Luks mit Gnupg-Card unter Debian Stretch

Update für Debian Stretch:

Debian Stretch kommt ausschließlich mit gnupg2 daher.
Daher sind die Skripte ein wenig anzupassen.

Dieser Ansatz beruht im Wesentlichen auf der Anleitung von Peter Lebbing. Allerdings sind für die Unterstützung von Kartenlesern, für die gnupg keine eigene Unterstützung mitbringt, noch kleine Veränderungen nötig. Wir benötigen zusätzlich den pcscd-Dämon innerhalb der Initramdisk:

- Kopieren zusätzlicher Bibliotheken in die Initramdisk
- Starten des pcscd Dämons im decrypt_gnupg_sc Skript

Diese Anleitung setzt voraus, dass diejenigen, die sie anwenden, wissen was sie tun und die zugrundeliegenden Skripte verstehen. Es besteht die Gefahr, dass das System nicht mehr ohne weiteres gebootet werden kann. Anwender_innen sollten wissen, wie sie dann vorgehen müssen.

Vorannahmen

Diese Anleitung geht davon aus, dass bereits ein vollverschlüsseltes System auf Basis von Debian Stretch existiert. Sie beinhaltet nicht die Beschreibung der grundsätzlichen Einrichtung der Festplattenverschlüsselung.

Voraussetzungen

- verschlüsseltes System (root in verschlüsseltem LVM)
- Funktionierendes Setup der Gnupg Card

Ansatz

- Wir erstellen einen zufälligen Schlüssel, den wir als Key dem Luks-Heaer hinzufügen.
- Dieser Key wird mit der GPG-Id einer Gnupg Card verschlüsselt und innerhalb der Initramdisk gespeichert
- Beim Start des Computers wird ein Skript aufgerufen, das nach der PIN-Nummer der Gnupg Card fragt
- Alternativ kann der Key zusätzlich mit einem symmetrischen Schlüssel (Passwort) verschlüsselt werden. Ist die Gnupg Card nicht zur Hand, kann dieser eingegeben werden.

Schlüssel erstellen

Für die nächsten Schritte, die alle als Root ausgeführt werden, benötigen wir die Gnupg User-ID, mit der der Luks-Key verschlüsselt werden soll auch innerhalb des Gnupg-Setups von root. Ggf. muss also der Öffentliche Schlüssel in den Root-Schlüsselbund importiert werden.

Als User:

```
gpg --export --armor USER-ID > /tmp/key.asc
```

als Root:

```
gpg --import /tmp/key.asc
```

Die Skripte gehen davon dass unter:

```
/etc/keys/cryptkey.gpg
```

ein mit der gpg-ID der Gnupg-Card verschlüsselter Key liegt, der die Root-Partition entschlüsseln kann.

Weiterhin müssen unter /etc/keys die Dateien pubring.gpg (mit dem öffentlichen Schlüssel der verwendeten Gnupg ID) und secring.gpg existieren. Keine Sorge: In letzterer existiert kein geheimer Schlüssel sondern nur der Verweis auf die Gnupg Card.

Dass lässt sich z.B. folgendermaßen bewerkstelligen (GNUPG-ID muss durch die richtige ID ersetzt werden):

```
mkdir -m 700 /etc/keys
dd if=/dev/random bs=1 count=256 | gpg -e -o /etc/keys/cryptkey.gpg -r
Gnupg-ID -ec
cd /root
mkfifo -m 700 keyfifo
gpg -d /etc/keys/cryptkey.gpg >keyfifo
```

In einem zweiten Terminal:

(„CRYPTDEVICE“ muss natürlich mit dem richtigen Device ersetzt werden.)

```
cd /root
cryptsetup luksAddKey /dev/CRYPTDEVICE keyfifo
```

Für die folgenden Kommandos muss die Gnupg Card als Root verwendet werden. Evtl. muss, wenn das ganze in einer User-X-Session stattfindet erst der scdaemon des Users beendet werden:

```
killall -9 scdaemon
```

