Arch Linux Full Disk Encryption



The **Paranoia** paragraphs haven't been finished just yet.

Full Disk Encryption is probable one of the most important things to do first, when setting up a new system in a world in which #BigBrother is always watching you. The issue we had was, having a keyfile which is needed to decrypt your system is nice, but if its [the keyfile] unencrypted on a USB device it doesn't satisfy our paranoia. So the solution is to encrypt to USB device as well; with a passphrase. And that's what we're going to show here.

Requirements

- Arch Linux Download
- Computer

USB Device Installation

Write the ISO to a removable flash drive. You can also use the traditional way and simply burn the ISO on a CD/DVD.

<sxh bash;> ~$: shasum archlinux-$VERSION-dual.iso ~$: dd if=archlinux-$VERSION-dual.iso of=/dev/$DEVICE bs=8192 </sxh>

Booting

<sxh bash;> # if necessary reconfigure your keyboard layout root@archiso ~ # loadkeys fr # check for network connectivity root@archiso ~ # ping 8.8.8.8 # request IP address root@archiso ~ # ifconfig -a root@archiso ~ # dhclient $NIC </sxh>

tmpfs (Paranoia)

<sxh> root@archiso ~ # fdisk -l | grep Disk root@archiso ~ # mkdir ./mytmpfs root@archiso ~ # mount tmpfs ./mytmpfs -t tmpfs -o size=32m root@archiso ~ # cd ./mytmpfs root@archiso ~ # dd if=/dev/urandom of=secretkey bs=1024 count=4 root@archiso ~ # mkdir /mnt/boot && mkdir /mnt/home </sxh>

Partitioning

The partitioning structure of a disk is every users own choice, that's why we recommend reading the paragraph Partition Scheme in order to get a short introduction about the subject and make up your mind.

Get a pen and a piece of paper and start-off drawing your structure. When done, continue to the next paragraph.

GPT

Basically there exist two "formats" of partitioning a disk: MBR and GPT. As MBR is from the last century and has many disadvantages in comparison with GTP, we are going to use the latter one. For more detailed information about MBR and other possibility, please refer to the Partion table paragraph.

<sxh> # gdisk disk-device </sxh>

You will be thrown in a own commandline of the gdisk program, so proceed as follows:

Step	Command	Explanation
1	o	Create a new GUID partition table.
2	n	Create a new partition. (All partition with GPT are primary)
X	w	Write the partition table to disk.
Y	q	Exit gdisk commandline.

Encryption

<sxh bash; title: with keyfile> ~$ cryptsetup -v –cipher aes-xts-plain64 –key-size 512 –hash sha512 –iter-time 5000 –use-urandom luksFormat <device> keyfile ~$ cryptsetup luksOpen -d keyfile /dev/$DEVICE root ~$ cryptsetup luksOpen -d keyfile /dev/$DEVICE home ~$ cfdisk /dev/$DEVICE ~$ cryptsetup -c aes-xts-plain -y -s 512 luksFormat /dev/$DEVICE # USB storage device ~$ mkfs.vfat -F 32 -I /dev/mapper/bootdevice ~$ cfdisk /dev/$DEVICE1 # make it bootable </sxh>

<sxh bash; title: with password> ~$ cryptsetup -v –cipher aes-xts-plain64 –key-size 512 –hash sha512 –iter-time 5000 –use-urandom –verify-passphrase luksFormat <device> ~$ cryptsetup luksOpen /dev/$DEVICE $CRYPTSETUP_DEVICE_NAME ~$ mkfs.btrfs /dev/mapper/$CRYPTSETUP_DEVICE_NAME ~$ mount /dev/mapper/$CRYPTSETUP_DEVICE_NAME /mnt </sxh>

/boot partition

<sxh bash; title: on disk boot device> ~$ mkfs.ext3 -L boot /dev/$BOOTDEVICE # DOS; primary partition + bootable ~$ cfdisk /dev/$BOOTDEVICE ~$ mount /dev/$BOOTDEVICE /mnt/boot </sxh>

<sxh bash; title: external boot device> ~$ mkfs.ext3 -L boot /dev/$BOOTDEVICE ~$ cfdisk /dev/$BOOTDEVICE # DOS; primary partition + bootable ~$ mkfs.ext3 -L boot /dev/$BOOTDEVICE ~$ mount /dev/$BOOTDEVICE /mnt/boot </sxh>

Mounting

<sxh bash; title: mounting> ~$ mkdir /mnt/boot ~$ mount /dev/mapper/$CRYPTSETUP_DEVICE_NAME /mnt ~$ mount /dev/$BOOTDEVICE /mnt/boot </sxh>

Bootstrapping

<sxh bash;> ~$ pacstrap -i /mnt base base-devel ~$ genfstab -U /mnt > /mnt/etc/fstab ~$ arch-chroot /mnt /bin/bash ~$ nano /etc/locale.gen ~$ locale-gen ~$ echo LANG=en_IE.UTF-8 > /etc/locale.conf ~$ nano /etc/vconsole.conf # KEYMAP=fr ~$ ln -sf /usr/share/zoneinfo/$CONTINENT/$COUNTRY /etc/localtime ~$ hwclock –systohc –utc ~$ echo $HOSTNAME > /etc/hostname ~$ nano /etc/mkinitcpio.conf # add: keymap keyboard encrypt **before** filesystem in the HOOKS variable ~$ mkinitcpio -p linux ~$ passwd root </sxh>

syslinux

<sxh bash;> ~$ pacman -S syslinux gptfdisk ~$ syslinux-install_update -iam ~$ nano /boot/syslinux/syslinux.cfg # APPEND root=/dev/mapper/group-name cryptdevice=/dev/sda2:name rw </sxh>

Unmount & Reboot

Good luck!

<sxh bash;> ~$ exit ~$ umount -R /mnt </sxh>

Troubleshooting

Tiny troubles might pup up. The few below have a tendency to occur due to human interaction failure. ;)

cryptsetup failed

```
Command failed with code 22: Invalid argument
```

Enter the 'yes' in capslock: YES and hit Enter?

syslinux

```
Error: /boot/syslinux is empty!
Is /boot mounted?
```

Reinstall syslinux package.

Write-Protected

If mount /dev/bootdevice /mnt gives you an error about write protection; ask yourself if you have formatted the relative partition in the first place.

Repairing boot partition

<sxh bash;> ~$ cryptsetup luksOpen /dev/$ROOTPARTITION root ~$ mount /dev/mapper/root /mnt ~$ mkfs.ext2 /dev/$BOOTPARTITION ~$ mount /dev/$BOOTPARTITION /mnt/boot ~$ genfstab -U /mnt > /mnt/etc/fstab ~$ arch-chroot /mnt /bin/bash ~$ pacman -R syslinux ~$ pacman -Syy syslinux ~$ syslinux-install_update -iam ~$ nano /boot/syslinux/syslinux.cfg # APPEND root=/dev/mapper/group-name cryptdevice=/dev/sda2:name rw ~$ pacman -S linux ~$ exit ~$ umount -R /mnt </sxh>

References

- https://wiki.archlinux.org/index.php/Dm-crypt/Drive_preparation
- https://wiki.archlinux.org/index.php/Partitioning#Using_GPT_-_modern_method
- http://fedoraproject.org/wiki/Disk_Encryption_User_Guide

From:
https://wiki.c3l.lu/ - **Chaos Computer Club Lëtzebuerg**

Permanent link:
https://wiki.c3l.lu/doku.php?id=projects:howtos:archlinux-full-disk-encryption&rev=1486403265

Last update: **2017/02/06 18:47**