Linux LUKS Lab
By Michael Ambeguia

Purpose: The Purpose of this lab is to gain hands-on experience applying LUKS full disk encryption. LUKS is used to protect data at rest on Linux systems and encrypts the full disk. Unlike other types of encryption such as volume level or file/folder level encryption LUKS will encrypt all the data on a drive which will reduce the chances of sensitive data not being encrypted. LUKS is also a great encryption option since it does not involve that much end user interaction once it is set up by an administrator. If LUKS is configured correctly the drive is unlocked automatically upon booting the system allowing users to read and modify the data.

Sections:

Section #1 Intro to LUKS:

1.1 What is LUKS? What is it used for?

LUKS or Linux Unified Key Setup is a Linux feature that is commonly used to perform full-disk encryption on Linux systems to secure data at rest. LUKS has user space commands but also works with the Kernel to perform the encryption. A Linux drive with LUKS encryption will be completely encrypted so there is no need to perform file, volume, or folder based encryption.

1.2 How does LUKS encrypt drives? How does it decrypt drives?

-Encryption: LUKS encrypts data using an encrypted master key that can only be unlocked by a user generated passphrase or a key file.

- Decryption: LUKS decrypts data by using the passphrase or key file to decrypt the master key. The master key is then used to decrypt the data itself.

It should be noted that LUKS only encrypts the data if the device is off! When the device is on and the drive is being used the data is not encrypted so anyone can view it.

1.3 How is LUKS secure?

1. LUKS uses strong symmetric key algorithms. LUKS can use AES-XTS or AES-CBC modes. It also supports serpent and twofish (2 other symmetric key algorithms).

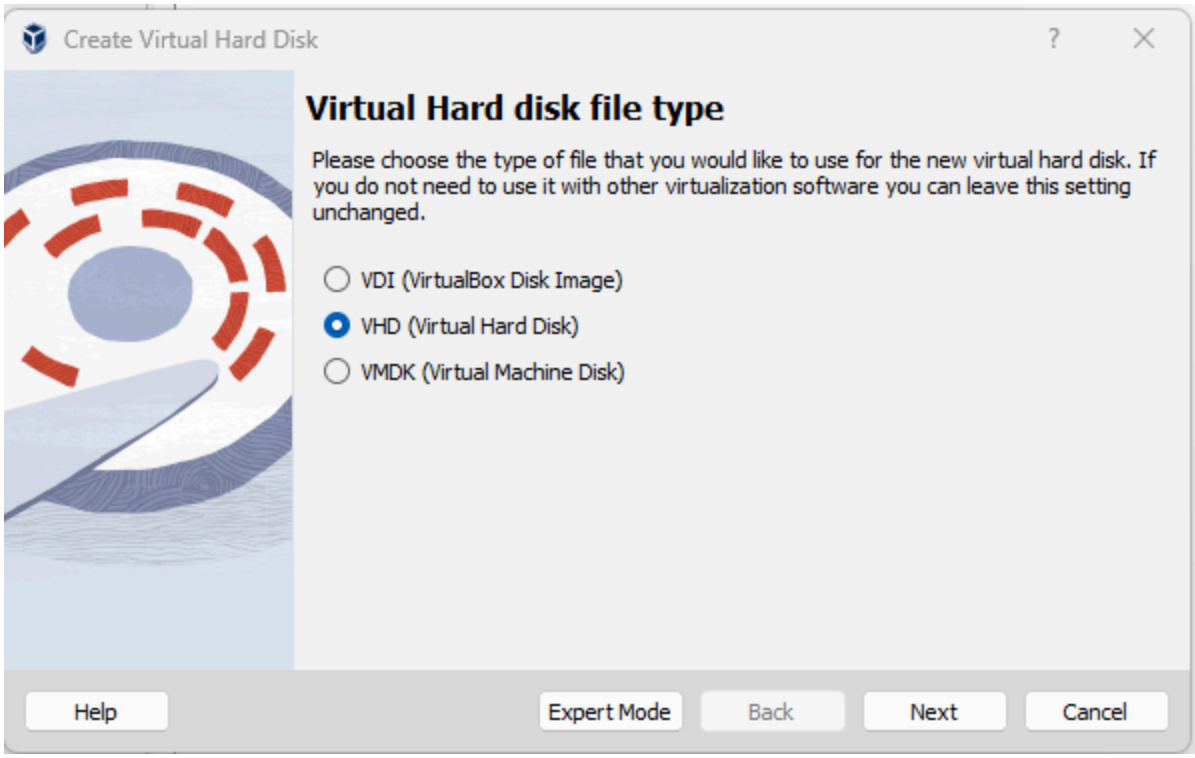
2. LUKS passphrase generated keys are strengthened with the use of Key Derivation Function algorithms. These algorithms make brute forcing the key impossible.