In einem der beiden Terminals (das andere kann geschlossen werden) (**{YOURKEYID}** mit der gewünschten Gnupg-ID ersetzen):

```
rm keyfifo
gpg --export-options export-minimal --export {YOURKEYID} | gpg \
      --no-default-keyring --keyring /etc/keys/pubring.gpg \
      --secret-keyring /etc/keys/secring.gpg --import
gpg --no-default-keyring --keyring /etc/keys/pubring.gpg \
      --secret-keyring /etc/keys/secring.gpg --card-status
```

Anpassung der /etc/crypttab

Vor der Anpassung sollte etwas folgendes in der /etc/crypttab stehen:

```
md1_crypt UUID=46ace282-0706-4fa3-9cb1-bd6f9aaba29d none luks
```

wir ersetzen alles ab `none luks`, so dass es in etwa folgendermaßen aussieht:

```
md1_crypt UUID=46ace282-0706-4fa3-9cb1-bd6f9aaba29d /etc/keys/cryptkey.gpg
luks,keyscript=decrypt_gnupg_sc
```

Anpassung der Initramdisk

Folgende Dateien sind nötig und müssen an die vorgesehenen Stellen kopiert werden:

- /lib/cryptsetup/scripts/decrypt_gnupg_sc
- /etc/initramfs-tools/hooks/cryptgnupg_sc

Für Cardreader ohne Pinpad, bei denen die PIN über die normale Tastatur eingegeben wird, sind weniger Binaries im initramfs erforderlich. Das Skript für Cardreader mit Pinpad funktioniert auch mit Cardreadern ohne Pinpad - allerdings darf der USB-Cardreader erst eingesteckt werden, wenn bereits nach dem Passwort gefragt wird.

: wir bräuchten eine Möglichkeit, Cardreader ohne Pinpad von solchen mit Pinpad zu unterscheiden.

Wird der Schlüssel „cryptkey.gpg“ zusätzlich zu den gpg-Ids auch noch mit einer symmetrischen Passphrase verschlüsselt (siehe oben), dann funktioniert die Freischaltung mit beiden Skripten auch mit der Eingabe der Passphrase über die Tastatur.

Das Skript `decrypt_gnupg_sc` wird später innerhalb der Initramdisk gestartet, startet dort den pcscd und fragt nach der PIN-Nummer der Karte bzw. nach einem Passwort für die zusätzlich vergebene symmetrische Verschlüsselung der gpg-Datei.

Die folgenden Skripte sind für Lesegeräte gedacht, die nicht vom internen ccid-Treiber des gnupg-Binaries unterstützt werden sondern einen laufenden pcsc Daemon benötigen. Falls der Kartenleser vom internen ccid unterstützt wird, können und müssen die entsprechenden Zeilen ausgelassen werden. Ob mit laufendem pcscd auch

Kartenleser unterstützt werden, die diesen normalerweise nicht benötigen, habe ich noch nicht getestet.

Cardreader ohne Pinpad

Seit Stretch funktioniert der Hack nach meiner Erfahrung so (s.u. #Cardreader mit Pinpad) nicht mehr für Cardreader ohne Pinpad. Genau habe ich das noch nicht analysiert - aber es hat wohl was mit der Art, wie die gnupg2 Version die pinentry Funktion aufruft, zu tun.

Hier gibt es eine funktionierende Anleitung:

- https://wiki.majic.rs/Openpgp/protecting_luks_decryption_key_in_debian_jessie_us/

Auf unser Szenario bezogen benötigen wir folgende Dateien (beide müssen ausführbar sein):

