



Harddiskverschlüsselung mit Linux

ACHTUNG: ALLES HIER IST SEHR GEFÄHRLICH, DATEN KÖNNEN VERLOREN GEHEN! DIE ANLEITUNG IST NUR FÜR FORTGESCHRITTENE Linuxanwender geeignet!

Einleitung

Spätestens seit man in SuSE 8.x bei der Festplattenpartitionierung als Option eine Verschlüsselung wählen kann, ist es auch für den Normalnutzer kein Problem, seine Daten sicher und geheim zu halten. Dieser Artikel soll eine Idee aufzeigen, wie der fortgeschrittenen Nutzer diesen Ansatz noch verbessern kann.

Meine Harddisk auf dem Notebook ist unterteilt in drei Partitionen, nebst Swap (benötigt für «Suspend-To-Disk»), eine kleine (11GB) für das System und eine grössere (64GB) für meine Daten. Die grosse Harddisk ist verschlüsselt. Dies verhindert, dass jemand mit meinen Daten etwas anfangen kann, sollte mir mein Notebook abhanden kommen. Beim Systemstart muss ein USB Speichermodul eingesteckt sein, auf dem sich der Schlüssel zur Entschlüsselung der Partition befindet. Dadurch ist keine Passwordeingabe beim Systemstart notwendig. Da sich der Nutzer mit dem Schlüssel identifiziert und die `/home` Daten nur eingebunden sind, wenn der Schlüssel beim starten steckte, erfolgt die Benutzeranmeldung ins X11 (KDE) automatisch und ohne Passwordeingabe (neuerdings verwende ich `ivman` dazu).

Der klassische Ansatz: Mit Passwort

Die Einrichtung für eine Verschlüsselung mit Passwort auf der Kommandozeile ist recht einfach (SuSE 9.1, Kernel 2.6). Angenommen, die verschlüsselte Partition ist `/dev/hda3` und sie soll XFS formatiert und 256 Bit AES verschlüsselt nach `/privat` eingehängt werden (die Wahl von `/dev/loop4` ist willkürlich):

Aufsetzen

Module laden

```
modprobe aes
modprobe cryptoloop
```

Gerät mir Rückschleife aufsetzen (Verschlüsselung)

```
losetup -T -e AES256 /dev/loop4 /dev/hda3
zweimal dasselbe 20 Zeichen Passwort eingeben und ja nicht vergessen
```

Dateisystem aufsetzen

```
mkfs.xfs /dev/loop4
Die Daten auf /dev/hda3 gehen dabei verloren!
```

Rückschleife wieder lösen

```
losetup -d /dev/loop4
```

Automatisieren

in Datei `"/etc/sysconfig/kernel"`

der Liste `MODULES_LOADED_ON_BOOT` die Werte `cryptoloop` und `aes` hinzufügen und `SuSEconfig` aufrufen

in Datei `"/etc/fstab"` folgende Zeile einfügen:

```
/dev/hda3 /privat xfs noauto,user,loop,encryption=AES256
damit jeder Benutzer mit mount /privat einbinden kann, oder so, dass beim Hochlauf automatisch eingebunden wird:
/dev/hda3 /privat xfs loop,encryption=AES256
```

Test

Danach sollte ein `mount /privat` (als Benutzer `root`) das Passwort abfragen und ohne Fehlermeldung die Partition entschlüsselt einbinden. Der Befehl `mount` ohne Parameter sollte dann u.a. dies anzeigen:

```
/dev/hda3 on /privat type xfs
```

Verbesserte Variante

Diese Kapitel setzt die vorherigen voraus, es werden nur die geänderten Befehle beschrieben!

Der ganz grosse Nachteil bei obiger Lösung ist der, dass man sich ein mindestens 20 stelliges Passwort merken muss, welches auch noch fehlerfrei einzutippen ist.

Ein weiterer Nachteil ist, dass man das Passwort nachträglich nicht mehr ändern kann, ohne alles neu anzulegen (und dabei alle Daten zu löschen).