3. LUKS is tamper proof. LUKS stores the header and metadata for the encrypted drive in plaintext but messing with it will corrupt the whole drive. So if an attacker wanted to somehow manipulate the header and its data the drive will simply become useless.

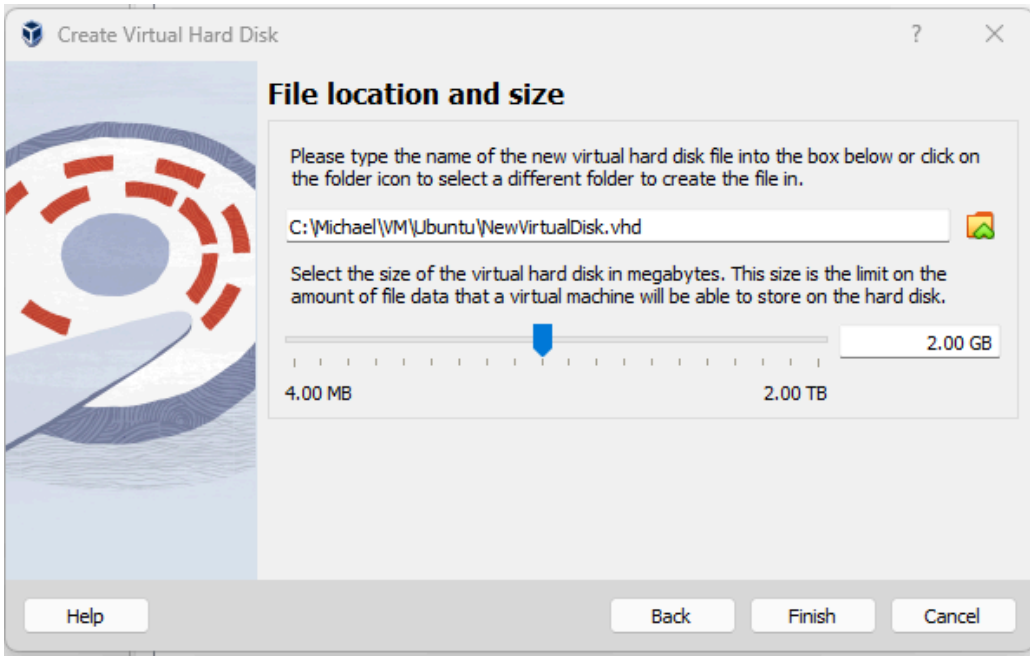
4. LUKS supports secure wiping functionality. If you wipe the header the drive will make the data useless. Another way to securely wipe LUKS is to get rid of keys (shred the key files and delete key slots).

Section #2 Prepare VHD:

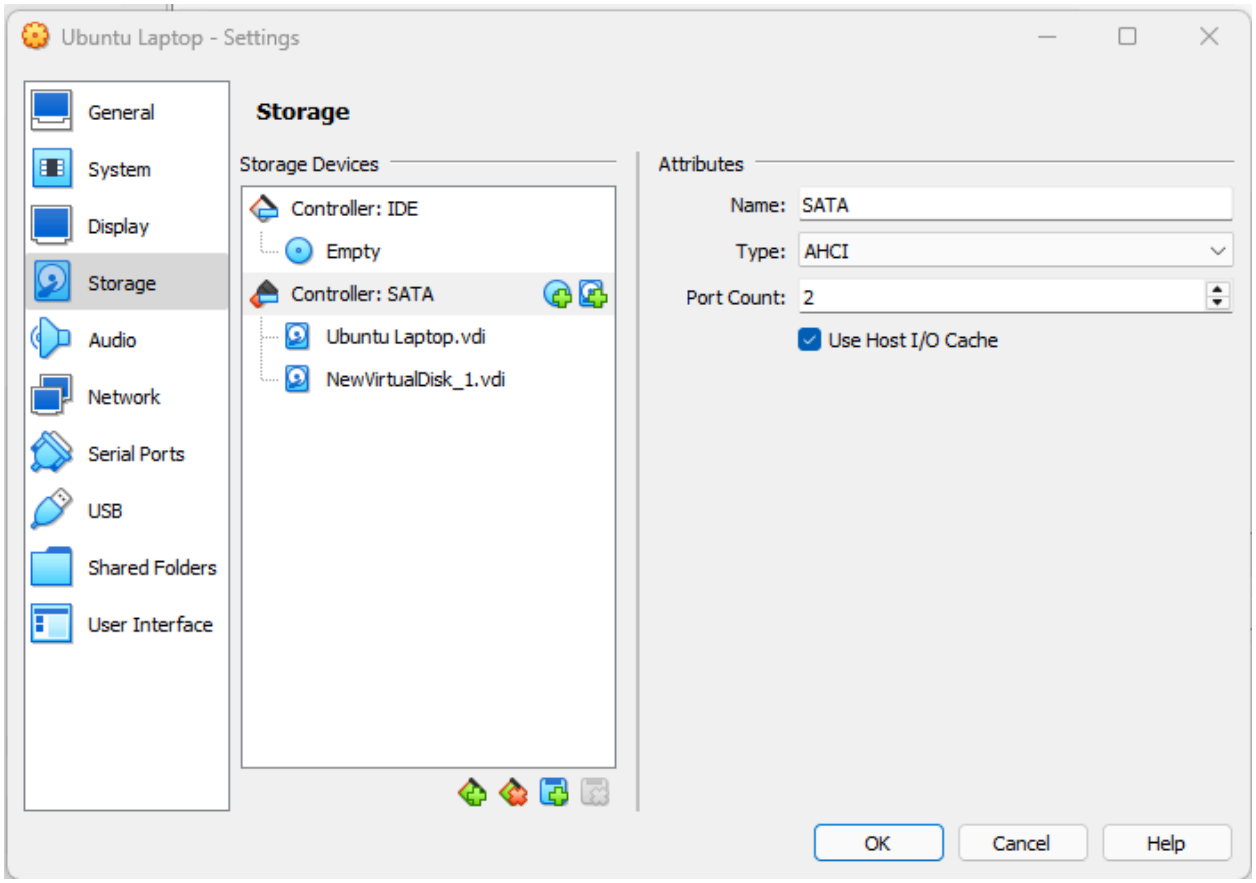
2.1 Create new VHD:


Create Virtual Hard Disk
Virtual Hard disk file type
Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.
VDI (VirtualBox Disk Image)
VHD (Virtual Hard Disk)
VMDK (Virtual Machine Disk)
Help
Expert Mode
Back
Next
Cancel


Create Virtual Hard Disk
Storage on physical hard disk
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).
A dynamically allocated hard disk file will only use space on your physical hard disk as it fills up (up to a maximum fixed size), although it will not shrink again automatically when space on it is freed.
A fixed size hard disk file may take longer to create on some systems but is often faster to use.
Pre-allocate Full Size
Split into 2GB parts
Help
Back
Next
Cancel