/etc/initramfs-tools/hooks/cryptgnupg_sc

```
#!/bin/sh

set -e

PREREQ="cryptroot"

prereqs()
{
        echo "$PREREQ"
}

case $1 in
prereqs)
        prereqs
        exit 0
        ;;
esac

. /usr/share/initramfs-tools/hook-functions

if [ ! -x ${DESTDIR}/lib/cryptsetup/scripts/decrypt_gnupg_sc ] ; then
    exit 0
fi


# Deploy the keyring.
cp -a /etc/keys/ "${DESTDIR}/etc/"

# Deploy terminfo (required for pinentry-curses).
mkdir -p "${DESTDIR}/etc/terminfo/l/"
cp -a /lib/terminfo/l/linux "${DESTDIR}/etc/terminfo/l/linux"
```

```
# Deploy GnuPG binaries and pinentry-curses.
copy_exec /usr/bin/gpg
copy_exec /usr/bin/gpg2
copy_exec /usr/bin/gpg-agent
copy_exec /usr/bin/pinentry-curses

# some more libs for pcscd
copy_exec /usr/sbin/pcscd
copy_exec /usr/lib/pcsc/drivers/serial/libccidtwin.so
copy_exec /usr/lib/pcsc/drivers/ifd-
ccid.bundle/Contents/Linux/libccid.so
copy_exec /usr/lib/pcsc/drivers/ifd-ccid.bundle/Contents/Info.plist
copy_exec /etc/libccid_Info.plist
if uname -r | grep -q amd64; then
    copy_exec /usr/lib/x86_64-linux-gnu/libpcsclite.so.1.0.0
    copy_exec /lib/x86_64-linux-gnu/libgcc_s.so.1
else
    copy_exec /usr/lib/i386-linux-gnu/libpcsclite.so.1.0.0
    copy_exec /lib/i386-linux-gnu/libgcc_s.so.1
fi

# we need some more stuff from gnupg2
copy_exec /usr/lib/gnupg/scdaemon
copy_exec /usr/bin/pinentry


echo "initramfs for luks decryption with gnupg-card done"

exit 0
```

```
chmod 755 /etc/initramfs-tools/hooks/cryptgnupg_sc
```

/lib/cryptsetup/scripts/decrypt_gnupg_sc

```
#!/bin/sh

# This is the safest way to ensure the GnuPG home directory is
correctly set.
export GNUPGHOME=/etc/keys/

# quick hack for starting pcscd
mkdir -p /var/run
pcscd &

sleep 3
```

```
gpg2 --no-tty --decrypt /etc/keys/cryptkey.gpg
```

```
chmod 750 /lib/cryptsetup/scripts/decrypt_gnupg_sc
```

Cardreader mit Pinpad

Die Anleitung bezieht sich auf Cardreader mit oder ohne Pinpads. Falls es Bei Cardreadern ohne Pinpad zu problemen kommt, hefen vielleicht die Hinweise aus den älteren Fassungen: Luks mit Gnupg-Card unter Wheezy und Jessie

Am Anfang des Skripts wird eine kleine Abfrage gemacht, ob ein Kartenleser existiert. Wenn nicht, fragt das Skript nach der Passphrase. Diese Unterscheidung ist nötig, weil „askpass“ immer auf eine Eingabebestätigung wartet.

~~Wir können diesen Umstand ausnutzen, um auch Kartenleser ohne Pinpad zu nutzen. Wenn der Kartenleser erst eingesteckt wird, wenn das Skript sich schon in der Schleife mit „askpass“ befindet, sollte es auch mit der Karte und der Eingabe der PIN über die Tastatur funktionieren.~~ Das funktioniert bei mir seit Debian Stretch nicht mehr (s.o.)

/lib/cryptsetup/scripts/decrypt_gnupg_sc

```
#!/bin/sh

# quick hack for starting pcscd
mkdir -p /var/run
pcscd &

sleep 3

decrypt_gpg () {

        # we check for attachted cardreader
        # cardreaders with pinpad: they have to be attached before
booting
        # for cardreaders without pinpad: just attach them later, if
the script already
        # asks for Passphrase or PIN
        if gpg -q --no-tty --homedir "$(dirname $1)" --card-status >
/dev/null 2>&1 && sleep 2; then
                echo "Please use the pinpad of your cardreader for PIN
entry." >&2
                if ! /usr/bin/gpg --homedir "$(dirname $1)" \
                        --trustdb-name /dev/null --decrypt $1; then
                        return 1
                fi
```

```
                return 0
        else
                echo "Performing GPG key decryption ..." >&2
                if ! /lib/cryptsetup/askpass \
                        "Enter passphrase for key $1, or PIN for your
cardreader: " | \
                        /usr/bin/gpg -q --batch --homedir "$(dirname
$1)" \
                        --trustdb-name /dev/null --passphrase-fd 0 --
decrypt $1; then
                        return 1
                fi
                return 0
        fi
}

if [ ! -x /usr/bin/gpg ]; then
        echo "$0: /usr/bin/gpg is not available" >&2
        exit 1
fi

if [ -z "$1" ]; then
        echo "$0: missing key as argument" >&2
        exit 1
fi

decrypt_gpg "$1"
exit $?
```

