How To Migrate to a full encrypted LVM system

Posted by gpall on Mon 28 Jan 2008 at 15:07

The point of this how-to is to describe the way to migrate to a full-encrypted LVM system (rootfs + data) (only the boot partition obviously stays unencrypted), either coming from an LVM system, either from a simple ext3 system. All you need is some kind of external storage.

It should be here noted that since the operations described below are not very trivial, this procedure should only be followed by people somewhat experienced.

PART I - Installing neccessary software and saving the current system

There are two ways to implement the LVM full encrypted system:
a) the debian way (TM), which has the LVM on an encrypted virtual device (cryptsetup)
b) the other way, which first makes an LVM on your physical devices, and then encrypts the logical volumes

I preferred the debian way of course, although I don't find any specific reason for that.

My initial setup had /dev/hda1 as NTFS Windows and the rest, you don't care, because we won't be keeping any of the partition/LVM structure.

Before you attempt any of the following, do yourself a favour and take a full disk backup using CloneZilla, to some external storage. This way, when you screw up, you won't lose your smile.

To begin with, install 'cryptsetup'. Also, your kernel should include a initrd image to boot with. If not, install package initramfs-tools. Then create the following file inside /etc/initramfs-tools/conf.d
filename: **cryptroot**
contents:
target=lukspace,source=/dev/hda3,key=none,lvm=vg-root

After, that run a:
# **update-initramfs -u**
This builds an initrd image which knows where to look for the encrypted partition that we will make.

Second, and this is mandatory, take a tar of your system to this external storage using:

# **tar cSjf /my/external/storage/sysbackup.tar.bz2 /bin/ /boot/ /etc/ /home/ /lib/ /opt/ /root/ /sbin/ /selinux/ /srv/ /usr/ /var/**
(if you prefer speed, you can omit 'j' from tar (and .bz2 of course))

This will save all necessary permissions, files, ownerships etc etc. As you can see I have omitted the following directories: /dev/, /media, /mnt, /proc, /sys, /tmp. These will have to be recreated on the final system (except /media).

My suggestion is not to use the system while doing the backup, i.e. no X-window sessions, and no other consoles other than root's.

Now, time to get all your data to the external storage. Let's say they reside on a partition mounted on /media/abyss. Of course, /media also has cdrom and other useful things.

So:
# tar cSjf /my/external/storage/databackup.tar.bz2 /media/cdrom /media/whatever /media/...
everything EXCEPT the mountpoint of your external storage!!!

OK, so at this point you can take a breath. Your system, if you have not done something unclever, is secured against any kind of misconfiguration and child-eating experiments.

PART II - Reformating disk and creating the encrypted LVM structures

Remember again that if at some point, things go bad, you can always use clonezilla to restore your old system.

Now that your system is secured let's play around!

I use the clonezilla CD for most of the jobs as live CD.

So, boot Clonezilla, and give it network. Then enter a console prompt (sudo su) and...

```
# aptitude update && aptitude dist-upgrade
```

Before that, you may need something like that:

```
# ifconfig eth0 up
# dhclient eth0
```

After sometime your live system will have all the latest & greatest debian software so time to install a couple of tools:

```
# aptitude install cryptsetup joe
```

Joe is my favourite editor, so for the rest of the how-to we'll pretend it is yours too.

First, let's destroy your disk!

```
# cfdisk /dev/hda
```

and delete all partitions except your /dev/hda1 (NTFS).

Then create a primary partition that will be used for booting: /dev/hda2, with a size of 200 MB.

Then create another primary partition with the rest of the free space (/dev/hda3). This will be your future rootfs and data home.

Then write the partition table to the disk and exit.

This whole fuss just rendered your operating system inoperable and your data a bit crappy. But don't ya worry, we got everything on tar!

We want now to utterly destroy everything on the disk and fill it with random little bits, while at the same time checking for badblocks. Here we go:

```
# badblocks -s -w -t random -v /dev/hda2
# badblocks -s -w -t random -v /dev/hda3
```

these two commands will utterly destroy all data on your disk. They will need some time though to do that, so be patient.

After that, let's first give a filesystem to the boot partition. Easy:

```
# mkfs.ext3 /dev/hda2
```

Then we will use cryptsetup to make the big partition a dm-crypt device. This is also as easy as stealing from children.

```
# cryptsetup -y -s 256 -c aes-cbc-essiv:sha256 luksFormat /dev/hda3
```

I may not have the options in the correct order but this will get you going. BEWARE! Give a good passphrase that you will also remember. You cannot change this passphrase afterwards!!!

Now, time to use this initialized space:

```
# cryptsetup luksOpen /dev/hda3 lukspace
```

This will ask you for the previous passphrase. After that, you will see a /dev/mapper/lukspace appearing. This can be now regarded as a device, ready to accept any kind of data. You can format it with ext3, or you can build an LVM on it, which is what we will now do.

So, let's create the LVM.

```
# pvcreate /dev/mapper/lukspace
# vgcreate vg /dev/mapper/lukspace
# lvcreate -L10G -nroot vg
# lvcreate -l 100%FREE -nmyspace vg
```

So, we now created the LVM logical volume /dev/vg/root which will host the root filesystem and /dev/vg/myspace which will hold our data! That wasn't difficult, was it?

PART III - Restoring system files and booting the new system

After we've created the space for our root filesystem, time to untar our system to its new space.

First of all, we create a mountpoint, say /media/rootfs and we...

```
# mount /dev/vg/root /media/rootfs
```

and now...

```
# cd /media/rootfs
# tar xSjf /external_storage/sysbackup.tar.bz2
```

and wait for it to finish.

Next, create the directories which are needed but were not backuped:

```
# mkdir proc sys media dev tmp mnt
# chmod 1777 tmp
```

Now create elsewhere a bootfs mountpoint and mount our future boot partition:

```
#mkdir bootfs
# mount /dev/hda2 bootfs
```

Then move the boot files from rootfs/boot to this new bootfs place. So, now the /dev/hda2 boot partition has the necessary files to boot.

Lastly, change your fstab file in rootfs/etc so that it points to your new root device: /dev/vg/root.

Pray, and reboot! Be prepared though! After grub has loaded, press 'e' for edit at the kernel option and set the correct root=... option so that the machine can boot!

Hopefully, LUKS will ask you for your passphrase and everything will go just fine!

After you enjoy a couple of minutes playing with your all encrypted system, edit /boot/grub/menu.lst to point to the correct root=... device, and run an update-grub.

Giorgos Pallas

PS. If you have come that far, you don't need me to tell you how to restore from the tar your data files!
