon installierten GRUB kollidieren würde, muss der alte GRUB zuvor deinstalliert werden:

```
# pacman -Rsync grub
```

Damit werden aber auch die Abhängigkeiten, wie z.B. das Manjaro Bootload Theme deinstalliert. Diese müssen später wieder installiert werden. Während der Deinstallation werden diese Pakete auch genannt. Diese sollten notiert werden. Bei mir waren dies:

- grub-btrfs

- update-grub
- grub-theme-manjaro

5.4.2 Modifiziertes GRUB Paket installieren

Da die Pakete vom AUR nur als normaler Benutzer installiert werden können, muss zunächst ein sudo mit dem zuvor eingerichteten temporären Benutzer erfolgen:

```
# sudo su - tempuser
```

Anschließend kann mit der Paketverwaltung yay der neue GRUB installiert werden:

```
$ yay -S grub-improved-luks2-git --noconfirm  
$ exit
```

Dies dauert nun einige Zeit, da erst die Quelltexte von GRUB und dessen Patches heruntergeladen werden müssen. Zusätzlich werden eventuelle Abhängigkeiten aufgelöst. Danach wird der GRUB kompiliert und ein entsprechendes Paket erzeugt, welches im Anschluß automatisch installiert wird. Nach der Installation wird mit exit wieder zurück auf die Root Shell gewechselt.

Danach können die zuvor deinstallierten Abhängigkeiten wieder installiert werden:

```
# pacman -S grub-btrfs update-grub grub-theme-manjaro
```

5.4.3 GRUB konfigurieren

Als erstes wird die UUID der verschlüsselten Partition benötigt. Diese kann über blkid abgefragt werden:

```
# blkid -s UUID -o value /dev/nvme0n1p3
```

Zurück wird die entsprechende UUID geliefert:

```
427763c1-500f-45e0-bb3f-08176cabfec
```

Diese UUID wird nun benötigt für die Konfiguration des GRUB Bootloaders. Dazu wird die Konfigurationsdatei in einem Editor geöffnet:

```
# vi /etc/default/grub
```

Hier werden zunächst die folgenden 3 Zeilen angepasst:

- GRUB_CMDLINE_LINUX_DEFAULT="..."
- GRUB_CMDLINE_LINUX=""
- GRUB_PRELOAD_MODULES=""

In den meisten Fällen wird bei den beiden letzten Zeilen vermutlich nichts weiter eingetragen sein. Diese müssen wie folgt ergänzt werden:

- GRUB_CMDLINE_LINUX_DEFAULT="... **cryptdevice=UUID=427763c1-500f-45e0-bb3f-08176cabfec:cryptbtrfs**"
- GRUB_CMDLINE_LINUX="**cryptkey=rootfs:/root/crypt_keyfile.bin**"
- GRUB_PRELOAD_MODULES="**part_gpt part_msdos luks2 cryptodisk argon2 gcry_whirlpool gcry_rijndael gcry_sha256 gcry_sha512 btrfs**"

Wichtig ist es, die UUID entsprechend der zuvor ermittelten UUID anzupassen.

Zusätzlich muss bei der Zeile

- `#GRUB_ENABLE_CRYPTODISK=y`

das "#" entfernt werden.

Auch muss das Theme wieder aktiviert werden. Dazu die Zeile

- `#GRUB_THEME="/path/to/gfxtheme"`

suchen und durch folgenden Eintrag ersetzen:

- `GRUB_THEME="/usr/share/grub/themes/manjaro/theme.txt"`

Darauf achten, dass auch hier ggf. das "#" vor der Zeile entfernt wird.