Das Skript muss ausführbar sein:

```
chmod 755 /lib/cryptsetup/scripts/decrypt_gnupg_sc
```

Das Hook-Skript erledigt die Anpassungen in der Initramdisk und kopiert die nötigen binaries und Bibliotheken in die Initramdisk (u.a. gpg, pcscd):

/etc/initramfs-tools/hooks/cryptgnupg_sc

```
<file bash /etc/initramfs-tools/hooks/cryptgnupg_sc>
#!/bin/sh

set -e

PREREQ="cryptroot"

prereqs()
{
        echo "$PREREQ"
}
```

```
case $1 in
prereqs)
        prereqs
        exit 0
        ;;
esac

. /usr/share/initramfs-tools/hook-functions

# Hooks for loading Gnupg software and key into the initramfs

# Check whether cryptroot hook has installed decrypt_gnupg_sc script
if [ ! -x ${DESTDIR}/lib/cryptsetup/scripts/decrypt_gnupg_sc ] ; then
    exit 0
fi

# Install cryptroot key files into initramfs
keys=$(sed 's/^\(.*,\|\)key=//; s/,.*//'
${DESTDIR}/conf/conf.d/cryptroot)

if [ "${keys}" != "none" ]; then
    if [ -z "${keys}" ]; then
        echo "$0: Missing key files in
${DESTDIR}/conf/conf.d/cryptroot" >&2
        cat ${DESTDIR}/conf/conf.d/cryptroot >&2
        exit 1
    fi
    for key in ${keys} ; do
        keydir=$(dirname ${key})
        echo "WARNING: GnuPG key $key is copied to initramfs" >&2
        echo "WARNING: GnuPG secret keyring ${keydir}/secring.gpg is
copied" \
             "to initramfs" >&2
        if [ ! -d ${DESTDIR}/${keydir} ] ; then
            mkdir -p ${DESTDIR}/${keydir}
        fi
        chmod 700 ${DESTDIR}/${keydir}
        cp -p ${key} ${DESTDIR}/${key}
        cp -p ${keydir}/pubring.gpg ${DESTDIR}/${keydir}
        cp -p ${keydir}/secring.gpg ${DESTDIR}/${keydir}
        if [ -e ${keydir}/gpg.conf ] ; then
            cp -p ${keydir}/gpg.conf ${DESTDIR}/${keydir}
        fi
    done
fi

# Install gnupg software
copy_exec /usr/bin/gpg
copy_exec /usr/bin/gpg
copy_exec /usr/bin/gpg-agent
```

```
# some more libs for pcscd
copy_exec /usr/sbin/pcscd
copy_exec /usr/lib/pcsc/drivers/serial/libccidtwin.so
copy_exec /usr/lib/pcsc/drivers/ifd-
ccid.bundle/Contents/Linux/libccid.so
copy_exec /usr/lib/pcsc/drivers/ifd-ccid.bundle/Contents/Info.plist
copy_exec /etc/libccid_Info.plist
if uname -r | grep -q amd64; then
        copy_exec /usr/lib/x86_64-linux-gnu/libpcsclite.so.1.0.0
        copy_exec /lib/x86_64-linux-gnu/libgcc_s.so.1
else
        copy_exec /usr/lib/i386-linux-gnu/libpcsclite.so.1.0.0
        copy_exec /lib/i386-linux-gnu/libgcc_s.so.1
fi

# we need some more stuff from gnupg2
copy_exec /usr/lib/gnupg/scdaemon
copy_exec /usr/bin/pinentry

exit 0
```

Achtung: das Skript funktioniert für i386 und amd64 Architekturen. Für andere Architekturen muss es entsprechend angepasst werden.

