kifarunix.com

Automount LUKS Encrypted Device in Linux - kifarunix.com

gen_too

8-11 minutes

In this tutorial, you will learn how to automount LUKS encrypted device in Linux on system startup. Unless you configure the device to automount, it usually doesn't by default. However, if you enabled device encryption with LUKS during system install, the automount is usually setup and the device automatically mounts once you supply the correct drive encryption passphrase.

Please note that security wise, automounting an encrypted device might not be a good practise, IMO.

In our previous tutorial, we learnt how to encrypt a disk partition with LUKS in Linux. We will be using the same device to demonstrate how to automount LUKS Encrypted Device in Linux.

Below command lists the block device that we will use to demonstrate the auto-mounting procedure.

```
lsblk
```

```
NAME
MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda
8:0    0   15G  0 disk
```

```
├─sda1
8:1    0   13G  0 part  /
├─sda2
8:2    0    1K  0 part
└─sda5
8:5    0    2G  0 part  [SWAP]
sdb
8:16   0    4G  0 disk
└─sdb1
8:17   0    4G  0 part
  └─luks-242c24d8-ac65-413d-b3a2-eb7f2f0993b0
254:0    0    4G  0 crypt
```

Create LUKS Key File

With LUKS encryption, you can unlock the device by interactively supplying the passphrase or automatically specifying a key file containing the passphrase to unlock the drive.

To automount LUKS encrypted device in Linux, then you need to use the key file containing the passphrase.

To create the LUKS key file, you use the **dd** command as follows.

```
dd if=/dev/random of=/etc/.crypt-me bs=32 count=1
```

So, we use the **/etc/.crypt-me** file as our LUKS key file, can be a different file for you. The command above fills random data on the key file as evident by the command below;

```
xxd /etc/.crypt-me
```

```
00000000: 62cc f2b2 b431 fdb5 d908 8cfd b6c5 b27d
b....1.........}
```

```
00000010: f38b 877a 6575 279c 3c20 5b36 a5fa ce7d
...zeu'.< [6...}
```

Add a Passphrase to LUKS Key File

Once you have created a LUKS key file, you need to add a new passphrase to the file using the `cryptsetup` utility:

```
cryptsetup luksAddKey <device> <path-to-key-file>
```

```
cryptsetup luksAddKey /dev/sdb1 /etc/.crypt-me
```

You will be prompted to enter any existing passphrase.

If you specified the existing passphrase using the key file as well, then use the command below;

```
cryptsetup luksAddKey <device> <path-to-key-file>
--key-file <path-to-existing-passphrase-key-file>
```

```
cryptsetup luksAddKey /dev/sdb1 /etc/.crypt-me
--key-file ~/luks-key
```

For now, the device has two key slots used, as per out setup. To confirm, print the device details.

```
LUKS header information
Version:        2
Epoch:          4
Metadata area:  16384 [bytes]
Keyslots area:  16744448 [bytes]
UUID:           242c24d8-ac65-413d-b3a2-
eb7f2f0993b0
Label:          (no label)
Subsystem:      (no subsystem)
Flags:          (no flags)
```

```
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:      argon2i
        Time cost:  4
        Memory:     1003317
        Threads:    2
        Salt:       b3 c8 b0 69 db 38 cb bd 1c 58
d0 a2 8a b8 92 12
                    05 47 ca dd c7 3d dd 94 c0 f7
51 04 12 fb 3a 56
        AF stripes: 4000
        AF hash:    sha256
        Area offset:32768 [bytes]
        Area length:258048 [bytes]
        Digest ID:  0
  1: luks2
        Key:        512 bits
        Priority:   normal
```

```
        Cipher:     aes-xts-plain64
        Cipher key: 512 bits
        PBKDF:      argon2i
        Time cost:  4
        Memory:     984615
        Threads:    2
        Salt:       17 9c 29 fc 61 a2 a4 b0 8b 10
42 6d 51 a0 5b 37
                    77 18 ef db 05 40 79 71 79 88
0a b1 85 41 ee 41
        AF stripes: 4000
        AF hash:    sha256
        Area offset:290816 [bytes]
        Area length:258048 [bytes]
        Digest ID:  0
Tokens:
Digests:
  0: pbkdf2
        Hash:       sha256
        Iterations: 133338
        Salt:       e1 9b 70 5e 87 25 46 d6 08 20
43 60 6c ae 2c 06
                    42 fa 61 32 f0 fc ca 5f 10 f9
3d 63 dd 22 a4 96
        Digest:     e9 62 ab 83 4c 3c 81 88 52 08
42 9b 47 c2 e1 b6
                    d5 8a 59 88 5c 17 02 54 c4 89
36 7e 5f e0 f5 ec
```

Verify that you can unlock the disk with the key file created using

the command;

```
cryptsetup luksOpen <device> <name> --key-file
<path-to-key-file>
```

If the drive is already opened, then close if first;

```
cryptsetup -v luksClose luks-242c24d8-ac65-413d-
b3a2-eb7f2f0993b0
```

Next, verify the new key file can unlock the LUKS drive;

```
cryptsetup -v luksOpen /dev/sdb1 luks-242c24d8-
ac65-413d-b3a2-eb7f2f0993b0 --key-file
/etc/.crypt-me
```

Sample output;

```
Key slot 1 unlocked.
Command successful.
```

Automount LUKS Encrypted Device in Linux on System Startup

Update crypttab file with device information

Next, you need to add an entry to `/etc/crypttab` describing the information about the LUKS encrypted device that you need to automount.