2.2 Attach VHD to system:



2.3 Identify new disk on VM:

Disk devices can be listed using the lsblk command. Note that the new drive I added is called sdb and that it has no mountpoint.

```
spy2@Laptop:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0     4K  1 loop /snap/bare/5
loop1    7:1    0  10.7M  1 loop /snap/canonical-livepatch/286
loop2    7:2    0  73.9M  1 loop /snap/core22/1748
loop3    7:3    0  73.9M  1 loop /snap/core22/1722
loop4    7:4    0  66.2M  1 loop /snap/core24/716
loop5    7:5    0  66.2M  1 loop /snap/core24/609
loop6    7:6    0 505.1M  1 loop /snap/gnome-42-2204/176
loop7    7:7    0   9.9M  1 loop /snap/nmap/3582
loop8    7:8    0 275.3M  1 loop /snap/firefox/5647
loop9    7:9    0  12.3M  1 loop /snap/nmap/3831
loop10   7:10   0  91.7M  1 loop /snap/gtk-common-themes/1535
loop11   7:11   0 271.6M  1 loop /snap/firefox/5014
loop12   7:12   0  44.4M  1 loop /snap/snapd/23545
loop13   7:13   0  12.2M  1 loop /snap/snap-store/1216
loop14   7:14   0   568K  1 loop /snap/snapd-desktop-integration/253
sda      8:0    0    25G  0 disk
├─sda1   8:1    0     1M  0 part
├─sda2   8:2    0   513M  0 part /boot/efi
└─sda3   8:3    0  24.5G  0 part /var/snap/firefox/common/host-hunspell
                                 /
sdb      8:16   0     2G  0 disk
sr0     11:0    1  1024M  0 rom
spy2@Laptop:~$
```

2.4 Initialize VHD and Partition VHD into one whole partition:

I will initialize the disk using the fdisk utility.

```
spy2@Laptop:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x0110aabb.

Command (m for help): g
Created a new GPT disklabel (GUID: 40185AFF-0977-4344-AD50-F97198E451BC).

Command (m for help): n
Partition number (1-128, default 1): 1
First sector (2048-4194270, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-4194270, default 4194270): +
2GB

Created a new partition 1 of type 'Linux filesystem' and of size 1.9 GiB.

Command (m for help):
```

The first thing I did was enter g for creating a new gpt guid partition table for the drive. I then entered n to create a new partition and 1 to signify that I want to only have one partition (the whole drive). I then entered the default first sector then set the last sector to 2GB.

2.5. Verify partition layout:

Using sudo fdisk -l:

The /dev/sdb drive is partitioned to a size of 2GB.

```
Disk /dev/sdb: 2 GiB, 2147483648 bytes, 4194304 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Section #3 Encrypt partition

3.1 Initialize LUKS on the drive:

Install LUKS on my Ubuntu VM:

I have to install the LUKS package (called cryptsetup) on my system since I don't have it.

```
spy2@Laptop:~$ sudo apt install cryptsetup
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cryptsetup-bin cryptsetup-initramfs
The following NEW packages will be installed:
  cryptsetup cryptsetup-bin cryptsetup-initramfs
0 upgraded, 3 newly installed, 0 to remove and 120 not upgraded.
Need to get 365 kB of archives.
After this operation, 1,245 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cryptsetup-bi
n amd64 2:2.4.3-1ubuntu1.2 [145 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cryptsetup am
d64 2:2.4.3-1ubuntu1.2 [193 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cryptsetup-in
itramfs all 2:2.4.3-1ubuntu1.2 [26.2 kB]
Fetched 365 kB in 1s (511 kB/s)
Preconfiguring packages ...
Selecting previously unselected package cryptsetup-bin.
(Reading database ... 281276 files and directories currently installed.)
Preparing to unpack .../cryptsetup-bin_2%3a2.4.3-1ubuntu1.2_amd64.deb ...
Unpacking cryptsetup-bin (2:2.4.3-1ubuntu1.2) ...
Selecting previously unselected package cryptsetup.
Preparing to unpack .../cryptsetup_2%3a2.4.3-1ubuntu1.2_amd64.deb ...
Unpacking cryptsetup (2:2.4.3-1ubuntu1.2) ...
Selecting previously unselected package cryptsetup-initramfs.
Preparing to unpack .../cryptsetup-initramfs_2%3a2.4.3-1ubuntu1.2_all.deb ...
Unpacking cryptsetup-initramfs (2:2.4.3-1ubuntu1.2) ...
Setting up cryptsetup-bin (2:2.4.3-1ubuntu1.2) ...
```

Once installed I was able to use sudo cryptsetup luksFormat [device] to encrypt the drive.

```
spy2@Laptop:~$ sudo cryptsetup luksFormat /dev/sdb

WARNING!
========
This will overwrite data on /dev/sdb irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/sdb:
Verify passphrase:
```

3.2 Set passphrase on partition:

LUKS requires a passphrase to be set in order to unlock a LUKS encrypted drive.

