CLI Cookbook

Lucas Westermann with E. Frank Sandig
and other readers of Command & Conquer

September 8, 2013

Note: Each command in a longer list is separated by a blank line. All commands on more than one line without a blank line are simply wrapped text - run it as one command!
You can find this on GitHub: `https://github.com/lswest/cli-cookbook`
If you want to contribute - feel free to clone the repository, or open an issue (if you aren't comfortable with LaTeX)

Contents

1 Get information on commands

1.1 Get a brief summary of bash commands

```
whatis command
```

Replace command with the name of the command you're curious about, and whatis should return a brief summary of what it does (I say should, as not all programs register themselves in the whatis database).

1.2 Get the location of a command's binary

```
whereis command
```

```
which command
```

Whereis and which both return the path to the main executable of the command.

2 Finding and managing files

2.1 sed

```
sed
Example: sed 's/[pP][iI][nN][kK] [fF][lL][oO][yY][dD]/Pink Floyd/g'
```

Sed can be used for word replacement in files. The example supplied will consolidate all spelling variations of Pink Floyd. It's also frequently used to unify file types (i.e. .JPG .jpeg .jpg into one uniform extension).

2.2 ls

```
ls -a|grep $searchterm
Note: Replace $searchterm with the term you're looking for.
```

This combination has ls list all files (including hidden files and directories), and passes the resulting list to grep, which then filters it according to your search term. The manpage of grep will give you the syntax to negate searches ("find everything that doesn't contain X").

2.3 convert command

```
convert
Example: convert *.jpg output.pdf
```

Convert is a powerful command-line tool that helps to convert between formats. However, it can also be used to merge files together. The example above takes all files that end in .jpg and merges them into a single PDF. The reader who supplied this notes that they find it most useful when scanning large numbers of files.

2.4 Search with updatedb and locate

```
sudo updatedb # updates database

locate search term
```

The locate command quickly searches the database for items matching the search terms supplied. It won't always find the file you're looking for, but is faster than find if you don't know what folder it's in.

2.5 Search with find

```
find $path | grep -Ri "search term"

find $path  -name "search term"

find $path -iname "search term"
```

Replace **search term** with your search terms. Find will also search from the specified path onwards (for current path, just use '**.**' minus the quotes). I wouldn't recommend using find to search through your directory from root ('/'), as it could take forever. Much better to use locate there. The only difference between -name and -iname is that -iname is case insensitive, which means .JPG and .jpg are treated as the same text.

2.6 Search every file recursively

```
find . -type f |xargs grep ''text to search'
```

The find command searches all files (-type f) in the current directory and below, using the search terms supplied to grep and parsed through xargs.

2.7 Convert files using LibreOffice

```
soffice --headless --nologo --convert-to odp *.ppt
# set source and destination formats as desire
```

Similar to the convert command, this can convert between file formats (any formats you can use/create in LibreOffice). The first two arguments are to hide the LibreOffice interface, and the third argument tells it what to do. The reader who supplied this command mentioned that they found it useful for converting the PowerPoint slides they received from their professors to PDF or ODP. However, converting the latest Microsoft formats (.docx, .pptx, etc.) can/will result in formatting issues in LibreOffice 3.5 in Ubuntu 12.04, as well as in LibreOffice 4.1 in Ubuntu 13.04. Ubuntu 12.10 has not been tested yet.

3 Packages

3.1 Debian based systems

All the following commands are for Debian based systems (which is what Ubuntu is). In the case of trusted keys - not all Debian systems have them.

3.1.1 Remove leftover configuration files from removed packages

```
sudo dpkg --purge `dpkg -l | grep ^rc | awk '{print $2}'`
```

3.1.2 Create a detailed list of installed packages

```
COLUMNS=200 dpkg-query -l > packages_list.list
```

COLUMNS is a variable being passed to dpkg-query, and isn't a mandatory part of the command. In this case, however, it helps to format the results for the resulting text file. It essentially queries dpkg for a list of installed packages. The resulting list will contain package names, versions and descriptions. Good for looking up what's installed and what is done by which package. Alphabetical order.

