Loopback Encrypted Filesystem HOWTO

Table of Contents

Loopback Encrypted Filesystem HOWTO

Copyright by Ryan T. Rhea, `rhear@cs.winthrop.edu`

v1.1, 29 November 1999

This document explains how to setup and then use a filesystem that, when mounted by a user, dynamically and transparently encrypts its contents. The filesystem is stored in a regular file, which can be hidden or named non−conspicuously such that it would most likely be overlooked. This allows for a high level of secure storage of data.

1.Before you begin

This process requires the kernel source code, knowledge of compiling this code, and a lot of patience. I highly recommend having a boot disk ready. Also, be sure to have a backup before you permanently store your important data on the encrypted filesystem – it can be corrupted like any other filesystem.

As a minimum, you will have to patch to at least version 2.2.9 of the linux kernel before continuing. There are further instructions on applying patches in the Details section later in this document.

Kernel source can be found at:

```
ftp://ftp.kerneli.org/
```

There is a HOWTO on the process of recompiling kernels at:

```
http://metalab.unc.edu/LDP/HOWTO/
```


2.Introduction

The process uses the device '/dev/loop*' (where * can be 0−7 on most installations) to mount a loopback filesystem. The same process can be used without encryption to store a linux filesystem on a non−linux partition. There is a HOWTO on this at the LDP site mentioned previously.

Different types of encryption can be used, including XOR, DES, twofish, blowfish, cast128, serpent, MARS, RC6, DFC, and IDEA. The program 'losetup' (loopback setup) is what associates your encrypted file with a filesystem and it's cipher type. According to Alexander Kjeldaas, who maintains kerneli.org and the international crypto patches, DES and losetup are currently incompatible. This is due to differences in the way the two handle parity bits. There are no plans to support DES as it is much more insecure than the other ciphers.

Twofish, blowfish, cast128, and serpent are all licensed free for any use. The others may or may not have licensing restrictions. Several of them are candidates for the AES standard. The finalists will provide royalty free use of their ciphers worldwide.

This document uses the serpent algorithm because it is strong yet remarkably fast, and it's freely distributable under the GPL. According to it's documentation, serpent uses a 128−bit block cipher designed by Ross Anderson, Eli Biham and Lars Knudsen. It provides users with the highest practical level of assurance that no shortcut attacks will be found. The documentation on serpent as well as the source code can be found at:

```
http://www.cl.cam.ac.uk/~rja14/serpent.html
```

Also, this document assumes that the ciphers are compiled directly into the kernel. You may install them as modules, but the technique is not discussed in this document. You will have to edit the file '/etc/conf.module'; the process is discussed in detail in the kernel compilation HOWTO referenced previously.

3.Summary

There are many steps involved in the process. I will provide Details for these steps in the next section. I thought it would be nice to provide a summary first to provide reference (if you are experienced with unix/linux you probably don't need the details anyway). Here they are summarized as follows:

1. Download the newest international crypto patch (I used 'patch−int−2.2.10.4' at the time this document was written) from:

   ```
   http://ftp.kerneli.org/pub/kerneli/
   ```

2. Patch the kernel
3. Run 'config' (or 'menuconfig' or 'xconfig') to configure your 'MakeFile' for the new kernel. The options to enable encryption are scattered. First of all, before you will see any other options you must enable 'Prompt for development and/or incomplete code/drivers' under 'Code Maturity level options'. Under 'Crypto options' enable 'crypto ciphers' and 'serpent'. Once again, this document assumes you are using serpent, but try whatever you want. Remember that DES is known to be incompatible as of 2.2.10.4 − it may never be supported at all. There are several important options to select under 'Block Devices'. These include 'Loopback device support', 'Use relative block numbers as basis for transfer functions (RECOMMENDED)', and 'General encryption support'. DO NOT select 'cast 128' or 'twofish' encryption here. Also note that you don't need any of the crypto options under the various network categories. I will not go any further into configuration of the kernel, it is out of the scope of this document and can be found at the LDP site.
4. Compile the new kernel.
5. Edit '/etc/lilo.conf' to add the new kernel image. Run 'lilo −v' to add the kernel to the boot loader.
6. Download the source for the newest 'util−linux' (I used 'util−linux−2.9v') package from:

   ```
   ftp://ftp.kernel.org/pub/linux/utils/util-linux/
   ```

7. Extract the 'util−linux' source.
8. Apply the corresponding patch found in your '/usr/src/linux/Documentation/crypto/' directory.