Anschließend sah die Datei `/etc/default/grub` bei mir wie folgt aus (Bitte nur als Beispiel nehmen, da zumindest die UUID bei jedem anders ist):

```
/etc/default/grub
# GRUB boot loader configuration

GRUB_DEFAULT=0
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="Manjaro"
GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet splash
cryptdevice=UUID=427763c1-500f-45e0-bb3f-08176cabfeca:cryptbtrfs"
GRUB_CMDLINE_LINUX="cryptkey=rootfs:/root/crypt_keyfile.bin"

# Preload both GPT and MBR modules so that they are not missed
```

```
GRUB_PRELOAD_MODULES="part_gpt part_msdos luks2 cryptodisk argon2
gcry_whirlpool gcry_rijndael gcry_sha256 gcry_sha512 btrfs"

# Uncomment to enable booting from LUKS encrypted devices
GRUB_ENABLE_CRYPTODISK=y

# Set to 'countdown' or 'hidden' to change timeout behavior,
# press ESC key to display menu.
GRUB_TIMEOUT_STYLE=menu

# Uncomment to use basic console
GRUB_TERMINAL_INPUT=console

# Uncomment to disable graphical terminal
#GRUB_TERMINAL_OUTPUT=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
GRUB_GFXMODE=auto

# Uncomment to allow the kernel use the same resolution used by grub
GRUB_GFXPAYLOAD_LINUX=keep

# Uncomment if you want GRUB to pass to the Linux kernel the old parameter
# format "root=/dev/xxx" instead of "root=/dev/disk/by-uuid/xxx"
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
GRUB_DISABLE_RECOVERY=true

# Uncomment and set to the desired menu colors. Used by normal and wallpaper
# modes only. Entries specified as foreground/background.
#GRUB_COLOR_NORMAL="light-blue/black"
#GRUB_COLOR_HIGHLIGHT="light-cyan/blue"

# Uncomment one of them for the gfx desired, a image background or a gfxtheme
#GRUB_BACKGROUND="/path/to/wallpaper"
GRUB_THEME="/usr/share/grub/themes/manjaro/theme.txt"
```

```
# Uncomment to get a beep at GRUB start
#GRUB_INIT_TUNE="480 440 1"

# Uncomment to make GRUB remember the last selection. This requires
# setting 'GRUB_DEFAULT=saved' above.
#GRUB_SAVEDEFAULT=true

# Uncomment to disable submenus in boot menu
#GRUB_DISABLE_SUBMENU=y
```

5.4.4 GRUB Bootloader installieren

Als erstes wird nun mit `grub-install` der Bootloader installiert:

```
# grub-install --target=x86_64-efi --efi-directory=/efi --bootloader-id=GRUB
--recheck --modules="part_gpt part_msdos luks2 cryptodisk argon2
gcry_whirlpool gcry_rijndael gcry_sha256 gcry_sha512 btrfs"
```

Und anschließend noch die Konfiguration nach `/boot/grub/grub.cfg` geschrieben:

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

Und die Rechte für das `/boot` Verzeichnis nur für den Root Benutzer gesetzt:

```
# chmod 700 /boot
```

5.5 Rechner neu starten

Jetzt kommt der große Moment, zum ersten Mal wird die verschlüsselte Manjaro Version gebootet. Dazu die virtuelle Root Umgebung verlassen:

```
# exit
```

Und dann anschließend den Rechner neu starten:

```
# reboot
```

6. Abschließende Maßnahmen

Nach dem Neustart wird man mit einer recht lieblosen Passwortabfrage begrüßt:

```
Enter passphrase for hd0.gpt3 (427763c1-500f-45e0-bb3f-08176cabfeca): _
```

Nach Eingabe des in Kapitel "3.3 Partition verschlüsseln" vergebenen Passwortes, sollte wie gewohnt der GRUB Bootlader Bildschirm angezeigt werden. Nach Eingabe des Kennwortes dauert es aber etwas, da die Partition erst entschlüsselt werden muss. Da der GRUB Bootlader keine Möglichkeit hat, auf die entsprechenden CPU Verschlüsselungs-Beschleuniger zuzugreifen, kann dies je nach Rechenleistung einige Sekunden dauern.

Nach dem Bootloader sollte der Rechner wie gewohnt starten und wieder mit der grafischen Oberfläche begrüßen. Hier kann sich wieder ggf. ganz normal mit dem eigenen Benutzer angemeldet werden, sofern nicht die automatische Anmeldung aktiv war.

6.1 Temporären Benutzer wieder löschen

Der am Anfang angelegte Benutzer kann nun wieder gelöscht werden. Dazu wieder über **Systemeinstellungen** --> **Benutzer** auf der rechten Seite den User auswählen und dort dann auf Benutzer löschen... klicken.

6.2 LUKS Passwort ändern

Jetzt kann das zukünftige Passwort gesetzt werden, sofern nicht direkt von Anfang an das endgültige Passwort gesetzt wurde. Die Kennwörter, bzw. Schlüsseldateien werden in sogenannten Slots gespeichert. Insgesamt sind dies 8 Slots. Es ist also auch möglich, mehrere Passwörter oder Keyfiles zu speichern. Wenn man sehen möchte, wie viele Slots besetzt sind, ist dies mit dem Parameter `luksDump` möglich:

```
sudo cryptsetup luksDump /dev/nvme0n1p3
```

In der Ausgabe sieht man dann, dass insgesamt 2 Slots in Benutzung sind:

LUKS header information

```
Version:          2
Epoch:           4
Metadata area:    16384
Keyslots area:    16744448
UUID:             427763c1-500f-45e0-bb3f-08176cabfeca
Label:            (no label)
Subsystem:        (no subsystem)
Flags:            (no flags)
```

Data segments:

```
0: crypt
   offset: 16777216
   length: (whole device)
   cipher: aes-xts-plain64
   sector: 512
```

Keyslots:

```
0: luks2
   Key:          512 bits
   Priority:      normal
   Cipher:        aes-xts-plain64
   Cipher key:    512 bits
   PBKDF:         argon2id
```

Time cost: 10
Memory: 1048576
Threads: 4
Salt: f9 9b 33 2f 2d 06 7f b7 53 03 c3 aa ee 43 1e 1f
4d a6 59 98 e5 81 2d 9e 99 07 32 1b 1b 32 87 c4
AF stripes: 4000
AF hash: whirlpool
Area offset:32768
Area length:258048
Digest ID: 0

1: luks2

Key: 512 bits
Priority: normal
Cipher: aes-xts-plain64
Cipher key: 512 bits
PBKDF: argon2id
Time cost: 10
Memory: 1048576
Threads: 4
Salt: 96 bf 20 97 88 8b e8 d2 2c e1 5e 75 fa 47 77 d6
33 51 bc 36 23 b7 a9 14 3c 5a 2d c6 f9 cf 58 d3
AF stripes: 4000
AF hash: whirlpool
Area offset:290816
Area length:258048
Digest ID: 0

Tokens:

Digests:

0: pbkdf2

Hash: whirlpool
Iterations: 86345
Salt: 5a 48 73 37 e8 c1 e2 b6 65 dd 40 8d 0a 41 5d 6c
45 63 ba 92 ad d3 a9 f6 1d 8c ab 41 f3 59 96 e2
Digest: c3 01 32 b4 ae 5a c1 06 e5 19 aa 16 99 7a 1b a3
c1 a3 b8 41 b2 76 ba ac bd ac 66 d4 e6 d8 72 45
77 ae 2f 3d cf b7 44 80 28 18 6f 09 d3 bd 00 ff
6e 2a 0b d8 fa 35 67 38 6c a9 9b b9 c2 c0 87 1b

Im ersten Slot 0 ist das Kennwort und im zweiten Slot 1 die Schlüsseldatei gespeichert. Geändert werden soll das Kennwort in Slot 0:

```
$ sudo cryptsetup luksChangeKey /dev/nvme0n1p3 -S 0
```

6.3 LUKS Header sichern

Im LUKS Header steht alles drin, damit der Container überhaupt entschlüsselt werden kann. Geht dieser kaputt, kann auch mit dem dazugehörigen Passwort der Container nicht mehr geöffnet werden. Daher sollte unbedingt ein Backup dieses Headers gemacht werden. Und dieses Backup muss natürlich sicher verwahrt werden. Denn mit diesem Backup wäre es Dritten möglich, die Festplatte auch ohne Passwort zu entschlüsseln. Der einfachste Weg wäre es, wenn man mehrere verschlüsselte Rechner hat. So könnte man die Backups gegenseitig auf die Rechner sichern.

Oder über Tools, wie zum Beispiel KeePassXC wäre eine sichere Speicherungsmöglichkeit gegeben.

Um den Header auszulesen und in eine Datei abzuspeichern, ist folgender Befehl hilfreich:

```
$ sudo cryptsetup luksHeaderBackup /dev/nvme0n1p3 --header-backup-file /tmp/luksheader.txt
```

Möchte man sich diese Datei anschauen, dann erfolgt dies wieder über den Parameter luksDump:

```
$ sudo cryptsetup luksDump /tmp/luksheader.txt
```

6.3.1 LUKS Header Backup zurück schreiben

Sollte dieser Header mal beschädigt worden sein, kann zum Beispiel über das Booten einer Live CD mit folgendem Befehl das Backup wieder zurück geschrieben werden:

```
$ sudo cryptsetup luksHeaderRestore /dev/nvme0n1p3 --header-backup-file /tmp/luksheader.txt
```

6.4 Recovery Keys generieren

Es kann auch nicht schaden, ein zusätzliches Recovery Passwort zu setzen. Dieses kann auch deutlich komplexer sein, weil es ja ggf. nur im Notfall benötigt wird. Entweder man "denkt" sich selber ein sehr aufwendiges Kennwort aus oder nutzt dazu die diversen Passwortgeneratoren im Netz.