```
spy2@Laptop:~$ sudo cryptsetup luksFormat /dev/sdb

WARNING!
========
This will overwrite data on /dev/sdb irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/sdb:
Verify passphrase:
```

3.3 Verify partition is encrypted:

To verify the status you can use the cryptsetup luksDump [device name].

```
Data segments:
  0: crypt
        offset: 16777216 [bytes]
        length: (whole device)
        cipher: aes-xts-plain64
        sector: 512 [bytes]

Keyslots:
  0: luks2
        Key:        512 bits
        Priority:   normal
        Cipher:     aes-xts-plain64
        Cipher key: 512 bits
        PBKDF:      argon2id
        Time cost:  4
        Memory:     1048576
        Threads:    3
        Salt:       02 ac 47 f9 1f e1 a8 ad dc 4d 00 c6 f4 8f d1 98
                    82 f0 2d 19 e3 71 e5 d3 1a 69 9a 2d 6b 26 c2 51
        AF stripes: 4000
        AF hash:    sha256
        Area offset:32768 [bytes]
        Area length:258048 [bytes]
        Digest ID:  0
Tokens:
Digests:
  0: pbkdf2
        Hash:       sha256
        Iterations: 130031
        Salt:       7e af 9e 96 8b 12 97 41 f2 76 81 21 41 a1 b1 21
                    85 09 2b ba b0 29 5c bd c0 dd 73 91 f3 17 b5 63
        Digest:     13 89 50 dc a9 c6 4a 15 5a fb eb 8b 64 d1 68 37
                    14 bd 41 5b bb 14 ea 13 22 02 2e 31 3f 54 d5 18
spy2@Laptop:~$
```

The cipher algorithm used is aes-xts-plain64 and it encrypted the whole drive. The cipher uses a 512 bit sized key that is derived using argon2id PBKDF. The passphrase I entered is used in the argon2id key derivation process.

Section #4 Mount and Format Partition

4.1 Format decrypted device with filesystem:

- Open drive:
  Before I can put a filesystem on the /dev/sdb drive I need to unlock it (decrypt it).

```
spy2@Laptop:~$ sudo cryptsetup open /dev/sdb test_luks
Enter passphrase for /dev/sdb:
spy2@Laptop:~$
```

  I also gave the decrypted drive a device name (test_luks) which is used to identify the decrypted drive. It can be found in /dev/mapper/test_luks.

```
spy2@Laptop:~$ cd /dev/mapper/
spy2@Laptop:/dev/mapper$ ls -l
total 0
crw------- 1 root root 10, 236 Feb  4 10:56 control
lrwxrwxrwx 1 root root       7 Feb  4 11:34 test_luks -> ../dm-0
spy2@Laptop:/dev/mapper$
```

- Format drive:

  I formatted the drive with an ext4 filesystem. I used the dev/mapper/test_luks device name rather than /dev/sdb since the drive is decrypted.

```
spy2@Laptop:/dev/mapper$ sudo mkfs.ext4 /dev/mapper/test_luks
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 520192 4k blocks and 130048 inodes
Filesystem UUID: 0384d22e-accc-43a7-ac03-509b80cfefd7
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

