Creating an external encrypted RAID 1

Keeping a local backup of your data is an essential step in protecting yourself from any sort of data loss. One of the most efficient ways to do this, as disks keep getting larger and cheaper, is to use an external dock or enclosure that plugs into your system. I prefer a dock because they are typically easier to change drives and / or remove them for offline storage.

Linux makes this possible by combining several layers of technology:

```
LVM (disk mirrorset, i.e. RAID1. I'm using version 2.02.176(2) on Kubuntu
18.04.4 LTS)
-- cryptsetup (encryption)
---- main file system (I use XFS but anything you trust)
------ secondary file system (optional… I use NILFS2 for versioning)
```

Create RAID 1 (my disks are /dev/sdd and /dev/sde, 2Tb for this example)

```
pvcreate /dev/sd{d,e}
vgcreate demo /dev/sd{d,e}
lvcreate --type raid1 -l 100%FREE -n test2T demo
```

This will create a volume group called "demo", a logical volume called "test2T" that is type RAID1. You can type "lvs" to see the sync activity. You can also type "lvs --segments" to see or verify the lv type. Note, you can continue with the steps below and start using the volume while this initial sync is running. It will pick up where it left off but I would recommend letting it finish the initial sync before unmounting or use "--nosync" when creating the lv.

Create encryption (for example we'll an AES-256 cipher with SHA 256 hashes)

```
cryptsetup luksFormat --cipher=serpent-xts-plain64 --hash=sha256
--key-size=256 /dev/demo/test2T
```

You will be asked to verify that you want to overwrite any existing data and then you'll be asked for a passphrase. Once you do that, now you can "unlock" the volume. With something like this:

```
cryptsetup luksOpen /dev/demo/test2T test2T
```

This will create the mapper device to the encrypted volume at /dev/mapper/test2T which can be formatted with:

```
mkfs.xfs /dev/mapper/test2T
```

and mounted to /mnt/tmp with,

```
mount /dev/mapper/test2T /mnt/tmp
```

Let's take a look at "lsblk" to review.

```
sdd                                                 8:48   0   1.8T  0 disk
├──demo-test2T_rmeta_0                            253:8    0     4M  0 lvm
│  └──demo-test2T                                  253:12   0   1.8T  0 lvm
│    └──test2T                                     253:13   0   1.8T  0 crypt
/mnt/tmp
└──demo-test2T_rimage_0                           253:9    0   1.8T  0 lvm
  └──demo-test2T                                  253:12   0   1.8T  0 lvm
    └──test2T                                     253:13   0   1.8T  0 crypt
/mnt/tmp
sde                                                 8:64   0   1.8T  0 disk
├──demo-test2T_rmeta_1                            253:10   0     4M  0 lvm
│  └──demo-test2T                                  253:12   0   1.8T  0 lvm
│    └──test2T                                     253:13   0   1.8T  0 crypt
/mnt/tmp
└──demo-test2T_rimage_1                           253:11   0   1.8T  0 lvm
  └──demo-test2T                                  253:12   0   1.8T  0 lvm
    └──test2T                                     253:13   0   1.8T  0 crypt
/mnt/tmp
```

Here you see both physical disks, the vg, lv (which is encrypted) and mount points. The volume is ready to use now. A typical thing I do is dump my system regularly with:

```
xfsdump -p 300 -J - / | lzop > /mnt/tmp/desktop.xfz
```

Obviously .xfz is arbitrary but I use that instead of .xfs to indicate that I have a compressed filesystem (I like lzo but you can use gzip, bzip, etc.). This is not necessary, especially with drive sizes being so large now. However, since I keep versions of my system backups and some data that is static, I want to maximize how much I can keep in the secondary file system container. To create that you can do something like:

```
dd if=/dev/zero of=/mnt/tmp/archive.img bs=1 count=0 seek=1T
```

This creates a 1Tb sparse file. I don't do more than 50% of the volume as a rule but you could do whatever makes sense to you. It will only consume what is in use but no more than 1Tb. Let's format it:

```
mkfs.nilfs2 /mnt/tmp/archive.img
```

and mount it,

```
mount -o loop /mnt/tmp/archive.img /mnt/archive/
```

To review, here is a look at the relevant lines from “df -h”

```
/dev/mapper/test2T  1.9T  1.9G  1.9T   1% /mnt/tmp
/dev/loop0          1.0T   16M  973G   1% /mnt/archive
```

I’m not going to get into the use of NILFS2 here but you can read up on it at:

https://www.kernel.org/doc/html/latest/filesystems/nilfs2.html
or
https://nilfs.sourceforge.io/en/index.html

This now has given me a dual ability to store versioned and non-versioned data in an encrypted RAID1 volume. 😄

I want to point out that I typically use this as a degraded RAID1. That means that I do not normally have both disks online. I keep one in the dock (powered off when not in use) and the other off-line someplace else. I let them sync regularly which will happen when LVM detects the second drive- it doesn’t have to be mounted or unlocked. That reduces the operational hours on one of the mirrors. If something ever happens I still have the other drive.

I also strongly recommend using a 4 bay dock since that makes for an easier upgrade path. You use 3 bays… 1 for the original RAID1 and 2 for your new RAID1. If you only have a 2-bay you would have to create the new RAID1 and then just run one drive from each lv. Once all your data is copied, you can remove the old mirror and insert the other new RAID1 disk so it can sync.

I hope this information helps. It may seem like a lot of steps but to me, it is worth it.

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
Keith C. Perry, MS E.E.
Managing Member, DAO Technologies LLC
(O) +1.215.525.4165 x2033
(M) +1.215.432.5167
www.daotechnologies.com