9. CAREFULLY read the 'INSTALL' file! This package contains the sources for many system dependent files (important tools such as 'login', 'passwd', and 'init'). If you don't carefully edit the MCONFIG file before compiling these sources have a boot disk and/or shotgun ready because your system will be quite confused. Basically you want to set almost all of the 'HAVE_*' fields equal to yes so that the important authentication tools are not compiled and written over. The tools you do want rebuilt are 'mount' and 'losetup' to accommodate the new encryption schemes. I suggest that you refer to the Details section below for this step.
10. Compile and install the 'util−linux' source
11. Reboot the machine with the new kernel.
12. Edit '/etc/fstab', adding an entry for your mount point as follows:

    ```
    /dev/loop0  /mnt/crypt  ext2  user,noauto,rw,loop 0 0
    ```
13. Create the directory that will hold your filesystem, as in '/mnt/crypt' above.
14. As the user, create your encrypted file as follows:

    ```
    dd if=/dev/urandom of=/etc/cryptfile bs=1M count=10
    ```
15. Run losetup as follows:

    ```
    losetup −e serpent /dev/loop0 /etc/cryptfile
    ```

    You only have one chance to enter the password, be careful. If you want to double−check your password, you can use the command:

    ```
    losetup −d /dev/loop0
    ```

    This will deactivate your loop device. Next you will run losetup again to test your password, as follows:

```
losetup -e serpent /dev/loop0 /etc/cryptfile
```

16. Make your ext2 filesystem as follows:

```
mkfs -t ext2 /dev/loop0
```

17. Now you can mount the encrypted filesystem with:

```
mount -t ext2 /dev/loop0 /mnt/crypt
```

18. When your done, you want to unmount and protect your filesystem as follows:

```
umount /dev/loop0
losetup -d /dev/loop0
```

4.Details

Kernel Patches:

You can upgrade from '2.2.x' releases by patching. Each patch that is released for '2.2.x' contains bugfixes. New features will be added to the Linux '2.3.x' development kernel. To install by patching, get all the newer patch files and do the following:

```
cd /usr/src
gzip -cd patchXX.gz | patch -p0
```

Repeat xx for all versions bigger than the version of your current source tree, IN ORDER.

The default directory for the kernel source is '/usr/src/linux'. If your source is installed somewhere else, I would suggest using a symbolic link from '/usr/src/linux'.

Editing 'MCONFIG' for the 'util−linux' package compilation:

The following are excerpts from the 'MCONFIG' file I used to compile the 'util−linux' package. Note that this is fairly specific for my setup, which is loosely based on RedHat 5.2. The point is to make sure you don't overwrite any important system tools such as 'login', 'getty', or 'passwd'. Anyway, here are the important lines as follows:

```
CPU=$(shell uname −m | sed s/I.86/intel/)

LOCALEDIR=/usr/share/locale

HAVE_PAM=no

HAVE_SHADOW=yes

HAVE_PASSWD=yes

REQUIRE_PASSWORD=yes

ONLY_LISTED_SHELLS=yes

HAVE_SYSVINIT=yes

HAVE_SYSVINIT_UTILS=yes

HAVE_GETTY=yes

USE_TTY_GROUP=yes

HAVE_RESET=yes

HAVE_SLN=yes

CC=gcc
```

Suggestions:

Note that you could use any of the eight loopback devices, from 'dev/loop0' to '/dev/loop7'. Use an inconspicuous directory for the mount point. I would suggest creating a folder with 700 permissions inside your home folder. The same goes for the file that holds the data. I use a filename like 'sysfile' or 'config.data' inside the '/etc' folder. This will usually get overlooked.

I created very simple Perl scripts to mount and unmount the filesystem with one command. Write these, make them executable (chmod u+x), and store them somewhere in your path.

```
#!/usr/bin/perl −w
#
#minimal utility to setup loopback encryption filesystem
#Copyright 1999 by Ryan T. Rhea
```

```
`losetup -e serpent /dev/loop0 /etc/cryptfile`;
`mount /mnt/crypt`;
```

Name the above script 'loop', and then you can be on your way with one command ('loop') and a password.

```
#!/usr/bin/perl -w
#
#minimal utility to deactivate loopback encryption filesystem
#Copyright 1999 by Ryan T. Rhea
`umount /mount/crypt`;
`losetup -d /dev/loop0`;
```

Name the second one 'unloop', and then typing 'unloop' will quickly deactivate your filesystem.