Encrypted Root Filesystem HOWTO

Christophe Devine

Revision History

Revision	Date	Author
Revision v1.3	2005-03-13	cd
Updated the packages version.		
Revision v1.2	2004-10-20	cd
Updated the packages version.		
Revision v1.1	2003-12-01	cd
Added support for GRUB.		
Revision v1.0	2003-09-24	cd
Initial release, reviewed by LDP.		
Revision v0.9	2003-09-11	cd
Updated and converted to DocBook XML.		

Abstract

This document explains how to make your personal data secure by encrypting your Linux root filesystem using strong cryptography.

Table of Contents

Preparing the system

Setting up the partition layout

Your hard disk (hda) should contain at least three partitions:

- hda1: this small unencrypted partition will ask for a password in order to mount the encrypted root filesystem.
- hda2: this partition will contain your encrypted root filesystem; make sure it is large enough.
- hda3: this partition holds the current GNU/Linux system.

At this point, both hda1 and hda2 are unused. hda3 is where your Linux distribution is currently installed; /usr and /boot must *not* be separated from this partition.

Here's an example of what your partition layout might look like:

```
# fdisk -l /dev/hda

Disk /dev/hda: 255 heads, 63 sectors, 2432 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1         1      8001   83  Linux
/dev/hda2             2       263   2104515   83  Linux
/dev/hda3           264       525   2104515   83  Linux
/dev/hda4           526      2047  12225465   83  Linux
```

Required packages

If you use Debian, the following packages are mandatory:

```
apt-get install gcc make libncurses5-dev patch bzip2 wget
```

To make copy & paste easier, you should also install:

```
apt-get install lynx gpm
```

Installing Linux-2.4.29

There are two main projects which add loopback encryption support in the kernel: cryptoloop and loop-AES. This howto is based on loop-AES, since it features an extremely fast and highly optimized implementation of Rijndael in assembly language, and therefore provides maximum performance if you have an IA-32 (x86) CPU. Besides, there are some security concerns [http://groups.google.com/groups?selm=1emrG-1Ck-25%40gated-at.bofh.it] about cryptoloop.

First of all, download and unpack the loop-AES package:

```
cd /usr/src
wget http://loop-aes.sourceforge.net/loop-AES/loop-AES-v3.0b.tar.bz2
tar -xvjf loop-AES-v3.0b.tar.bz2
```

Then you must download and patch the kernel source:

```
wget http://ftp.kernel.org/pub/linux/kernel/v2.4/linux-2.4.29.tar.bz2
tar -xvjf linux-2.4.29.tar.bz2
cd linux-2.4.29
rm include/linux/loop.h drivers/block/loop.c
patch -Np1 -i ../loop-AES-v3.0b/kernel-2.4.28.diff
```

Setup the keyboard map:

```
dumpkeys | loadkeys -m - > drivers/char/defkeymap.c
```

Next, configure your kernel; make sure the following options are set:

```
make menuconfig
```

```
    Block devices  --->

        <*> Loopback device support
        [*]   AES encrypted loop device support (NEW)

        <*> RAM disk support
        (4096)   Default RAM disk size (NEW)
        [*]   Initial RAM disk (initrd) support

    File systems  --->

        <*> Ext3 journalling file system support
        <*> Second extended fs support

(important note: do not enable /dev file system support)
```

Compile the kernel and install it:

```
make dep bzImage
make modules modules_install
cp arch/i386/boot/bzImage /boot/vmlinuz
```

If grub is your bootloader, update /boot/grub/menu.lst or /boot/grub/grub.conf:

```
cat > /boot/grub/menu.lst << EOF
default 0
timeout 10
color green/black light-green/black
title Linux
    root (hd0,2)
    kernel /boot/vmlinuz ro root=/dev/hda3
EOF
```

Otherwise, update /etc/lilo.conf and run lilo:

```
cat > /etc/lilo.conf << EOF
lba32
boot=/dev/hda
prompt
timeout=60
image=/boot/vmlinuz
    label=Linux
    read-only
    root=/dev/hda3
EOF
lilo
```

You may now restart the system.