4.2 create mount point for partition:
I created a mount point under the / directory for the LUKS drive.

```
spy2@Laptop:/$ sudo mkdir test_luks
[sudo] password for spy2:
spy2@Laptop:/$ ls -l
total 2744408
lrwxrwxrwx   1 root root          7 Mar  3  2024 bin -> usr/bin
drwxr-xr-x   4 root root       4096 Feb  4 11:20 boot
drwxrwxr-x   2 root root       4096 Mar  3  2024 cdrom
drwxr-xr-x  19 root root       4280 Feb  4 11:34 dev
drwxr-xr-x 150 root root      12288 Feb  4 11:18 etc
drwxr-xr-x   7 root root       4096 Dec 26 22:33 home
lrwxrwxrwx   1 root root          7 Mar  3  2024 lib -> usr/lib
lrwxrwxrwx   1 root root          9 Mar  3  2024 lib32 -> usr/lib32
lrwxrwxrwx   1 root root          9 Mar  3  2024 lib64 -> usr/lib64
lrwxrwxrwx   1 root root         10 Mar  3  2024 libx32 -> usr/libx32
drwx------   2 root root      16384 Mar  3  2024 lost+found
drwxr-xr-x   3 root root       4096 Feb  4 11:39 media
drwxr-xr-x   2 root root          0 Feb  4 10:56 misc
drwxr-xr-x   2 root root       4096 Feb 20  2024 mnt
drwxr-xr-x   3 root root       4096 Aug 11 13:02 nfs_mount
drwxr-xr-x   2 root root       4096 Feb 20  2024 opt
dr-xr-xr-x 277 root root          0 Feb  4 10:56 proc
drwx------   8 root root       4096 Feb  1 14:10 root
drwxr-xr-x  41 root root       1160 Feb  4 11:18 run
lrwxrwxrwx   1 root root          8 Mar  3  2024 sbin -> usr/sbin
drwxr-xr-x  14 root root       4096 Jan 14 13:27 snap
drwxr-xr-x   2 root root       4096 Feb 20  2024 srv
-rw-------   1 root root 2810183680 Mar  3  2024 swapfile
dr-xr-xr-x  13 root root          0 Feb  4 10:56 sys
drwxr-x---   2 root root       4096 Feb  4 11:41 test_luks
```

4.3 Add a keyfile to LUKS management.

In order to decrypt the drive automatically I will need to configure auto-unlocking which uses a key file. The key will be separate from the passphrase derived key I created since it will be in a file. To create the key I used the dd command to write 512 bits of random data to a key file.

```
spy2@Laptop:/$ sudo dd if=/dev/urandom bs=512 count=1 of=/etc/key_file
1+0 records in
1+0 records out
512 bytes copied, 0.0045343 s, 113 kB/s
spy2@Laptop:/$
```

Lock the drive.

```
spy2@Laptop:/$ sudo cryptsetup luksClose test_luks
spy2@Laptop:/$
```

Add the keyfile to the drive.

```
spy2@Laptop:/$ sudo cryptsetup luksAddKey /dev/sdb /etc/key_file
Enter any existing passphrase:
spy2@Laptop:/$
```

4.4 Configure auto-unlocking with keyfile:

Add the drive and keyfile to the /etc/crypttab:

On the /etc/crypttab file I need to add the mapper name of the LUKS drive, its UUID, and the keyfile I created.

```
spy2@Laptop: /
spy2@Laptop: /    spy2@Laptop: /
GNU nano 6.2                    /etc/crypttab *
# <target name> <source device>         <key file>      <options>
  test_luks UUID=4cedf707-cae4-4de2-8a9f-fcddfae277f0 /etc/key_file luks
```

4.4 Mount partition and verify it is accessible:

Now that the /etc/crypttab file is done I can now configure the drive to be mounted when the system boots in the /etc/fstab file. I added the UUID for the device, its mount point, and the settings that apply for the drive to the file. Note that the o after defaults is used to tell the system to not back up the drive. The 2 lets the fsck utility know that the drive is not containing the root filesystem.

```
spy2@Laptop: /dev/mapper
spy2@Laptop: /dev/mapper        spy2@Laptop: /
  GNU nano 6.2                    /etc/fstab *
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda3 during installation
UUID=1ba369d1-6cc1-467c-9eeb-3c728a37e7dd /               ext4    errors=r
# /boot/efi was on /dev/sda2 during installation
UUID=122E-0A0D  /boot/efi       vfat    umask=0077      0       1
/swapfile                                 none            swap    sw

# Encrypted drive
UUID=fedae4ec-32c5-4b75-976d-9bc5357089d0  /test_luks ext4 defaults 0  2
```

After modifying the /etc/fstab file you should always use the mount -a command to verify that the configuration works.

```
spy2@Laptop:/$ sudo mount -a
spy2@Laptop:/$
```

Section #5 Test reading and writing to partition:

5.1. Test the drive:
Restart the system then see if the drive is unlocked.

The drive is decrypted since it has the name test_luks. It is also auto-mounted on the /test_luks mount point.

```
spy2@Laptop:~$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINTS
loop0         7:0    0     4K  1 loop  /snap/bare/5
loop1         7:1    0  74.2M  1 loop  /snap/core22/1122
loop2         7:2    0  10.7M  1 loop  /snap/canonical-livepatch/286
loop3         7:3    0 266.6M  1 loop  /snap/firefox/3836
loop4         7:4    0  73.9M  1 loop  /snap/core22/1748
loop5         7:5    0   497M  1 loop  /snap/gnome-42-2204/141
loop6         7:6    0  12.3M  1 loop  /snap/snap-store/959
loop7         7:7    0  40.4M  1 loop  /snap/snapd/20671
loop8         7:8    0  91.7M  1 loop  /snap/gtk-common-themes/1535
loop9         7:9    0  44.4M  1 loop  /snap/snapd/23545
loop10        7:10   0   568K  1 loop  /snap/snapd-desktop-integration/253
loop11        7:11   0   452K  1 loop  /snap/snapd-desktop-integration/83
sda           8:0    0    25G  0 disk
├─sda1        8:1    0     1M  0 part
├─sda2        8:2    0   513M  0 part  /boot/efi
└─sda3        8:3    0  24.5G  0 part  /var/snap/firefox/common/host-hunspell
                                       /
sdb           8:16   0     2G  0 disk
└─test_luks 252:0    0     2G  0 crypt /test_luks
sr0          11:0    1  1024M  0 rom
```

Fix ownership and permissions for the mount. The mount point is owned by root but I want spy2 to have access. I changed the group ownership to the spy2 group and gave the spy2 group rwx permissions.

```
spy2@Laptop:/$ sudo chown :spy2 test_luks/
[sudo] password for spy2:
spy2@Laptop:/$ sudo chmod 760 test_luks/
spy2@Laptop:/$ ls -l
total 2744404
lrwxrwxrwx   1 root root          7 Mar  3  2024 bin -> usr/bin
drwxr-xr-x   4 root root       4096 Feb  5 15:38 boot
drwxrwxr-x   2 root root       4096 Mar  3  2024 cdrom
drwxr-xr-x  19 root root       4220 Feb  6 12:46 dev
drwxr-xr-x 131 root root      12288 Feb  6 12:43 etc
drwxr-xr-x   5 root root       4096 Mar  3  2024 home
lrwxrwxrwx   1 root root          7 Mar  3  2024 lib -> usr/lib
lrwxrwxrwx   1 root root          9 Mar  3  2024 lib32 -> usr/lib32
lrwxrwxrwx   1 root root          9 Mar  3  2024 lib64 -> usr/lib64
lrwxrwxrwx   1 root root         10 Mar  3  2024 libx32 -> usr/libx32
drwx------   2 root root      16384 Mar  3  2024 lost+found
drwxr-xr-x   3 root root       4096 Feb  5 15:16 media
drwxr-xr-x   2 root root       4096 Feb 20  2024 mnt
drwxr-xr-x   2 root root       4096 Feb 20  2024 opt
dr-xr-xr-x 249 root root          0 Feb  6 12:45 proc
drwx------   5 root root       4096 Mar  3  2024 root
drwxr-xr-x  34 root root        880 Feb  6 12:46 run
lrwxrwxrwx   1 root root          8 Mar  3  2024 sbin -> usr/sbin
drwxr-xr-x  12 root root       4096 Feb  5 15:14 snap
```

Changed the ownership of the mount point.

```
spy2@Laptop:/$ sudo chmod 770 test_luks/
spy2@Laptop:/$
```

Gave spy2 and the spy2 group full permissions on the mount point.

Create a test file.

```
spy2@Laptop:/$ cd test_luks/
spy2@Laptop:/test_luks$ touch secretfile.txt
spy2@Laptop:/test_luks$ echo "This file is top secret!!" >> secretfile.txt
echo "This file is top secrettouch secretfile.txt" >> secretfile.txt
spy2@Laptop:/test_luks$
```

I was able to create a file so everything is okay.

5.2 Verify that the drive is not encrypted:

```
spy2@Laptop:~$ sudo blkid /dev/mapper/test_luks
/dev/mapper/test_luks: UUID="fedae4ec-32c5-4b75-976d-9bc5357089d0" BLOCK_SIZE="4
096" TYPE="ext4"
spy2@Laptop:~$
```

The drive is decrypted since the type is now labeled as ext4 instead of crypt.