3.1.3 Create a list of installed packages for restoring your system

```
dpkg --get-selections | awk '!/deinstall|purge|hold/ {print $1}' > packages.list
```

This list contains only package names of installed packages. One per line. It is therefore suitable for re-installing all packages from the command line (see below). Alphabetical order.

3.1.4 Create a list of automatically installed packages

```
apt-mark showauto > package-states-auto
```

This command generates a list containing the names of all packages that have been installed automatically due to dependency reasons, and saves it to the file package-states-auto. This is needed for restoring an installation properly.

3.1.5 Restore a list of packages

```
dpkg --get-selections | awk '!/deinstall|purge|hold/ {print $1}' > packages.list
```

3.1.6 Save all software sources to one file

```
find /etc/apt/sources.list -type f -name '*.list' -exec bash -c 'echo
-e "\ n## $1 ";grep "^[[:space:]]*[^#[:space:]]" $1' _ {} \;
> sources.list.save
```

This long command finds all entries in sources.list and the sub-directories, putting the entire list in a file called sources.list.save.

3.1.7 Save the keys of trusted software sources

```
sudo cp /etc/apt/trusted.gpg trusted-keys.gpg
```

This command simply makes a copy of the trusted key list (stored under /etc/apt/) and copies it to the current directory.

3.1.8 Restore installation

```
# firstly, restore sources manually!
sudo apt-key add trusted-keys.gpg  # import keyring

sudo apt-get update # update sources

xargs -a "packages.list" sudo apt-get install  # install software

xargs -a "package-states-auto" sudo apt-mark auto # set package state
```

The way I would recommend restoring the software sources is to copy and paste sources not already present in the existing sources.list file. It can be tedious, but can avoid issues with different versions and installations. The

reader who supplied this said that they apply the above procedure to save time when re-installing a system. Or when they have to create a few identical installations.

3.2 Archlinux

3.2.1 Create list of installed packages

```
pacman -Qqe | grep -v "$(pacman -Qqm)" > pkglist-off.txt

pacman -Qqm > pkglist-loc.txt
```

The first command creates a list of all officially installed packages (i.e. nothing from the AUR). The second command creates a list of all locally installed packages. If you use an AUR manager, copy the install package over along with the backup list - or else make note of the URL so you can wget it later.

3.2.2 Create a local copy of installed packages

```
cp /var/cache/pacman/pkg/* $backup-location

sudo pacman -Sc
```

The first command will copy the local cached copies of installed packages. However, pacman stores old copies too, so the second command will remove all the old copies, in case space is an issue. Most AUR managers offer a similar option of caching files (not necessarily enabled by default).

3.2.3 Restoring package installations

1. Install Arch as you would usually from a live CD using the AIF (Arch Installation Framework).

2. Mount backup device (in live CD)

   ```
   mkdir /backup-files

   mount /dev/<disk-drive-partition>

   /backup-files
   ```

3. Now copy these to the newly created Arch install (mounted under /mnt)

```
mkdir -p /mnt/opt/restore

cd /backup-files

cp -a * /mnt/opt/restore
```

4. Now chroot (change root) to the new Arch install

```
cd /mnt

cp /etc/resolv.conf /mnt/etc

mount -t proc none /mnt/arch/proc

mount -t sysfs none /mnt/arch/sys

mount -o bind /dev /mnt/arch/dev

chroot . /bin/bash
```

5. OPTIONAL: If you made a local copy of the cache, copy the cache:

```
cp /mnt/opt/restore/pkg/ /var/cache/pacman/pkg/
```

6. Now install the official packages into the new system

```
pacman -Sy

pacman -S --needed $(cat /opt/restore/pkglist-off.txt)
```

7. If you have AUR packages installed, install the AUR helper, and then do the following

```
$AURhelper -S $(cat /opt/restore/pkglist-loc.txt | grep -vx
"$(pacman -Qqm)")
# Replace $AURhelper with the name of your preferred helper
```

   The above command installs all packages listed in the local files backup from 3.2.1 .