Installing Linux-2.6.10

Proceed as described in the previous section, using loop-aes' *kernel-2.6.10.diff* patch instead, and make sure cryptoloop support is *not* activated. Note that modules support require that you have the module-init-tools package installed.

Installing util-linux-2.12p

The losetup program, which is part of the util-linux package, must be patched and recompiled in order to add strong cryptography support. Download, unpack and patch util-linux:

```
cd /usr/src
wget http://ftp.kernel.org/pub/linux/utils/util-linux/util-linux-2.12p.tar.bz2
tar -xvjf util-linux-2.12p.tar.bz2
cd util-linux-2.12p
patch -Np1 -i ../loop-AES-v3.0b/util-linux-2.12p.diff
```

To use passwords that are less than 20 characters, enter:

```
CFLAGS="-O2 -DLOOP_PASSWORD_MIN_LENGTH=8"; export CFLAGS
```

Security is certainly your major concern. For this reason, please do not enable passwords shorter than 20 characters. Data privacy is not free, one has to 'pay' in form of long passwords.

Compile losetup and install it as root:

```
./configure && make lib mount
mv -f /sbin/losetup /sbin/losetup~
rm -f /usr/share/man/man8/losetup.8*
cd mount
gzip losetup.8
cp losetup /sbin
cp losetup.8.gz /usr/share/man/man8/
chattr +i /sbin/losetup
```

Creating the encrypted root filesystem

Fill the target partition with random data:

```
shred -n 1 -v /dev/hda2
```

Setup the encrypted loopback device:

```
losetup -e aes256 -S xxxxxx /dev/loop0 /dev/hda2
```

To prevent optimized dictionary attacks, it is recommended to add the -S xxxxxx option, where "xxxxxx" is your randomly chosen seed (for example, you might choose "gPk4lA"). Write down your seed on a piece of paper so that you don't loose it afterwards. Also, in order to avoid boot-time problems with the keyboard map, do not use non-ASCII characters (accents, etc.) in your password. The Diceware [http://www.diceware.com/] site offers a simple way to create strong, yet easy to remember, passphrases.

Now create the ext3 filesystem:

```
mke2fs -j /dev/loop0
```

Check that the password you entered is correct:

```
losetup -d /dev/loop0
losetup -e aes256 -S xxxxxx /dev/loop0 /dev/hda2
```

```
mkdir /mnt/efs
mount /dev/loop0 /mnt/efs
```

You can compare the encrypted and unencrypted data:

```
xxd /dev/hda2  | less
xxd /dev/loop0 | less
```

It's time to install your encrypted Linux system. If you use a GNU/Linux distribution (such as Debian, Slackware, Gentoo, Mandrake, RedHat/Fedora, SuSE, etc.), run the following command:

```
cp -avx / /mnt/efs
```

If you use the Linux From Scratch book, proceed as described in the manual, with the modifications below:

- Chapter 6 - Installing util-linux:

  Apply the loop-AES patch after unpacking the sources.

- Chapter 8 - Making the LFS system bootable:

  Refer to the next section (Setting up the boot device).

Setting up the boot device

Creating the ramdisk

To begin with, chroot inside the encrypted partition and create the boot device mount point:

```
chroot /mnt/efs
mkdir /loader
```

Then, create the initial ramdisk (initrd), which will be needed afterwards:

```
cd
dd if=/dev/zero of=initrd bs=1k count=4096
mke2fs -F initrd
mkdir ramdisk
mount -o loop initrd ramdisk
```

If you're using grsecurity, you may get a "Permission denied" error message; in this case you'll have to run the mount command outside chroot.

