The Beginners Guide to eCryptfs 0.1

Michael C. Thompson

December 12, 2007

Introduction - So What Is This eCryptfs Thing Anyways?

eCryptfs is a stacked filesystem which allows you to have encrypted files on your hard drive without having to reformat. eCryptfs being a stack file system means that in order to mount eCryptfs, all you need is an existing host directory on an existing filesystem, which will be where eCryptfs stores the encrypted versions of your files, and an existing target directory, which will be the eCryptfs mount point and the window through which you can access your encrypted files.

How eCryptfs Uses Your Existing Filesystem

You can imagine using eCryptfs as a three layered system. The bottom most layer is your hard drive. The middle layer is a formatted filesystem of your choice (say ext3). The top layer is of course eCryptfs, which uses two directories in the filesystem, one for the eCryptfs encrypted data (the **host** directory), and one for the decrypted view of that data (the **target** directory). These two directories will parallel each other in structure since the **target** directory is the decrypted view of the **host** directory.

Mounting eCryptfs

eCryptfs is currently under development, with future versions being backwards compatable. Currently, eCryptfs is in version 0.1, and you can read about the various ways eCryptfs can be mounted in the appropriate subsection.

Mounting eCryptfs v0.1

eCryptfs v0.1 currently supports only one authentication option: mount-wide passphrase. This is provided as the mount option "-o passphrase=12345abcde" Additional parameters to this are as follows: -o salt=1234567890abcdef

Writing Data And Getting It Back

As stated in the previous section, there are two directories which eCryptfs uses on your file system, the **host** directory and **target** directory. The **host** directory is only used to store the encrypted versions of the files you make through eCryptfs, you should not access these files directly, as changing them will break them!

The **target** directory is where you will want to do your work, because that provides the layer of eCryptfs functionality which will handle all of the encryption and decryption of files automatically.

Inside the **target** directory, you can do a simple test to make sure that what you write, is what you'll get back. Here is an example: # echo "This is my eCryptfs read/write test" ¿ file # cat file

You should get back what you wrote. That is all there is to it.

Cryptographic Operations - How eCryptfs Does Its Magic

eCryptfs uses multiple layers of encryption to protect the confidentiality of your data. Each file is stored as a string of blocks on the hard drive (usually about 4K in size), and each of these blocks in the file, which will vary based on the size of the file, is individually encrypted with a common special key that is generated randomly for each file. That randomly generated key which was used to encrypt the data is then encrypted by a secret (such as a different key or passphrase) you provide, and written to disk. This allows you to retrieve the key used to encrypt the data when it is time to decrypt, and if your key or passphrase is truly secret, your data can only be decrypted by you.

Using Mount-Wide Passphrase

Mount-wide passphrase was introduced in eCryptfs version 0.1. For more information on using this to encrypt your data, see the "Mounting eCryptfs v0.1" section for details.

Mount-wide passphrase is exactly as it sounds, when you mount eCryptfs, you provide it with a single passphrase which is then used to encrypt all new files you create, as well as used when you try to open any existing files. It is highly advised that you also provide a salt along with the password, which will help make an attack against your files harder than if you use the default salt.

File Format & Other Detailed Information

If you would like to know more details about how eCryptfs works, please consult the design documentation which should be coupled with the source code.

References

[1] See `http://www.linuxsymposium.org/2006/proceedings.php`.

[2] See ecryptfs/doc/design_doc/ecryptfs_design_doc.tex