This long process will result in having your old packages installed into a new system. This can help make the occasional re-install infinitely easier. Or, if you have a system you're completely satisfied with, you can then move it to a new system. The ArchWiki has plenty of information for those of you trying to move more than just the installed packages.

4 Hard drive commands

4.1 Full list of partitions

```
sudo fdisk -l # for legacy BIOS

sudo gdisk -l # for GPT (UEFI) systems
```

These commands will print a list of all hard drives and their partitioning structure. If you're unsure of which command to use, fdisk is probably for you. Don't worry - no harm running the wrong command, and it will prompt you that you ought to be using the other command.

4.2 Back up MBR (Master Boot Record)

```
sudo dd if=/dev/hda of=mbr_backup bs=512 count=1
```

This command runs dd on the "input file" (if) of /dev/hda (replace this with the disk you want to back up the MBR of most likely /dev/sda). It then reads the first 512 bits (bs=512 size count=1 - how many chunks of size 512 to read) and saves them in the "output file" (of) mbr_backup. If you're running this from a live CD or disk, remember to save/move the output file to a persistent drive (i.e. hard drive, USB stick, etc.). Otherwise it will be lost the moment you reboot.

4.3 Back up boot code

```
dd if=/dev/hda of=bootcode_backup bs=446 count=1
```

Similar as with the MBR command, this command backs up a section of the hard drive. In this case, it backs up only the section of the hard drive that contains boot code.

4.4 Display detailed information on your Hard Disk

```
sudo hdparm -I /dev/sdx
```

This command lists thorough information about the supplied hard disk (replace /dev/sdx with your actual drive).

4.5 Correct partitioning order

According to the reader who submitted this: "*Due to using gparted over and over again for my external 2Gb harddisk my partitioning was off (first p2, then p3, p1 was the extended, in there p7, then p5, then p6). I corrected this by doing:*"

```
sudo sfdisk -d /dev/sdb > sdb.old

sudo cp sdb.old sdb.new

sudo gedit sdb.new

# Change order, don't forget p4 which did not exist
sudo sfdisk /dev/sdb < sdb.new

If it's wrong, do the following:
sudo sfdisk /dev/sdb < sdb.old
Then retry the gedit step.
```

I'm honestly of two minds about this suggestion. On one hand, I can accept doing this on usb sticks - though I don't really think you'll need more than one partition on any but the largest. Doing this on any kind of actual hard drive seems extremely dangerous as countless things could go wrong. An out of sequence partition table isn't a problem in any way, so only risk this ***if you can't possibly live with it***

4.6 Fill hard disk with zeros

```
BE CAREFUL!!!
dd if=/dev/zero of=/dev/sdb bs=64k
BE CAREFUL!!!
```

This command can be extremely dangerous to use if you're not sure what you're doing. It writes to the disk and fills it with zeros, effectively wiping the hard drive. For the security-conscious of you, doing this a couple of times on a drive before throwing it away/giving it away, it will make it much more difficult to restore data from the drive. It's best used in conjunction with the following command. Be sure to replace /sda/sdb with the correct path.

4.7 Fill hard drive with random data

```
BE CAREFUL!!!
dd if=/dev/urandom of=/dev/sdb
BE CAREFUL!!!
```

This command is extremely dangerous to use blindly without understanding what you're doing. This is because, similar to the previous command, it fills the hard drive with data. Instead of zeros, it instead uses random data. The reader who submitted this code mentions as well that they do this with old drives before selling them. They also mention that their preference is to re-format them as **NTFS** at the end as well. Remember to replace /dev/sdb with the correct path.