Create the filesystem hierarchy and copy the required files in it:

```
mkdir ramdisk/{bin,dev,lib,mnt,sbin}
cp /bin/{bash,mount} ramdisk/bin/
ln -s bash ramdisk/bin/sh
mknod -m 600 ramdisk/dev/console c 5 1
mknod -m 600 ramdisk/dev/hda2    b 3 2
mknod -m 600 ramdisk/dev/loop0   b 7 0
cp /lib/{ld-linux.so.2,libc.so.6,libdl.so.2} ramdisk/lib/
cp /lib/{libncurses.so.5,libtermcap.so.2}    ramdisk/lib/
cp /sbin/{losetup,pivot_root} ramdisk/sbin/
```

It's ok if you see a message like "/lib/libncurses.so.5: No such file or directory", or "/lib/libtermcap.so.2: No such file or directory"; bash only requires one of these two libraries. You can check which one is actually required with:

```
ldd /bin/bash
```

Compile the sleep program, which will prevent the password prompt being flooded by kernel messages (such as usb devices being registered).

```
cat > sleep.c << "EOF"
#include <unistd.h>
#include <stdlib.h>

int main( int argc, char *argv[] )
{
    if( argc == 2 )
        sleep( atoi( argv[1] ) );

    return( 0 );
}
EOF

gcc -s sleep.c -o ramdisk/bin/sleep
rm sleep.c
```

Create the init script:

```
cat > ramdisk/sbin/init << "EOF"
#!/bin/sh

/bin/sleep 3

echo -n "Enter seed value: "
read SEED

/sbin/losetup -e aes256 -S $SEED /dev/loop0 /dev/hda2
/bin/mount -r -n -t ext3 /dev/loop0 /mnt

while [ $? -ne 0 ]
do
    /sbin/losetup -d /dev/loop0
    /sbin/losetup -e aes256 -S $SEED /dev/loop0 /dev/hda2
    /bin/mount -r -n -t ext3 /dev/loop0 /mnt
done

cd /mnt
/sbin/pivot_root . loader
exec /usr/sbin/chroot . /sbin/init
EOF

chmod 755 ramdisk/sbin/init
```

Umount the loopback device and compress the initrd:

```
umount -d ramdisk
```

```
rmdir ramdisk
gzip initrd
mv initrd.gz /boot/
```

Booting from a CD-ROM

I strongly advise you to start your system with a read-only media, such as a bootable CD-ROM.

Download and unpack syslinux:

```
wget http://ftp.kernel.org/pub/linux/utils/boot/syslinux/syslinux-3.07.tar.bz2
tar -xvjf syslinux-3.07.tar.bz2
```

Configure isolinux:

```
mkdir bootcd
cp /boot/{vmlinuz,initrd.gz} syslinux-3.07/isolinux.bin bootcd
echo "DEFAULT /vmlinuz initrd=initrd.gz ro root=/dev/ram0" \
    > bootcd/isolinux.cfg
```

Create and burn the bootable cd-rom iso image:

```
mkisofs -o bootcd.iso -b isolinux.bin -c boot.cat \
        -no-emul-boot -boot-load-size 4 -boot-info-table \
        -J -hide-rr-moved -R bootcd/

cdrecord -dev 0,0,0 -speed 4 -v bootcd.iso

rm -rf bootcd{,.iso}
```

Booting from a HD partition

The boot partition can come in handy if you happen to lose your bootable CD. *Remember that hda1 is a writable media and is thus insecure; use it only in case of emergency!*

Create and mount the ext2 filesystem:

```
dd if=/dev/zero of=/dev/hda1 bs=8192
mke2fs /dev/hda1
mount /dev/hda1 /loader
```

Copy the kernel and the initial ramdisk:

```
cp /boot/{vmlinuz,initrd.gz} /loader
```

If you use grub:

```
mkdir /loader/boot
cp -av /boot/grub /loader/boot/
cat > /loader/boot/grub/menu.lst << EOF
default 0
timeout 10
color green/black light-green/black
title Linux
    root (hd0,0)
    kernel /vmlinuz ro root=/dev/ram0
    initrd /initrd.gz
EOF
grub-install --root-directory=/loader /dev/hda
umount /loader
```