Nun gibt es die folgende Möglichkeit:

1. Man speichert das Passwort in einer Datei, z.B. `/key`
2. Nun wird diese Datei mit GPG symmetrisch verschlüsselt:

- ```
gpg -c /key
```
- Die Datei mit dem Schlüssel im Klartext sollte man an einem sicheren Ort aufbewahren, und auf der Festplatte löschen:  
wipe /key
  - Nun kann man die Partition so aufsetzen, dass der (einfachere) GPG Schlüssel verwendet wird, um die Schlüsseldatei zu entschlüsseln, während die Festplatte mit der entschlüsselten Schlüsseldatei verschlüsselt wird:  
losetup -e AES256 -K /key.gpg /dev/loop4 /dev/hda3
  - Dateisystem formatieren (mkfs.xfs) und Rückschleife lösen (losetup -d) nicht vergessen
  - In der Datei /etc/fstab trägt man nun ein:  
/dev/hda3 /privat xfs loop,encryption=AES256,pgpkey=/key.gpg  
(optional auch mit noauto, user)
  - Jetzt kann man beim Einbinden der verschlüsselten Partition das GPG Passwort angeben

Das GPG Passwort kann man nun ändern, indem man einfach die Schlüsseldatei mit dem alten Passwort entschlüsselt (gpg -d key.gpg<) und dann mit einem neuen Passwort wieder verschlüsselt. Den Schlüssel für die Festplatte kann man aber nach wie vor nicht ändern.

## Entschlüsselung über USB Speicher

Wenn man noch einen Schritt weiter geht, speichert man den GPG Schlüssel auf einem USB Speichermodul und das GPG Passwort für den Schlüssel auf der (unverschlüsselten) Systempartition des Notebooks. So kann man weder aus dem Notebook, noch aus dem Speichermodul allein Informationen erwerben, man benötigt beides. Ohne das Notebook kann man den Harddiskschlüssel nicht entschlüsseln:

### Partitionen

| Partition | Pfad    | Inhalt                                   |
|-----------|---------|------------------------------------------|
| /dev/hda1 | swap    | Auslagerungsspeicher                     |
| /dev/hda2 | /       | System Partition                         |
| /dev/hda3 | /privat | die verschlüsselte Partition             |
| /dev/sda1 | /mnt    | das externe Speichermodul, der Schlüssel |

### Dateien

| Gerät     | Datei         | Inhalt                                                                           |
|-----------|---------------|----------------------------------------------------------------------------------|
| /dev/sda1 | /mnt/.key.gpg | Schlüssel für /dev/hda3 (>20 Zeichen GPG verschlüsselt)                          |
| /dev/hda2 | .key          | Schlüssel für GPG verschlüsselten Schlüssel /mnt/.key.gpg (>20 Zeichen Klartext) |

### Zeile in /etc/fstab

```
/dev/hda3 /privat xfs noauto,acl,user_xattr,loop,encryption=AES256,pgphome=/mnt/.gnupg,pgpkey=/mnt/.key.gpg 0 0
```

### Eintrag in /etc/sysconfig/kernel

```
MODULES_LOADED_ON_BOOT="usbcore uhci_hcd ehci_hcd hci_usb usb_storage scsi_mod cryptoloop aes"
```

Die USB Module müssen geladen werden, um das Speichermodul einbinden zu können.

### Die Datei /etc/init.d/boot.local

Dieses Skript entschlüsselt die Festplatte beim Hochlauf und versucht es so lange immer wieder, bis es geht.

```
#!/bin/sh
echo -n "mounting /privat "
while (mount /dev/sda1 /mnt 2> /dev/null; [$? -ne 0]); do
 echo -n "."
 sleep 1
done
while (mount -p 99 /privat 99< /.key 2> /dev/null; [$? -ne 0]); do
 echo -n "+"
 sleep 1
done
umount /mnt
echo " done."
```

## Einrichtung

### Vorbereitung

```
/etc/fstab, /etc/sysconfig/kernel, /etc/init.d/boot.local wie oben beschrieben anpassen
SuSEconfig
modprobe aes; modprobe cryptoloop
```

