Author: gargamel
Date: 2009-01-24
Version: 0.01
Status: Release Candidate 1

Installing Slackware 12.2 with encrytped root on LVM and RAID-1

Conditions of use

Table of contents

Foreword

Before you read on or start following the steps described below, you must accept that data stored on the discs you use for it will unavoidably be lost and cannot be restored. This is true, regardless, if the procedure described works for you or fails. **So backup your data NOW!**
I also want to make clear, that I am not guaranteeing anything, apart from a complete loss of data. The following may or may not work for you, and all responsibility for potential damages and your data is completely on your side! So you have a back-up of your data? Well, **you have been warned...**
Most of the following has been borrowed from the standard documentation that comes with Slackware 12.2 and supporting contributions by various well-known LQ.org and Slackware team members, like Alien Bob. This article contains nothing that hasn't been said already elsewhere, but I am trying to bring all this information in a concise, readable form, hoping that it has some benefit for some of you outthere. This is my way of saying thank you to all the people (and if I should name one, it would be Eric Hameleers, again) who helped me get this working on my systems by holding my hand and even providing patches just about immediately after my problems were tracked down.
Now, that you accepted, that I am not responsible for anything, and in particular not for the correctness of the following (although I am trying my very best, of course), let's start.

1. Pre-requisites

You have an installation medium with Slackware 12.2 and you have already downloaded the updated mkinitrd 1.3.3 package and copied it to a removable medium. With mkinitrd 1.3.2 shipped with Slackware 12.2 the following won't work!

2. Partitioning and RAID-1

We assume that we are installing Slackware 12.2 on a system with two identical harddiscs. Each disc has three partitions, one for swap, one for /boot and one for everything else. Let's setup two RAID-1 arrays: /dev/md0 is for /boot, and /dev/md1 for the rest. Swap is not part of a RAID array, as mirroring would slow it down significantly and striping is already a swap partition feature of modern Linux systems.
Now boot with your installation medium (usually this means to insert CD1 or the DVD) and boot your system with it. Login as root, when the prompt asks you to do so.

2.1 Partitioning

Logged in as root to the installation system, use cfdisk (or fdisk, if you prefer) to setup the following partitioning scheme for the two harddiscs. Note that on some systems device names may be different, such as /dev/sda and /dev/sdb, but you will have no problem to translate the following to your environment. Both harddiscs will be partitioned identically for RAID-1 mirroring.

/dev/hda

```
hda1        Linux swap               2GB
hda2        Linux raid autodetect    128MB
hda3 Boot   Linux raid autodetect    78GB
```

/dev/hdb

```
hdb1        Linux swap               2GB
hdb2        Linux raid autodetect    128MB
hdb3 Boot   Linux raid autodetect    78GB
```

2.2 RAID-1

After partitioning we'll start the two RAID-1 arrays. We will use /dev/md0 for the small partitions reserved for /boot and /dev/md1 for / and /home etc.

```
# mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/hd[ab]2
# mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/hd[ab]3
```

The syncing of the two discs may take quite a while depending on the size of your harddiscs and partitions. So brew yourself some fresh Java, or have a good cup of tea. And some cookies. Or lunch. Or keep a diet, but don't just sit in front of your screen waiting until the process finishes, and don't interrupt it, either.

2.3 Encryption with LUKS (Linux Unified Key Setup)

Now that RAID-1 mirroring is running, we can continue setting up LVM and harddisc encryption following the chapter *Combining LUKS and LVM* in README_CRYPT.TXT (on Slackware 12.2 CD 1 or DVD or, e. g., here: README_CRYPT.TXT).

```
# dd if=/dev/urandom of=/dev/md1
```

Again, be patient: Depending on your hardware and partition sizes, the first step, filling the file system with random content using dd can take many hours. This is normal, see e. g.: RootCryptoraid (although the author uses shred instead of dd). The good news is, that we (1) have to do this only once and (2) it gives added security and makes root encryption really useful. So brew yourself some fresh Java, or have a good cup of tea. And some cookies. Or lunch. Or keep a diet, but don't just sit in front of your screen waiting until the process finishes, and don't interrupt it, either. (Hmm, somehow I feel I have heard this before...)

```
# cryptsetup -s 256 -y luksFormat /dev/md1
```

It is highly recommended that you write down the passphrase you specify here and keep the paper at a hidden, but easily accessible place. If you forget your passphrase you won't get access to your system, anymore. At least, make a note on a tissue!
Note, that we apply encryption to the RAID device /dev/md1, not to an ordinary harddisc partition. And do NEVER encrypt a partition reserved for /boot, as your system won't boot then. That's why we did not encrypt /dev/md0, here. Once the encrypted filesystem is set up, we need to open it:

```
# cryptsetup luksOpen /dev/md1 slackluks
```

After entering the just defined LUKS passphrase we can access the filesystem.