4.8 Get UUIDs of your hard drive partitions

```
sudo blkid
```

blkid prints the UUIDs of your hard drive partitions. It can also be limited to one hard drive or one partition by just passing /dev/sda1, for instance, to blkid. This is useful if you had to reformat your /home or another partition or if you have wiped your swap and reformatted it using the dd command above and your partitioner of choice. After you know the UUIDs, you can put them in your fstab so the automount at startup works properly with the newly formatted partitions.

5 Password generation

5.1 Numeric passwords

```
makepasswd --string 0123456789 --chars 32
```

This command generates a single 32-character long numeric password.

5.2 Alphanumeric passwords with special characters

```
pwgen -sy 32 64
```

This command generates 64 passwords of a 32-character length that contains letters, numbers and special characters. The reason why we show how to generate multiple ones at first is because it can be useful when setting up a large number of networks at once. **pwgen is not installed by default**

6 wget

6.1 Make an offline copy of a website

Make an offline copy of a website

```
wget --user-agent="Mozilla/5.0 (X11; U; Linux i686; de; rv:1.9b5)
Gecko/2008050509 Firefox/3.0b5" -r -k -E -l 8 http://example.com
```

This command uses wget to impersonate Firefox's user agent, and then runs recursively through the website (-r defaults to 5 levels, and -l sets the depth, to 8 in this case). The -k switch tells wget to fix the links to make them suitable for local viewing, -E tells it to adjust extensions (i.e. change .asp to .html). Don't forget to change the URL!

6.2 Download a file (with the ability to continue)

```
wget -c http://example.com
```

wget grabs the supplied link and downloads the file - in the example it would download the index.html file, and if you give it a file link, it will download the file. The -c switch lets you continue a download if it's interrupted.

7 Display hardware information

```
lshw -C SECTION
Example: sudo lshw -C network
# lists all information about networking devices
Formatting example (HTML): sudo lshw -html > hardware.html
#creates an html page with the lshw results

lsusb
Example: sudo lsusb -v | more
# lists verbose output and drops it into the more tool for easy reading.

lspci -nn
# -nn prints also device and vendor id. Useful if you're looking
for a driver, e.g. for WLAN, using databases at ubuntuusers.de or other

dmesg | grep KEYWORD
```

```
ifconfig
```

```
iwconfig
```

```
dmidecode -t TYPENUMBER
```

Logically, lsusb prints information about USB devices, and lspci lists information about PCI devices (system bus). Lshw lists information about almost all hardware in your computer. Dmesg prints out system messages, ifconfig prints hardware information about ethernet and wireless interfaces, and iwconfig display information on wireless information (such as ESSID). Dmidecode dumps a computer's DMI (sometimes called SMBIOS) table contents into a human-readable format. SMBIOS stores specific information about the computer.

The name for the SECTION (called class by lshw) can be found by running either of the following commands:

```
lshw -businfo
```

```
lshw -short
```

8 View running processes

8.1 htop

```
htop
```

Htop is a more functional (I find) version of top. It's not always installed by default, but I have yet to find an official repository that doesn't offer the package.

8.2 View logged in users and processes they own

```
w
```

The command "w" lists all logged in users and the processes associated with those accounts.

9 Administration

9.1 Add user to group

```
usermod -aG 'groupname' 'user'
```

This command modifies a user and adds them to the group specific as groupname.

9.2 Create aliases in your shell

This applies to almost any shell, though the configuration file will be different. I will assume the use of the Bourne Again Shell (Bash) and therefore the file .bashrc.

```
vim ~/.bashrc

Then add this anywhere in the bashrc file:
alias $name="command"

This alias will then be active in any new shell you open.
If you want to use it in a currently open shell, run this:
source ~/.bashrc
```

After that, just type whatever the name of your alias was, and it will run the specified command. Remember to replace $name and command with the actual name you'd like the alias to have, and the command you want it to run.

9.3 Check who opened which ports

```
sudo netstat -lntup
```

This is a fairly straightforward command - it lists users and the ports the have opened.

