NAME

eCryptfs − an enterprise-class cryptographic filesystem for linux

SYNOPSIS

mount -t ecryptfs [SRC DIR] [DST DIR] -o [OPTIONS]

DESCRIPTION

eCryptfs is a POSIX-compliant enterprise-class stacked cryptographic filesystem for Linux. It is derived from Erez Zadok's Cryptfs, implemented through the FiST framework for generating stacked filesystems. eCryptfs extends Cryptfs to provide advanced key management and policy features. eCryptfs stores cryptographic metadata in the header of each file written, so that encrypted files can be copied between hosts; the file will be decryptable with the proper key, and there is no need to keep track of any additional information aside from what is already in the encrypted file itself. Think of eCryptfs as a sort of "gnupgfs."

OPTIONS

KERNEL OPTIONS

Parameters that apply to the eCryptfs kernel module.

ecryptfs_sig=(fekek_sig)
Specify the signature of the mount wide authentication token. The authentication token must be in the kernel keyring before the mount is performed. ecryptfs-manager or the eCryptfs mount helper can be used to construct the authentication token and add it to the keyring prior to mounting.

ecryptfs_fnek_sig=(fnek_sig)
Specify the signature of the mount wide authentication token used for filename crypto. The authentication must be in the kernel keyring before mounting.

ecryptfs_cipher=(cipher)
Specify the symmetric cipher to be used on a per file basis

ecryptfs_key_bytes=(key_bytes)
Specify the keysize to be used with the selected cipher. If the cipher only has one keysize the keysize does not need to be specified.

ecryptfs_passthrough
Allows for non-eCryptfs files to be read and written from within an eCryptfs mount. This option is turned off by default.

no_sig_cache
Do not check the mount key signature against the values in the user's ˜/.ecryptfs/sig-cache.txt file. This is useful for such things as non-interactive setup scripts, so that the mount helper does not stop and prompt the user in the event that the key sig is not in the cache.

ecryptfs_encrypted_view
This option provides a unified encrypted file format of the eCryptfs files in the lower mount point. Currently, it is only useful if the lower mount point contains files with the metadata stored in the extended attribute. Upon a file read in the upper mount point, the encrypted version of the file will be presented with the metadata in the file header instead of the xattr. Files cannot be opened for writing when this option is enabled.

ecryptfs_xattr
Store the metadata in the extended attribute of the lower files rather than the header region of the lower files.

verbose
Log ecryptfs information to /var/log/messages. Do not run eCryptfs in verbose-mode unless you are doing so for the sole purpose of development, since secret values will be written out to the

system log in that case.

MOUNT HELPER OPTIONS

Parameters that apply to the eCryptfs mount helper.

key=(keytype):[KEY MODULE OPTIONS]
Specify the type of key to be used when mounting eCryptfs.

ecryptfs_enable_filename_crypto=(y/n)
Specify whether filename encryption should be enabled. If not, the mount helper will not prompt the user for the filename encryption key signature (default).

verbosity=0/1
If verbosity=1, the mount helper will ask you for missing values (default). Otherwise, if verbosity=0, it will not ask for missing values and will fail if required values are omitted.

KEY MODULE OPTIONS

Parameters that apply to individual key modules have the alias for the key module in the prefix of the parameter name. Key modules are pluggable, and which key modules are available on any given system is dependent upon whatever happens to be installed in /usr/lib*/ecryptfs/.

passphrase_passwd=(passphrase)
The actual password is passphrase. Since the password is visible to utilities (like ps under Unix) this form should only be used where security is not important.

passphrase_passwd_file=(filename)
The password should be specified in a file with passphrase_passwd_file=(passphrase). It is highly recommended that the file be stored on a secure medium such as a personal usb key.

passphrase_passwd_fd=(file descriptor)
The password is specified through the specified file descriptor.

passphrase_salt=(hex value)
The salt should be specified as a 16 digit hex value.

openssl_keyfile=(filename)
The filename should be the filename of a file containing an RSA SSL key.

openssl_passwd_file=(filename)
The password should be specified in a file with openssl_passwd=(openssl-password). It is highly recommended that the file be stored on a secure medium such as a personal usb key.

openssl_passwd_fd=(file descriptor)
The password is specified through the specified file descriptor.

openssl_passwd=(password)
The password can be specified on the command line. Since the password is visible in the process list, it is highly recommended to use this option only for testing purposes.

EXAMPLE

The following command will layover mount eCryptfs on /secret with a passphrase contained in a file stored on secure media mounted at /mnt/usb/.

mount -t ecryptfs -o key=passphrase:passphrase_passwd_file=/mnt/usb/file.txt /secret /secret

Where file.txt contains the contents **"passphrase_passwd=[passphrase]"**.

SEE ALSO

mount(8)

/usr/share/doc/ecryptfs-utils/ecryptfs-faq.html

http://ecryptfs.org/

NOTES

Do not run eCryptfs in verbose-mode unless you are doing so for the sole purpose of development, since secret values will be written out to the system log in that case. Make certain that your eCryptfs mount covers all locations where your applications may write sensitive data. In addition, use dm-crypt to encrypt your swap space with a random key on boot, or see **ecryptfs-setup-swap**(1).

Passphrases have a maximum length of 64 characters.

BUGS

Please post bug reports to the eCryptfs bug tracker on Launchpad.net: https://bugs.launchpad.net/ecryptfs/+filebug.

For kernel bugs, please follow the procedure detailed in Documentation/oops-tracing.txt to help us figure out what is happening.

AUTHOR

This manpage was (re-)written by Dustin Kirkland <kirkland@ubuntu.com> for Ubuntu systems (but may be used by others). Permission is granted to copy, distribute and/or modify this document under the terms of the GNU General Public License, Version 2 or any later version published by the Free Software Foundation.

On Debian systems, the complete text of the GNU General Public License can be found in /usr/share/common-licenses/GPL.