If you use lilo:

```
mkdir /loader/{boot,dev,etc}
cp /boot/boot.b /loader/boot/
mknod -m 600 /loader/dev/hda  b 3 0
mknod -m 600 /loader/dev/hda1 b 3 1
mknod -m 600 /loader/dev/hda2 b 3 2
mknod -m 600 /loader/dev/hda3 b 3 3
mknod -m 600 /loader/dev/hda4 b 3 4
mknod -m 600 /loader/dev/ram0 b 1 0
cat > /loader/etc/lilo.conf << EOF
lba32
boot=/dev/hda
prompt
timeout=60
image=/vmlinuz
    label=Linux
    initrd=/initrd.gz
    read-only
    root=/dev/ram0
EOF
lilo -r /loader
umount /loader
```

Final steps

Still inside chroot, modify /etc/fstab so that it contains:

```
/dev/loop0      /               ext3    defaults       0       1
```

Delete /etc/mtab and exit from chroot. Finally, run "umount -d /mnt/efs" and reboot. If something goes wrong, you can still boot your unencrypted partition by entering "Linux root=/dev/hda3" at the LILO: prompt.

If everything went well, you can now re-partition your disk and encrypt hda3 as well as hda4. In the following scripts, we assume that hda3 will hold the swap device and hda4 will contain /home; you should initialize both partitions first:

```
shred -n 1 -v /dev/hda3
shred -n 1 -v /dev/hda4
losetup -e aes256 -S xxxxxx /dev/loop1 /dev/hda3
losetup -e aes256 -S xxxxxx /dev/loop2 /dev/hda4
mkswap /dev/loop1
mke2fs -j /dev/loop2
```

Then create a script in the system startup directory and update fstab:

```
cat > /etc/init.d/loop << "EOF"
#!/bin/sh
```

```
if [ "`/usr/bin/md5sum /dev/hda1`" != \
     "5671cebdb3bed87c3b3c345f0101d016  /dev/hda1" ]
then
    echo -n "WARNING! hda1 integrity verification FAILED - press enter."
    read
fi

echo "1st password chosen above" | \
    /sbin/losetup -p 0 -e aes256 -S xxxxxx /dev/loop1 /dev/hda3

echo "2nd password chosen above" | \
    /sbin/losetup -p 0 -e aes256 -S xxxxxx /dev/loop2 /dev/hda4

/sbin/swapon /dev/loop1

for i in `seq 0 63`
do
    echo -n -e "\33[10;10]\33[11;10]" > /dev/tty$i
done

EOF

chmod 700 /etc/init.d/loop
ln -s ../init.d/loop /etc/rcS.d/S00loop
vi /etc/fstab
...
/dev/loop2      /home           ext3    defaults       0       2
```

About this HOWTO

The Encrypted Root Filesystem HOWTO was first written in november 2002 for the Linux From Scratch [http://www.linuxfromscratch.org/lfs/news.html] project. I'd like to thank the many people who have since contributed to this document (in reverse chronological order): Micha Borrmann, Dennis Lemckert, Oleg Vyushin, Ellen Bokhorst, Daczi László, Gaetano Zappulla, Guillaume Lehmann, Claude Thomassin, Jean-Philippe Guérard, Luc Vo Van, Jacobus Brink, Ernesto Pérez Estévez, Matthew Ploessel, Mike Lorek, Lars Bungum, Michael Shields, Julien Perrot, Grant Stephenson, Cary W. Gilmer, James Howells, Pedro Baez, Josh Purinton, Jari Ruusu and Zibeli Aton.

This HOWTO has been translated in various languages:

- French [http://www.traduc.org/docs/HOWTO/lecture/Encrypted-Root-Filesystem-HOWTO.html]
- Italian [http://www.linux.it/~gaetano/erfs/]
- Hungarian [http://tldp.fsf.hu/HOWTO/Encrypted-Root-Filesystem-HOWTO-hu/]
- Dutch [http://doc.nl.linux.org/HOWTO/Encrypted-Root-Filesystem-HOWTO-NL/article.html]

Please send any comment to Christophe Devine [http://www.cr0.net:8040/about/].