2.4 Logical Volume Management (LVM)

Now that we have a encrypted file system running upon a RAID-1 array we continue following the instructions in README_CRYPT.TXT:

```
# pvcreate /dev/mapper/slackluks
# vgcreate -s 32M cryptvg /dev/mapper/slackluks
# lvcreate -L 8G -n root cryptvg
# lvcreate -L 30G -n home cryptvg
```

We did not create a logical volume for swap, as I prefer not to put swap under LVM control. The next step is to identify and activate the logical volumes just created.

```
# vgscan --mknodes
# vgchange -ay
```

Again I skipped the mkswap step described in README_CRYPT.TXT, because it isn't necessary, when swap is not under LVM control. Then I ran setup, and selected the mountpoints for /, /boot and /home:

```
/dev/cryptvg/root  /
/dev/cryptvg/home  /home
/dev/md0           /boot
```

The only difference compared to a „conventional“ installation are the device names. Note that

we use the LUKS device names instead of /dev/md1 for / and /home, but the RAID-1 device name for /boot. A mistake here can throw you back to square 1, so be careful, as repeating some of the steps is time-consuming.
Once the package installation completes, the system will ask you to configure and install the bootloader LILO. At this point it doesn't matter, if you choose „simple“ or „expert“ installation, as we will modify the LILO configuration later, anyway, but it is important that you install to the MBR. LILO is then installed to the MBRs of both RAID-1 discs /dev/hda and /dev/hdb, according to the screen messages. By the way: Despite all the comfort of more „modern“ boot loaders like GRUB, to my knowledge this is one advantage of LILO. You have to install GRUB to the MBRs of all RAID devices by hand yourself, if you prefer this bootloader. If you see messages with different device names, such as /dev/md0 or /dev/md1, you probably chose the wrong device names for your mountpoints, see above.

2.5 Generic kernel and initial RAM disk

When the the installation completes, we select EXIT. Next, we replace the huge SMP kernel used for the system installation with the generic SMP kernel, as recommended in the Slackware documentation. As we have an encrypted root file system, we need support for LUKS at boot time. As the encrypted file system is running upon a RAID-1 array, we also need RAID support. Finally, we add international keyboard mapping support, in this case for a German keyboard. We need this, in order to be able to enter our LUKS passphrase, when asked for it at boot time. For the generic SMP kernel we need to make support for these features available as modules at boot time, i. e. Before the actual system is up and running and all system programs and directories, such as /lib/modules are accessible. Therefore we need to create an initial RAM disk, an initrd. Again, the precise steps for this are documented in README_CRYPT.TXT. As all the mounting of directories has already be done by the setup program, we can just enter a change root environment and then issue mkinitrd with some options:

```
# chroot /mnt
```

Usually we would now just issue the command mkinitrd with some options, but stop! The Slackware 12.2 installatin media ship with mkinitrd 1.3.2. This version has some bugs, so we need to upgrade the mkinitrd package to version 1.3.3 or above. It's available from the usual Slackware mirrors. The easiest way to get it, is searching for mkinitrd with the official Slackware package browser.

Insert a USB stick, or whatever removable medium you have stored the update package on, and mount it. Then upgrade the package. Assuming that your USB device is identified by the system as /dev/sda1, this is all you have to do:

```
# mount /dev/sda1 /media/memory
# cd /media/memory
# upgradepkg mkinitrd*.tgz
```

Now, with a working mkinitrd installed, we can create the inital RAM disk.

```
# mkinitrd -c -k 2.6.27.7-smp -m ext3 -f ext3 -r /dev/cryptvg/root -C
/dev/md1 -L -R -l de-latin1-nodeadkeys
```

Here you see a few differences in my command compared to the one specified in README_CRYPT.TXT. First of all, I again replaced the ordinary harddisc device with the RAID device name /dev/md1. Secondly, I added `-R` for RAID support, as recommended in README_RAID.TXT, and `-l de-latin1-nodeadkeys` for German keyboard support.

As you see, this is a rather lengthy command. When things go wrong, we may find ourselves in a situation, where we have to specify it more than once. This is tedious and error-prone. I therefore recommend that, instead of the command above, you just issue a simpler variant. All you need to do for this, is to edit or create a configuration file:

/etc/mkinitrd.conf

```
SOURCE_TREE="/boot/initrd-tree"
CLEAR_TREE="0"
OUTPUT_IMAGE="/boot/initrd.gz"
KERNEL_VERSION="2.6.27.7-smp"
KEYMAP="de-latin1-nodeadkeys"
MODULE_LIST="ext3:mbcache:jbd:uhci-hcd:usbhid"
LUKSDEV="/dev/md1"
ROOTDEV="/dev/cryptvg/root"
ROOTDEV="ext3"
RAID="1"
LVM="1"
WAIT="1"
```

Now, a simple command like

```
# mkinitrd -F
```

will do. This command reads the configuration and does the same things as the much more complicated one-liner shown above. If you look at the configuration file closely, you may notice some entries in the module list not specified in the original command. Probably they aren't needed, but I am paranoic. The modules mbcache and jbd usually are added to the initrd automatically when you add support ext3 to it, and uhci-hcd and usbhid are for the support of USB keyboards.