Die copy_exec Funktion aus /usr/share/initramfs-tools/hook-functions ist eigentlich für Binaries gedacht, um auch abhängige Bibliotheken zu kopieren. Das funktioniert so auch mit anderen Dateien, ist aber vermutlich nicht dafür gedacht.

Die nötigen Bibliotheken stecken in den Debian-Packeten libccid, libpcsclite1 und libgcc1.

Auch dieses Skript muss ausführbar sein:

```
chmod 755 /etc/initramfs-tools/hooks/cryptgnupg_sc
```

Update der Initramdisk

Jetzt muss noch die Initramdisk neu erstellt werden. Vorher ist dringend geraten, eine Kopie der Initramdisk zu erstellen, mit der gebootet werden kann, falls das über die Gnupg Card aus irgendeinem Grund nicht funktioniert.

Z.B.

```
cp initrd.img-3.2.0-4-amd64 initrd.img-3.2.0-4-amd64-orig
```

Dann kann im Fall des Falles der Grub-Eintrag editiert werden. (Funktioniert einfach, indem beim Booten „e“ auf dem Grub-Eintrag gedrückt wird. Jetzt muss nur die Zeile mit der initrd gesucht werden und ein „-orig“ angehängt werden.) Dann bootet das System wie gewohnt. Das erspart einen den umständlicheren Einsatz eines Live-Rettungs-Systems.

Wir können jetzt aber unsere Initramdisk mit

```
update-initramfs -u
```

neu erstellen.

Der Output sollte in etwas so aussehen:

```
update-initramfs: Generating /boot/initrd.img-3.2.0-4-686-pae
WARNING: GnuPG key /etc/keys/cryptkey.gpg is copied to initramfs
WARNING: GnuPG secret keyring /etc/keys/secring.gpg is copied to initramfs
WARNING: GnuPG key /etc/keys/cryptkey.gpg is copied to initramfs
WARNING: GnuPG secret keyring /etc/keys/secring.gpg is copied to initramfs
```

Wenn Fehlermeldungen auftauchen, ist etwas schief gelaufen. Dann sollte vor einem Reboot unbedingt der Fehler gefunden werden, bzw. die Änderungen rückgängig gemacht werden. (Hook-Skript löschen und Initramdisk neu erstellen).

Ein Reboot des Systems zeigt, ob es geklappt hat.

Debugging

Debugging von Skripten im initramfs ist mühsam, weil der Computer immer wieder neu gebootet werden muss - und die Debugging-Ausgaben und Logs begrenzt sind. Hilfreich ist es oft, die Skripte unter den Live-Bedingungen des initramfs in einer Shell zu testen. Nach einem (recht langen) Timeout booten Debian-Systeme in eine Rescue-Shell des initrafs. Dort lassen sich dann die Skripte direkt ansprechen - (mit vi auch bearbeiten - allerdings natürlich nicht persistent).

Um nicht jedesmal lange zu warten, bis das System in die initramfs-Shell bootet, lässt sich das auch mit einem Kernel-Boot-Parameter provozieren:

```
break=premount
```

im Grub Bootloader zur Kernel-Zeile hinzugefügt (lässt sich vor dem Starten des Grub Menüeintrags mit STRG + E editieren) bootet vor dem Mounten der Dateisysteme (also auch vor dem Luks-Unlocking) in die initramdisk.

Dort lässt sich dann z.B. das Skript direkt aufrufen mit /lib/cryptsetup/scripts/decrypt_gnupg_sc /etc/keys/cryptkey.pgp

Ein Unterschied zum Bootvorgang: Während des Boot-Vorgangs steht kein tty zur Verfügung.

Links

Weitere Ansätze zur Nutzung der Gnupg Karte mit Luks finden sich auf folgenden Seiten:

- http://lists.gnupg.org/pipermail/gnupg-users/2009-November/037599.html → hier wird gezeigt, wie mensch die vier private „DO“ data auf der Karte verwenden kann
- Hier eine komplette Anleitung für Ubuntu Luks mit 2-Faktor Authentifizierung: https://blog.kumina.nl/2010/07/two-factor-luks-using-ubuntu/