CifrarFS – Encrypted File System Using FUSE

Anagha Kulkarni kulkarniar07@comp.coep.org.in
Department of Computer Engineering
and Information Technology,
College of Engineering,
Pune, 411005, India

Vandana Inamdar vhj@comp.coep.org.in
Department of Computer Engineering
and Information Technology,
College of Engineering,
Pune, 411005, India

Abstract

This paper describes a file system that enables transparent encryption and decryption of files by using advanced, standard cryptographic algorithm, Data Encryption Standard (DES) [1]. Any individual, including super user, or program, that doesn't possess the appropriate passphrase for the directory which contains encrypted files cannot read the encrypted data. Encrypted files can be protected even from those who gain physical possession of the computer on which files reside [2].

‘CifrarFS’, an encrypted file system using ‘File system in USEr space (FUSE)’ maintains all the files in a specific directory in an encrypted form and decrypts them on demand. It encodes the file name while storing but decodes it while viewed from the mount point. We propose an idea of watermark in every encrypted file that is validated before decryption and also log all the operations on ‘CifrarFS’. It is a stackable file system that operates on top of ext3. It does not need root privileges.

Keywords: File system, Operating System, Cryptography, Security, FUSE.

1. INTRODUCTION

File System is the only module of the operating system that is most visible to the user. It deals with the easy storage and fast retrieval of the data without user actually knowing the details of the storage device [3].

There is an increasing opportunity to use Linux in enterprise systems, where the users expect very high security. Amount of information that is being stored on computer systems is increasing and therefore, ensuring the security of this information is very important for all businesses today. Access control mechanism is the most common security mechanism employed in operating systems [4]. If this security mechanism is broken, the whole data is exposed to the unauthorized user. Even the super user of the system has access to all the data. If he makes improper use of his authorities, the whole data is exposed. The users may need an additional level of security to protect their sensitive data.

So, there is a need of such a mechanism that gives the user an ability to make his documents not accessible to unauthorized users. This will generate a trust in the mind of the user that his data cannot be accessed by anybody except himself. Use of cryptographic techniques is a promising way to offer extra security to user's documents. Many standard cryptographic algorithms (such as Data Encryption Standard (DES) [1], Advanced Encryption Standard (AES) [5], and International Data Encryption Algorithm (IDEA) [6] etc) are available and are strong and speedy. They encrypt the data using user supplied key and give assurance that encrypted data cannot be decrypted unless the correct key is supplied. These algorithms are commonly used while data is being transmitted from one computer to another on a network where the data could be hijacked by third person and could be modified, replayed or just tapped [7]. However, the use of these algorithms is not so common for protecting user data which is permanently stored on the storage device.

In this paper, we introduce a scheme, called 'CifrarFS'. This system merges the secure technique of 'Cryptography' with an important component of the operating system, the 'File System'. It is not a basic file system like ext2, ext3 but is a stackable file system that works on top of underlying file system [8]. 'CifrarFS' operates in user space and provides an encryption/decryption engine, making all system calls to a specific directory pass through this engine, so that, the data is encrypted before storing on the storage device and decrypted before accessing through user-specified mount point.

2. RELATED WORK

Cryptography can be employed as a basic part of file systems at kernel level. There exist many file systems in UNIX that use cryptographic techniques for protecting a file or a complete file system, like Reiser4, CFS, CryptFS etc. In practice, encryption is carried out at different levels. These levels are as follows.

Low Level Encryption

Reiser4 [9] is an advanced file system that gives exotic security support in terms of encryption using flexible plug-in infrastructure at source level. The main problem is that users who wish to use the cryptographic features are confined to a specific file system i.e. Reiser4. Extra security cannot be offered on demand for an existing file system or only for a specific directory.

Middle Level Encryption

Cryptographic File System (CFS) [10] is designed on the principles that trusted components should do the encryption of data before sending it to untrusted components. It lies in between user level and file system level. It operates by pushing encryption using DES into client side file system interface and protects those aspects of file storage that are vulnerable to attack.

CryptFS [11] is a stackable 'vnode' level encryption file system that can be implemented on modern operating systems without having to change the rest of the system. The file system interposes (mounts) itself on top of any directory, encrypts file data using Blowfish before it is passed to the interposed-upon file system, and decrypts it in the reverse direction.

EncryptFS [12], a versatile cryptographic file system for Linux, provides dynamic data encryption and decryption at system level. It works on Linux Virtual File System (VFS) layer. It uses a symmetric key algorithm AES to encrypt file contents and a public key algorithm, Rivest, Shamir, Adleman (RSA) to encrypt the key that symmetric key algorithm uses.

User Level Encryption

eCryptFS [13] is an extension of CryptFS. It integrates kernel cryptographic Application Program Interface (API), kernel key ring, Pluggable Authentication Modules (PAM), Trusted Platform Module (TPM) and GNU Privacy Guard (GnuPG). It stores metadata directly in the files.