10 Miscellaneous fixes

10.1 Upside down webcam in Skype

Before using this command, you should find the path of your v4l1compat.so file. For information on how to find it, see 2.4 (using updatedb and locate)

```
bash -c 'LD_PRELOAD=/usr/lib/i386-linux-gnu/libv4l/v4l1compat.so skype'

#if you're on 11.10, you'll need the following:
sh -c 'LD_PRELOAD=/usr/lib/i386-linux-gnu/libv4l/v4l1compat.so
/usr/bin/skype "$@"'
```

Since you most likely don't want to type this every time, you can alias it in your bashrc file (9.2), or else create a small bash script executable.

10.2 GRUB boots into black screen

Occasionally some graphics cards have issues booting into GRUB's framebuffer, which results in a black screen. If you run into this issue, you can try adding **nomodeset** to the end of your kernel boot line. GRUB2 and legacy GRUB have different ways to edit entries, but the bottom of the GRUB menu should have instructions.

10.3 Reinstall GRUB using chroot

Chroot lets you work on an Linux installation that is currently not bootable. This can happen when you set up a dual boot system and install Windows after Linux. Sometimes it happens that you have no other option than reinstalling Windows to fix it.
Firstly, boot into a live Linux, e.g. the latest Ubuntu live CD, then open the Terminal.

```
sudo mount /dev/sda2 /mnt

sudo mount -o bind /dev /mnt/dev

sudo mount -o bind /sys /mnt/sys

sudo mount -t proc /proc /mnt/proc

sudo chroot /mnt /bin/bash

grub-install /dev/sda

exit
```

Replace sda2 with your / partition. In Ubuntu live mode, sudo will not prompt for a password. This example works only if you have no seperate

/boot partition. If you have not located your /home on a seperate partition, it can happen that you are not the owner of your /home/user directory any more, which leads to problems. You can easily fix that using this command after booting into the installed Linux:

```
sudo chown -hR user:user /home/user
```

If you have a seperate /boot partition, the command list is as follows:

```
sudo mount /dev/sda2 /mnt

sudo mount /dev/sda3 /mnt/boot

sudo mount -o bind /dev /mnt/dev

sudo mount -o bind /sys /mnt/sys

sudo mount -t proc /proc /mnt/proc

sudo \codeHighlight{chroot} /mnt /bin/bash

cp /proc/mounts /etc/mtab

grub-install /dev/sda

exit
```

Again, you have to replace sda2 and sda3 with your actual partitions.

11 Video tips

11.1 ffmpeg

11.1.1 Record your desktop

If you don't mind using a pre-build tool, using recordmydesktop is probably easiest. However, if you want total control over the recording, you can use ffmpeg

```
ffmpeg -f alsa -ac 2 -i hw:0,0 -f x11grab -r 20 -s 1680x1050+0+0 -i :0.0
-acodec pcm_s16le -vcodec libx264 -vpre lossless_ultrafast -threads 0 -y
vidcap.mkv
```

11.1.2 Use ffmpeg to cut a section of video out

```
ffmpeg -ss 00:02:23 -i input.foo -t 25 -vcodec copy -acodec copy output.foo
```

This command tells ffmpeg that starting from 2 minutes and 23 seconds into the video, to copy out the following 25 seconds (-t 25) from input.foo into output.foo

11.2 Use vdpau (hardware accelerated decoding in mplayer)

Nvidia uses the vdpau library to offload video encoding and decoding to the graphics card. To use this in mplayer, do the following:

```
mplayer -vo vdpau -vc ffh264vdpau file
```

Replace the word "file" with the path to the video you want to play.

12 Encrypted thumbdrive installation

This chapter shows how to create a fully encrypted, persistent installation of Linux on a USB thumbdrive, using the command line a lot. This is a larger project, as it takes a lot of time. It is recommended that you already have some experience in normal installation and using the command line. The procedure below has been tested with Backtrack 5R3, Ubuntu 10.04 LTS and Backbox Linux 3.05, all 32bit. It should also work similarily with Ubuntu 12.04 LTS as well as Debian Wheezy and other distros, but this is still to be tested. In this tutorial I will assume you have a Debian based distro.