6.4.1 Eigenes Passwort nutzen

Wenn man ein eigenes Passwort nehmen möchte, dann geht dies so:

```
$ cryptsetup -v luksAddKey --cipher=aes-xts-plain64 --key-size=512 --hash=whirlpool --type=luks2 --pbkdf=argon2id /dev/nvme0n1p3
```

Danach gibt man sein zusätzliches Passwort ein.

6.4.2 Automatisch Recovery-Key generieren lassen

Oder man nutzt die entsprechend vorhandenen Mechanismen:

```
§ sudo systemd-cryptenroll /dev/nvme0n1p3 --recovery-key
```

Anschließend wird ein entsprechend komplexer Key ausgegeben.


```
Tools : zsh — Konsole
Datei Bearbeiten Ansicht Lesezeichen Module Einstellungen Hilfe
Neues Unterfenster Ansicht teilen
Kopieren Einfügen Suchen

Please enter current passphrase for disk /dev/sda1: .....
A secret recovery key has been generated for this volume:

  utfhtvbb-gijtrngk-lbdvulvh-lunirjfi-vgrucdtj-dfthicvj-djctvilf-ktrddrg

Please save this secret recovery key at a secure location. It may be used to
regain access to the volume if the other configured access credentials have
been lost or forgotten. The recovery key may be entered in place of a password
whenever authentication is requested.

You may optionally scan the recovery key off screen:



New recovery key enrolled as key slot 3.
```

6.5 Alte Manjaro Partition löschen

Letztendlich kann nun auch die alte Partition, wo das unverschlüsselte Manjaro noch installiert ist, gelöscht werden. Dies kann man recht einfach über das grafische Tool GParted erledigen. Dort die entsprechende Partition mit der Maus auswählen, dort dann rechte Maustaste drücken und auf **Löschen** klicken. Danach auf **Bearbeiten** und dann auf **Alle Operationen ausführen** klicken.

7. Fazit

Die Kombination aus Manjaro Linux, Btrfs Subvolumes und vollständiger Festplattenverschlüsselung bietet eine robuste und hochsichere Umgebung für Benutzer, die ihre Daten schützen möchten. Durch die Verwendung von Btrfs Subvolumes können Benutzer ihr Dateisystem effizient verwalten und von den Vorteilen der Schnappschüsse profitieren, während die vollständige Verschlüsselung der Festplatte, einschließlich des /boot-Verzeichnisses, sicherstellt, dass sensible Daten vor unbefugtem Zugriff geschützt sind.

Das manuelle Anlegen von Btrfs Subvolumes erfordert zwar einige technische Kenntnisse, bietet aber eine flexible Möglichkeit, das Dateisystem zu organisieren und Snapshots zu erstellen, die bei der Wiederherstellung oder Fehlerbehebung hilfreich sind.

Die vollständige Verschlüsselung der Festplatte gewährleistet einen umfassenden Schutz sensibler Daten vor Diebstahl oder unbefugtem Zugriff. Dieser Prozess erfordert zwar zusätzliche Schritte beim Einrichten des Systems, bietet jedoch eine beruhigende Sicherheit und ermöglicht es Benutzern, ihre Privatsphäre zu wahren.

Das Übertragen einer bestehenden Manjaro Linux-Installation in eine verschlüsselte Systemumgebung erfordert Sorgfalt und technisches Know-how, aber mit den richtigen Schritten ist dies machbar und ermöglicht es Benutzern, von den Sicherheitsvorteilen der Verschlüsselung zu profitieren, ohne ihre bereits vorhandene Konfiguration aufgeben zu müssen.

Insgesamt bietet die Kombination aus Manjaro Linux, Btrfs Subvolumes und vollständiger Verschlüsselung eine leistungsstarke und sichere Plattform, auf der Benutzer ihre Daten effektiv schützen können, ohne dabei Kompromisse bei der Benutzerfreundlichkeit einzugehen.

Wer anstatt Manjaro lieber Arch Linux benutzt, findet [hier](#) eine entsprechende Anleitung zur Verschlüsselung.

Quellen:

<https://manjaro.org/>
<https://de.wikipedia.org/wiki/Dm-crypt>
https://github.com/eightbyte81/MAW/blob/main/README_en.md
<https://github.com/Szwendacz99/Arch-install-encrypted-btrfs>
<https://blog.wiuma.de/arch/2017/03/04/Arch-Migration>
<https://amaikinono.github.io/install-minimal-manjaro.html>
<https://subshell.com/de/blog/authentifizierung-yubikey100.html>
<https://aur.archlinux.org/packages/grub-improved-luks2-git>

There are no comments yet.