EncFS [14] is a free FUSE-based cryptographic file system that transparently encrypts files, using an arbitrary directory as storage for the encrypted files. It uses AES for encryption and decryption. It implements primary and secondary file systems by having two different passwords. If it is unable to decrypt a file with a volume key, that file is ignored. If it is forced to ignore the password, it decodes the key differently and hence files are encrypted and decrypted with a different key. This allows two different encrypted volumes for two different passwords.

3. DESIGN OF 'CifrarFS'

'CifrarFS' is a convenient file system that offers an extra security to the user, in addition to regular access control mechanism. It is a file system in User Space and is a virtual or stackable file system. It maintains all the files in a specific directory in an encrypted form and decrypts them on demand. In other words, when a specific file is being viewed only via mount point, it will be shown as a plain text, but it will always be stored in an encrypted form in the underlying file system.

Design Goals

'CifrarFS' has four major goals:

- Security- to secure data from malicious access.

If the person trying to read encrypted documents does not know the passphrase to mount the file system, there is no way he can retrieve the plain text. The salt for passphrase and passphrase in encrypted form is stored into a repository.

- Portability- to be able to install the system when required. (Re-installation of operating system not required).

'CifrarFS' works in the user space. So, the file system can be installed on demand without affecting rest of the working. It operates on top of native file system.

- Ease of use- to give the simple and known interface.

Passphrase is required to be entered once per session. The same passphrase is used by encryption/decryption engine to encrypt and decrypt all the documents. The user has to specify only the passphrase and the mount point every session, and use the file system as his native file system. The user does not need to use any new commands.

- Use of only one key- to use only one key.

Encryption key is required to be supplied only once while mounting the file system. All subsequent operations are associated with this key.

In view of above design goals, functionality of 'CifrarFS' is as follows:

Functionality

An important goal of 'CifrarFS' is to provide the user with second-level security. 'CifrarFS' provides extra security by asking the user to enter a passphrase for mounting the file system only once per session. So the user is not required to enter different passphrase for encrypting every file or same passphrase multiple times for encrypting many files.

'CifrarFS' operates at user level. It makes use of FUSE module. FUSE [15, 16, 17, 18, 19] has three parts:

- Kernel module – which registers with VFS. It resides in kernel space.
- FUSE library – which resides in user space.
- User written file system – which resides in user space. It implements required system calls.

'CifrarFS' is a user written file system that uses Linux semantics. User has to simply attach the directory on which encrypted files reside to the mount point. The user is prompted for the passphrase. If correct passphrase is entered, the file system is mounted on the specified mount point. All the documents on the mounted directory (directory which contains encrypted documents) are now available to the user in the plain text when accessed from mount point. There is no change in any Linux command semantics.

Typically, mounted directory resides in user's home directory. Mount point could be any directory. For example, a directory could be created in '/tmp' for using as a mount point. When a document is created using any Linux command via mount point, it is stored into the mounted directory in encrypted form. File names of such files also are encrypted. The file names and their contents appear as clear texts to the user who is viewing them only via mount point. There is no other way, to view the file names and their contents in clear texts. 'CifrarFS' resides on ext3 and no space is required to be preallocated for the files of 'CifrarFS'.

The passphrase has to be of sufficient length (maximum allowable length is 40 characters) and if the passphrase is disclosed, the whole file system will be under trouble. It may contain any printable character.

'CifrarFS' uses environment variable 'CIFRARFS_PATH' as the directory to be mounted. It can be declared in '.bash_profile'.

The user interface of 'CifrarFS' is very simple, and is shown below:

User Interface

'CifrarFS' can be run using a command, $ CifrarFS –m /tmp/plain. It prompts the user for passphrase which is not echoed. 'CifrarFS' checks salt value of entered passphrase with the one stored in the repository. If it matches, the file system is mounted.

All standard Linux functions work normally. User can see his documents in plain text on /tmp/plain. The files are stored in mounted directory in cipher text with encrypted file name.