An entry in **`/etc/crypttab`** should look like;

```
<target name> <source device> <key-file>
<options>
```

Where:

- **`target name`**: describes the mapped device name. For example,

if your device mapping is **/dev/mapper/name**, then **name** is the required target.

- **source device**: describes either the block special device or file that contains the encrypted data. This is specified using UUID=<uuid>, or LABEL=<label>, PARTUUID=<partuuid> or PARTLABEL=<partlabel>.

  You can obtain the UUID, PARTUUID using the **blkid** command. For example:

  ```
  blkid /dev/sdb1
  ```

  ```
  /dev/sdb1: UUID="242c24d8-ac65-413d-b3a2-eb7f2f0993b0" TYPE="crypto_LUKS" PARTUUID="629e6177-01"
  ```

  To obtain the LABEL, use **lsblk** command;

  ```
  lsblk -f /dev/sdb1
  ```

  ```
  NAME
  FSTYPE      LABEL UUID
  FSAVAIL FSUSE% MOUNTPOINT
  sdb1
  crypto_LUKS       242c24d8-ac65-413d-b3a2-
  eb7f2f0993b0
  └─luks-242c24d8-ac65-413d-b3a2-eb7f2f0993b0 ext4
  e940b45b-dbc8-4c40-aaa5-9acf9fcb2119
  ```

  Also, you can obtain the UUID using the command below;

  ```
  cryptsetup luksDump /dev/sdb1 | grep "UUID"
  ```

- **key file**: describes the file to use as a key for decrypting the data of the source device. Note that the passphrase must not be

followed by a newline character.

- **`options`**: describes the cryptsetup options associated with the encryption process. At minimum, the field should contain either the string luks respectively tcrypt or the cipher, hash and size options. Options are in the format: **`key=value [,key=value …]`**.

  Consult **`man crypttab`** for more information.

  Therefore, this is how our device entry looks on **`/etc/crypttab`** file.

```
luks-242c24d8-ac65-413d-b3a2-eb7f2f0993b0
UUID="242c24d8-ac65-413d-b3a2-eb7f2f0993b0"
/etc/.crypt-me luks
```

Update fstab file with Device information

Next, you need to update the **`/etc/fstab`** file with device information as well to define how to mount the LUKS device.

The entry in the **`/etc/fstab`** file should take the format;

```
<file system> <mount point>   <type>  <options>
<dump>  <pass>
```

```
/dev/mapper/luks-242c24d8-ac65-413d-b3a2-
eb7f2f0993b0 /mnt/luks-242c24d8 ext4 defaults 0 0
```

Make the changes accordingly.

Ensure the mount point exists.

Verify the mounting using the `mount` command before you can reboot your system. If all is well, you should see "**successfully mounted**" for your LUKS device.

```
mount -av

/                          : ignored
/mnt/luks-242c24d8         : successfully mounted
```

You can now reboot your system to confirm the same.

```
systemctl reboot
```

Once the reboot is done, check the mounting;

```
lsblk

NAME
MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda
8:0    0   15G  0 disk
├─sda1
8:1    0   13G  0 part  /
├─sda2
8:2    0    1K  0 part
└─sda5
8:5    0    2G  0 part  [SWAP]
sdb
8:16   0    4G  0 disk
└─sdb1
8:17   0    4G  0 part
  └─luks-242c24d8-ac65-413d-b3a2-eb7f2f0993b0
254:0    0    4G  0 crypt /mnt/luks-242c24d8
```

Or use df command.

```
df -hT

Filesystem     Type      Size  Used Avail Use%
Mounted on
```

```
udev            devtmpfs   984M     0  984M   0%
/dev
tmpfs           tmpfs      200M  3.1M  197M   2%
/run
/dev/sda1       ext4        13G  3.6G  8.5G  30% /
/dev/dm-0       ext4       3.9G   16M  3.7G   1%
/mnt/luks-242c24d8
```

That concludes our guide on how to automount LUKS encrypted device in Linux on system startup.

Other tutorials;

How to Use VeraCrypt on Command Line to Encrypt Drives on Ubuntu 18.04

How to Encrypt Files and Folders with eCryptFS on Ubuntu 18.04

Install and Setup VeraCrypt on Ubuntu 20.04