LVM on LUKS (encrypted, Laptop)

LUKS (Linux Unified Key Setup) is the standard for Linux hard disk encryption. By providing a standard on-disk-format, it does not only facilitate compatibility among distributions, but also provides secure management of multiple user passwords. LUKS stores all necessary setup information in the partition header, enabling to transport or migrate data seamlessly.

Management of LUKS encrypted devices is done via the `cryptsetup` utility.

Nomenclature

Term	Description
Physical Volume (PV)	On-disk partitioning format to be combined in a VG to a common storage pool
Volume Group (VG)	Grouping of one or more PVs to provide a combined storage pool from which storage can be requested in the form of LVs.
Logical Volume (LV)	Logical partition format which can be accessed like a block device to hold file systems and data.

Partitioning Setup

> **NOTE:** This partitioning scheme does **NOT** include an LVM cache device.
>
> While it is technically possible to add an LVM cache device to this setup, it is not advised to do so, ***as this will leak plain text contents of the unlocked LUKS container into the cache,*** which can be read in a hex editor by opening the raw device file directly — entirely defeating the purpose of encrypting the disk!
>
> A LUKS on LVM setup is recommended instead.

LVM on LUKS has the benefit of being able to encrypt an entire drive (useful for laptops with encrypted swap for resume) while only needing to provide a single passphrase to unlock it entirely for simplicity.

However, since the LVM container resides inside the LUKS container it cannot span multiple disks, as it is confined by the boundaries by the parent LUKS container.

This guide assumes the following:

- This is used on a laptop computer with resume capabilities (Swap partition)
- There is only one drive: `/dev/nvme0n1`
- The root file system will be btrfs, with subvolumes for `/` and `/home`
- To tighten security, this setup assumes a unified kernel image and booting via EFISTUB, with the *ESP* mounted at `/efi`. Extra steps will be necessary to make the machine bootable.

Preparing the drive

1. List available disks

   ```
   fdisk -l
   ```

2. Start partitionaing tool for primary disk (*`cfdisk` is a little easier to use as it has a nice TUI)*

   > **WARNING:** Make sure to select your actually desired device!

   ```
   cfdisk /dev/nvme0n1
   ```

3. Partition with the following scheme

FS Type	Size	Mount Point	Comment
vfat	1G	`/efi`	EFI System
LUKS	(remaining)		Linux file system

Creating the LUKS container

1. Create the LUKS container and enter a passphrase

   > **WARNING:** Do **NOT** forget your passphrase! In case of loss you won't be able to access the data inside the container anymore!

   ```
   cryptsetup luksFormat /dev/nvme0n1p2
   ```

2. Open the newly created LUKS container

   > **NOTE:** `cryptlvm` is used as an example here. Use whatever you like.

```
cryptsetup open /dev/nvme0n1p2 cryptlvm
```

Creating LVM inside the LUKS container

1. Create an LVM physical volume inside LUKS container

   ```
   pvcreate /dev/mapper/cryptlvm
   ```

2. Create the volume group:

   ```
   vgcreate vg0 /dev/mapper/cryptlvm
   ```

3. Create the logical volumes

   > **NOTE:** When using resume, make `lv_swap` as large as RAM. In this example the machine has **16 GB** of RAM.

   ```
   lvcreate -L 16G -n lv_swap vg0       # Swap as big as RAM (16 GB)
   lvcreate -l 100%FREE -n lv_root vg0  # Root file system
   ```

Formatting devices

1. Create partitions

   ```
   mkfs.fat -F 32 /dev/nvme0n1p1        # EFI System Partition
   mkfs.btrfs /dev/mapper/vg0-lv_root   # Btrfs root volume
   mkswap /dev/mapper/vg0-lv_swap       # Swap space
   ```

2. Create Btrfs subvolumes

   ```
   # First, mount the root file system
   mount /dev/mapper/vg0-lv_root /mnt

   # Create subvolumes
   btrfs subvolume create /mnt/@
   btrfs subvolume create /mnt/@home
   ```

3. Mount partitions

   ```
   # Unmount the root file system
   umount -R /mnt

   # Mount the @ subvolume
   ```

```
mount /dev/mapper/vg0-lv_root -o noatime,compress-force=zstd,space_cache=v2,subvol=@
/mnt

# Create mountpoints
mkdir -p /mnt/{efi,home}

# Mount the remaining partitions/subvolumes
mount /dev/nvme0n1p1 /mnt/efi
mount /dev/mapper/vg0-lv_root -o noatime,compress-
force=zstd,space_cache=v2,subvol=@home /mnt/home

# Activate swap
swapon /dev/mapper/vg0-lv_swap
```

Revision #17
Created 2021-09-11 13:44:38 UTC by Sebin
Updated 2024-03-06 07:18:50 UTC by Sebin