4. IMPLEMENTATION OF 'CifrarFS'

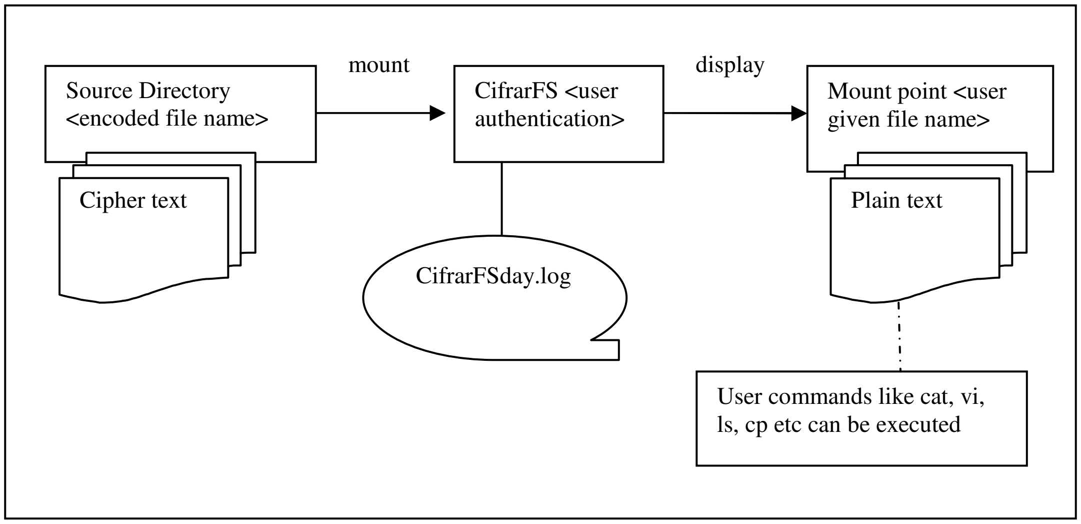
mount
display
Source Directory <encoded file name>
CifrarFS <user authentication>
Mount point <user given file name>
Cipher text
Plain text
CifrarFSday.log
User commands like cat, vi, ls, cp etc can be executed

FIGURE 1: General Architecture of 'CifrarFS'.

'CifrarFS' is implemented on Fedora (kernel 2.6.x) using FUSE (2.4.x). Figure 1 shows general architecture of 'CifrarFS'. As shown, source directory contains encrypted files whose names are encoded. Upon mounting 'CifrarFS' on a mount point, the files can be viewed/accessed in plain text form from the mount point. 'CifrarFS' also maintains a log of all operations.

File Viewing

When a document is being read via mount point, 'CifrarFS' does the following:

- Encrypts the user entered clear text file name
- Checks the watermark.
- Decrypts the encrypted contents of the file.
- Displays the plain text contents of the file.
- Maintains log of the operation.

Directory Listing

When a directory listing is taken on mount point, ‘CifrarFS’ does the following:

- Reads a directory listing one by one.
- Gets metadata of each name.
- Decrypts each name into clear text.
- Displays the clear text name.
- Maintains log of the operation.

Passphrase Generation

Passphrase is generated using crypt () utility, a passphrase encryption function, in Linux. It uses one-way hash function. Seed is a function of time and process id. It consists of 11 characters out of which first three are “1” indicating the use of MD5-based algorithm. Crypt () outputs 34 bytes as shown in figure 2 below, which are stored in ‘.cifrarobj’.

1	Salt - 8 chars	$	Encrypted Passphrase - 22 chars

FIGURE 2: Encrypted Passphrase format.

Salt value - Depends upon process id and time of creation. Contains any characters from “/, ., A..Z, a..z, 0..9”.
Encrypted passphrase - Contains any characters from “/, ., A..Z, a..z, 0..9”.

While verifying the passphrase, salt value is used to re-encrypt the entered passphrase. If it matches with rest of the 22 characters of encrypted passphrase, then passphrase is verified.

Encryption and Decryption of File Name

Encryption of file name is done using transposition. The key, 4 bytes long, used for the encryption and decryption is derived by doing mathematical operations on 8 characters from passphrase.

If length of file name is less than 5, to every character of file name, 1 byte of key is added and encrypted word is derived. If length of file name is greater than 4, encoding is done in groups of 4. This algorithm does not abstract file name length.

Writing and Reading of Watermark

To identify whether a file is encrypted by this system, every file that is being encrypted, contains a watermark. Watermark is generated using same mechanism used for generating passphrase. The only difference is that 22 characters are generated from the word “CifrarFS”.

While displaying the file contents in clear text, watermark is verified, using similar logic for passphrase, but for word “CifrarFS”. Once this verification is done, file is decrypted.

Encryption and Decryption of File Contents

For encrypting file contents DES_cfb64_encrypt () with encrypt flag 0 is used. It needs 8 bytes key schedule and 8 bytes initialization vector (IV). Both are derived from stored passphrase. It takes 64 bit plain text. In cipher feedback mode (CFB), 8 byte IV is encrypted using 8 byte key which is then XORed with 64 bit plain text. The 64 bit cipher text which is generated in current phase is fed as IV for next phase. It encrypts arbitrary number of bytes, without 8-byte padding. For decrypting file contents DES_cfb64_encrypt () with encrypt flag 1 is used.

Saving and Displaying a File

When a document is newly created, user given file name is encrypted. When a document is released, encryption of file contents is done and the document is saved as encrypted name. When existing document is accessed, user provided file name is encrypted to get attributes of encrypted file name.