In Ubuntu, you can easily have encrypted /home and swap on your laptop by just checking one box during the installation process. But this tutorial takes paranoia to a new level: Firstly, everthing exept the /boot partition is encrypted. Secondly, the drive is removeable. So you even can put it in a safe. Should be even a bit more theft proof.

As Ubuntu recognizes new hardware at each startup, you can even use the thumbdrive we create below on several machines.

At last, I recommend not to create the encrypted drive on your productive machine. As we perform hard disk operations, you can easily break your system by misspelling a command. Use a spare computer or a less important one, if you have one.

12.1 What you need to start

1. A USB thumbdrive, at least 16GB. I recommend 32GB to have more space for your data, but of course you can use even larger ones if you need more space in your /home. 8GB may be sufficient for the installation of Ubuntu, for instance, but you quickly run out of space if you really use the stick, save some larger files and install some more software.

2. A live Linux, best is the Linux you want to install on the destination thumbdrive. You can use a distro on a second thumb drive (e.g. 2GB) or a DVD. Last time I created the encrypted boot stick I used Backbox Linux.

3. Working internet connection.

12.2 Steps to carry out

12.2.1 Boot your live Linux

Sometimes you have to select your CD/DVD drive or thumbdrive as primary boot device in your BIOS. Usually, this step is not much of a problem. The latest live distros recognize a lot of different hardware, so booting usually does not go wrong. If it does, you can always try another distro or some boot parameters. You can find help on those problems on the community pages of your distro.

12.2.2 Get the internet connection working

This is also not much of a problem with new distros. Operating systems like Ubuntu usually make it work out of the box. Wired connection will default to DHCP. For Wifi, you usually just have to select the network, hand your system the Key and click connect. For some newer Wifi cards it can be necessary to install e.g. the package linux-firmware-nonfree first or do something else. If any problems occur, the community of your chosen distro can help you.

12.2.3 Partition your thumbdrive

Pick your partitioner of choice. Gparted does a good job under Ubuntu, but of course it can be done with fdisk on the command line as well. The single steps with fdisk will follow in a later version of this document. You need one /boot partition, that is not going to be encrypted. I recommend creating a

partition of about 2GB and formatting it to ext3. The rest of the thumbdrive should be a logical partition in an extended one. You don't have to format that second partition now. We will create the encrypted volume there later. Make sure that the first partition is set active. I don't know how to do that with gparted, but it's easily done with fdisk as follows:

```
sudo fdisk /dev/sdb

a [enter]

1 [enter]

w [enter]
```

In this tutorial, sdb is the thumb drive. You may have to change that on your system. You get a good overview of your drives and partitions to find proper name in gparted or by the following command:

```
sudo fdisk -l
```

12.2.4 Install needed software

We need software for logical volume management (lvm) and encryption (luks). You get it installed with:

```
sudo apt-get update && sudo apt-get install hashalot cryptsetup lvm2
```

12.2.5 Fill the installation partition with random data

This step is not necessary, but levels up your paranoia skills. With an encrypted volume in a partition filled with random data, it is very hard to even tell where your data actually reside. This step will take some time, as some GBs of random data have to be written. We can accomplish it with our old friend dd as follows:

```
dd if=/dev/urandom of=/dev/sdb5
```

Due to sdb being our thumbdrive, the second partition is called sdb5 because it's the first logical one.

12.2.6 Set up encryption

The following command will set up encryption on our installation partition. I think the selected options are a good trade off between performance and security.

```
cryptsetup -y --cipher aes-xts-plain --key-size 512 luksFormat /dev/sdb5
```

Cryptsetup will ask if you are really sure and state that you will lose all data on the destination partition. If you are sure to have named the correct partition, type uppercase YES. After this, cryptsetup prompts two times for your passphrase. I recommend using a real phrase of at least five random words or even passwords you can remember instead of just a single password. Correct horse battery staple is already taken ;) After cryptsetup has set up encryption, we can open the encrypted partition for the further steps using:

```
cryptsetup luksOpen /dev/sdb5 pvcrypt
```

Cryptsetup will prompt for your passphrase.