Before we proceed, we verify our mkinitrd options with a most helpful little script provided by Eric Hameleers. Download mkinitrd_command_generator.sh and run it like this on your system:

```
mkinitrd_command_generator.sh -c
```

Compare the output with your previously posted mkinitrd.conf. Or try

```
mkinitrd_command_generator.sh -i -c
```

if you want to build your mkinitrd.conf file interactively. Don't be confused if some device names in the output slightly differ from what you have in /etc/mkinitrd.conf. For example, if you have defined LUKSDEV=/dev/md1 and the script output is LUKSDEV=/dev/md/1, this is equivalent. Also it shouldn't worry you, if you get to see /dev/mapper/cryptvg-root instead of /dev/cryptvg/root, as the latter is a symlink to the first.

If all is well, we follow the instructions in README_RAID.TXT. To switch to the generic kernel we redefine some symlinks in /boot:

```
# cd /boot
# ln -sf vmlinuz-generic-smp-2.6.27.7-smp vmlinuz
# ln -sf System.map-generic-smp-2.6.27.7-smp System.map
# ln -sf config-generic-smp-2.6.27.7-smp config
```

2.6 LILO

Hold on, we are almost there! However, in order to boot with the generic SMP kernel we need to make our new initial RAM disk accessible to it and ensure that LILO finds the MBR on the RAID disks. To this end we need to modify /etc/lilo.conf (only relevant or modified lines are shown, leave everything else untouched):

/etc/lilo.conf

```
boot = /dev/md0
raid-extra-boot = mbr-only
image = /boot/vmlinuz
  initrd = /boot/initrd.gz
  root = /dev/cryptvg/root
  label = linux
  read-only
```

After writing the file to disc, we must not forget to run lilo, otherwise we will be facing a kernel panic.

```
# lilo
```

This command will give a couple of warnings that can be ignored, as mentioned in README_CRYPT.TXT. If we don't get an error, we are ready to reboot. At some point in the boot process, we will be asked for our LUKS passphrase. You wrote that down, didn't you? Ok, then you can simply type it in and hope for the best.

3. Troubleshooting

If something goes wrong, you need to get access to the installed system. But how, you may ask, do I get access to an encryted mirroring system partition? Well, it's no wizardry, but, of course, it is not as comfortable as you might know it. These are the steps:

1. Boot with your installation medium

2. Log into the installation as root, when prompted

3. Start RAID, i. e. scan for, and then assemble the RAID arrays

```
# mdadm -Es > /etc/mdadm.conf
# mdadm -As
```

4. Activate the logical volumes

```
# vgscan --mknodes
# vgchange -ay
```

5. Open the LUKS device

```
# cryptsetup luksOpen /dev/md1 slackluks
```

6. Mount root „partition“

```
# mount /dev/cryptvg/root /mnt
```

7. Switch to installed OS

```
# chroot /mnt
```

8. Mount remaining filesystems

```
# mount /boot
# mount /proc
# mount sys /sys -t sysfs
```

At this point, you can bring up your favorite editor, tweak config files, re-run mkinitrd/lilo/etc as you wish, or anything else you need to do to make your system bootable again. When you're finished making your changes, rebooting is simple:

```
# cd /
# umount boot proc sys
# exit
# reboot
```

References

Most of this text has bee „borrowed“ from other sources, partially word by word, in the hope that the respective authors won't mind, as I give them credits.

Official Slackware documentation, to be found on the installation media (DVD or CD 1):

- Slackware RAID HOWTO by Amritpal Bath (README_RAID.TXT)
- Installing Slackware on Logical volumes by Eric Hameleers (README_LVM.TXT)
- Installing Slackware on encrypted volumes (README_CRYPT.TXT) by Eric Hameleers
- Slackware initrd mini HOWTO (README_INITRD.TXT) by Patrick Volkerding

Related threads at http://www.linuxquestions.org:

- Slackware-12.2: RAID-1 + LVM + LUKS + encrypted root

Thanks to all contributors to this and all the other threads where one or some of you have saved my day a hundred times (at least). The Slackware community is so great!

Other useful resources:

- Disk encryption in Fedora: Past, present and future (Red Hat Magazine) by Michael Petullo
- SW-RAID und LVM (Grundlagen) by anniyka (German)
- SW-RAID und LVM (Howto) by anniyka (German)
- Encrypted Storage with LUKS, RAID and LVM2 by René Pfeiffer
- Setup with Software Raid and Cryptofs on whole system by nextime

I hope I credited all contributors to this miniHOWTO appropriately. If, however, I should have forgotten someone, please notify me. Also, if you are the author of some of the referenced documents and don't like to be quoted here, please let me know. I will, of course, respect your request, up to the point that I will redraw this document, if this should be the only reasonable solution (of course, I hope we find a better alternative)! Impressum