### Schlüsselpartition einbinden

```
mount /dev/sda1 /mnt
```

### Datei erstellen

```
mindestens 20 Zeichen speichern auf einer Zeile in /mnt/.key
cd /mnt
gpg -c .key
wipe .key
```

### Passwort speichern

```
soeben benutztes Passwort speichern auf einer Zeile in /.key
```

### GPG Verzeichnis anlegen

```
cp -a /root/.gnupg /mnt/
```

**Festplatte verschlüsseln**

Alle Daten werden gelöscht!

```
dd if=/dev/urandom of=/dev/hda3
losetup -e AES256 -K /mnt/.key.gpg -p 99 /dev/loop4 /dev/hda3 99< /.key
mkfs.xfs /dev/loop4
losetup -d /dev/loop4
```

**Fertig!**

Jetzt kann man die Festplatte einbinden und den Schlüssel freigeben:

```
mount -p 99 /privat 99< /.key
umount /mnt
```

nun kann man die Partition /privat füllen

bei jedem Neustart muss nun der Schlüssel stecken...

Bei mir sieht es so aus:

```
> ls -lA /
insgesamt 208
drwxr-xr-x 2 root root 3048 2004-08-04 23:39 bin
drwxr-xr-x 3 root root 728 2004-08-31 23:38 boot
drwxr-xr-x 34 root root 180456 2004-09-01 17:23 dev
drwxr-xr-x 80 root root 7880 2004-09-01 19:55 etc
lrwxrwxrwx 1 root root 12 2004-07-28 15:28 home -> /privat/home
-rw-r--r-- 1 root root 60 2004-08-03 09:17 .key
drwxr-xr-x 10 root root 3392 2004-08-18 20:15 lib
drwxr-xr-x 5 root root 184 2004-08-27 15:27 media
drwxr-xr-x 2 root root 48 2004-04-06 18:04 mnt
drwxr-xr-x 14 root root 392 2004-08-19 20:05 opt
drwxr-xr-x 6 root root 160 2004-08-06 15:34 privat
dr-xr-xr-x 109 root root 0 2004-09-01 17:22 proc
lrwxrwxrwx 1 root root 10 2004-07-28 17:46 root -> /home/root
drwxr-xr-x 3 root root 8320 2004-08-18 19:51/sbin
drwxr-xr-x 5 root root 120 2004-07-28 01:08 srv
drwxr-xr-x 8 root root 0 2004-09-01 17:22 sys
drwxrwxrwt 25 root root 2088 2004-09-01 23:00 tmp
drwxr-xr-x 15 root root 416 2004-08-31 23:14 usr
drwxr-xr-x 17 root root 432 2004-08-18 20:15 var

> ls -lA /privat
insgesamt 4
drwxr-xr-x 6 root root 168 2004-09-01 17:30 home
drwx----- 4 root root 264 2004-07-28 15:14 root
> mount
/dev/hda2 on / type xfs (rw,acl,user_xattr)
proc on /proc type proc (rw)
tmpfs on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/hdc on /media/cdrecorder type subfs (ro,noexec,nosuid,nodev,fs=cdfss,procluid,icharset=utf8)
/dev/hda3 on /privat type xfs (rw,loop=/dev/loop1,gpgkey=/mnt/.key.gpg,gpghome=/mnt/.gnupg,encryption=AES256,acl,user_xattr)
usbfs on /proc/bus/usb type usbfs (rw)
```

**Mögliche Probleme**

Die («*twofish*») Verschlüsselung hat geändert von Kernel 2.4 zu Kernel 2.6, früher angelegte Partitionen müssen gesichert, gelöscht, neu angelegt und zurückgespielt werden.

Auf einer verschlüsselten Partition muss nur ein einziges Bit kaputt sein, schon sind alle Daten verloren! Regelmässige Sicherheitskopien sind Pflicht! Diese aber nicht ungesichert herumliegen lassen!

**Diskussion**