12.2.7 Create logigal volume

The following commands set up the physical volume, volume group and our logical volume for the root partition. The last command sets up an ext4 file system on the newly created logical volume, which we will use later for the installation.

```
pvcreate /dev/mapper/pvcrypt

vgcreate vg /dev/mapper/pvcrypt

lvcreate -n root -l 100%FREE vg

mkfs.ext4 /dev/mapper/vg-root
```

12.2.8 Deactivate journaling

This step is also not necessary, but can increase file system performance a bit.

```
tune2fs -o journal_data_writeback /dev/mapper/vg-root

tune2fs -O ^has_journal /dev/mapper/vg-root

e2fsck -f /dev/mapper/vg-root
```

12.2.9 Install Linux

Now you can install Linux on your thumbdrive using the graphical installer provided with the live system. It is important that you select "something else" or "manual partitioning" when asked what to do by the installer. Then you have to select /dev/sdb1 as /boot partition and /mapper/vg-root as / partition (option "mount point"). And you have to select /dev/sdb ans destination drive for the boot loader. This is very important. If you don't select the proper drive you may break your system! When the installation process has finished, click on "continue testing". **Don't click on restart!** We still have some things to do in the live system to get the encrypted installation working.

12.2.10 Configure installed system to use lvm and encryption

Open a terminal window. We will now configure our new installation using the commands blkid and chroot we got to now above. First, we need the UUID of the encrypted volume. Execute the following and note the UUID:

```
sudo blkid
```

Next, we use chroot to configure the installed Linux to use lvm and encryption. The commands are as follows:

```
sudo mkdir /mnt/cryptOS

sudo mount /dev/mapper/vg-root /mnt/cryptOS

sudo mount /dev/sdb1 /mnt/cryptOS/boot

chroot /mnt/cryptOS

mount -t proc proc /proc

mount -t sysfs sys /sys

apt-get update

apt-get update && apt-get install hashalot cryptsetup lvm2
```

Finally, we have to create a file so that our Linux finds and uses the encrypted volume. Open your favourite console editor, e.g. vim or nano, and create the file /etc/crypttab. Do not leave the chroot environment! Save the following line to that file, containing the UUID you noted above:

```
pvcrypt /dev/disk/by-uuid/uuid-from-above ^none ^luks
```

At last, we have to check if our fstab contains the proper information; still being in the chroot environment. Open /etc/fstab in your favourite editor. The file should look like that:

```
#/etc/fstab: static file system information. #
                <file system> <mount point> <type> <options> <dump> <pass>
                 #proc                /proc              proc   defaults  0
                 #/dev/mapper/vg-root UUID=c8d9b9a0-2198-4966-bc3a-39259df6a2c2
                 /dev/sdb1 UUID=6af425ad-99b8-44a5-9ee1-0349141f9b1f /boot ext3
```

Of course, you will have different UUIDs. I recommend using noatime on the thumbdrive as mount option. This will again increase file system performance a bit. But still, the performance will not be as good as with a non-encrypted drive.

12.2.11 Boot into the incrypted sytem

Congratulations, you're done with the set up! You can now restart, select your thumbdrive as boot device and test the installtion. If the boot process seems to be stuck, check the ttys (Alt+F1 and so on). One will prompt for your luks passphrase. After the system is up, you can install your favourite software, e.g. gnunet, i2p, tor and other stuff you need and start using your fully encrypted sytem. Have fun!

13 Useful Links

- http://www.linuxcommand.org
- http://www.commandlinefu.com
- http://www.shell-fu.org
- http://www.ss64.com