If user has enough privileges, watermark is verified; the document is decrypted and displayed.

Maintaining Log of the Operations

Log of operations is maintained weekly. ‘CifrarFS’ creates one log file for each day of the week, for example, “CifrarFSsun.log”, “CifrarFSmon.log” etc. When ‘CifrarFS’ is run, it checks the day of the week and builds the name of the file. It then checks if the file with the same name exists. If it does, then it checks whether it is the first session of the day and if so truncates the log file. If not, it just appends the log file with time and current operation.

5. EXPERIMENTAL SETUP AND TEST RESULTS

We tested ‘CifrarFS’ on following setup:

- Fedora Release 8 (Codename: Werewolf)
- Kernel Linux 2.6.23.1-42-fc8
- Hardware: Memory - 248.7 MiB , Peocessor – Intel ®, Pentium ® processor 1400 MHz.

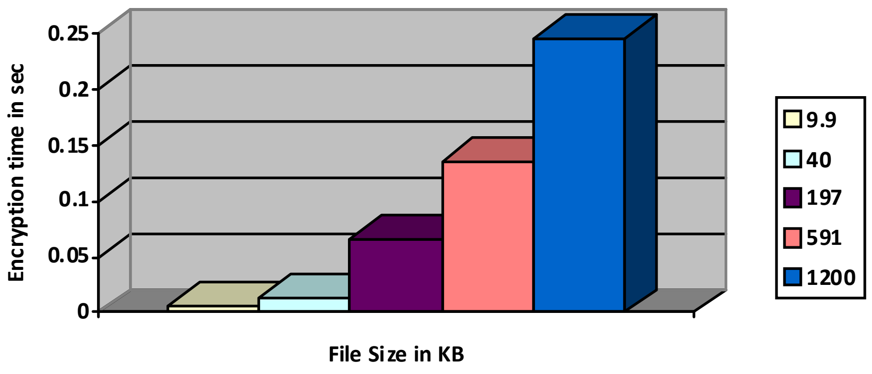

Figure 3: Encryption Time vs File size

We measured the time required for encrypting files of sizes between 9.9 KB and 1200 KB. The graph of encryption time in sec (along Y-axis) vs file size in KB (along X-axis) is as shown in Figure 3.

It can be concluded from the graph that as the file size goes on increasing, the time required for encrypting increases but by the smaller factor than the file size. Thus, once the encryption starts off encryption is faster.

We ran the scripts which created 1000 files using 'touch' command, deleted 1000 files using 'rm' and wrote 100 bytes into 1000 files on ext3 and 'CifrarFS' each. We plotted a graph of time in sec (along Y-axis) and different operations on 'CifrarFS' and ext3 (along X-axis) as shown in figure 4.

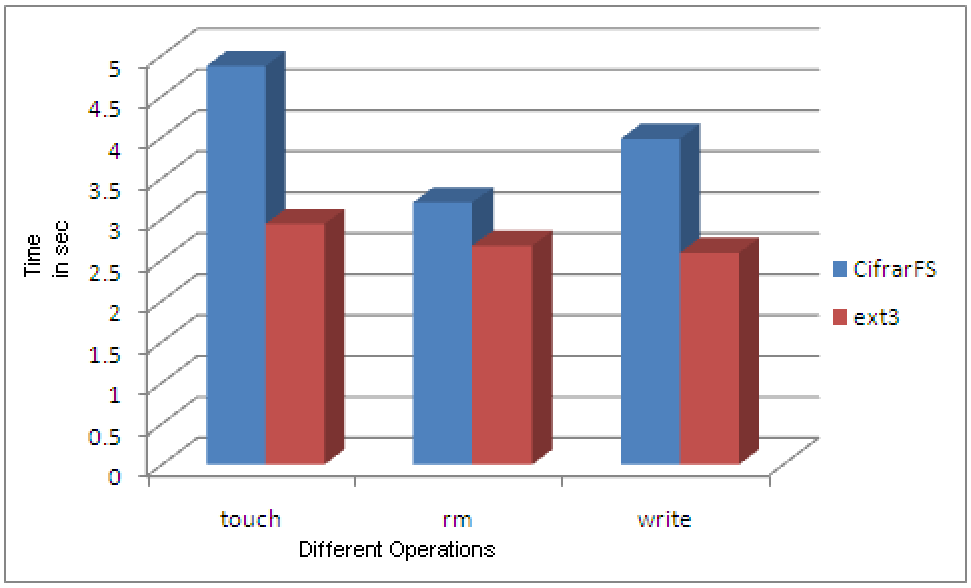

Figure 4: Time vs different operations

It is seen that time required to carry out an operation on 'CifrarFS' is slightly more than that required on ext3. This extra time is due to encryption of data.

6. CONCLUSION AND FUTURE WORK

'CifrarFS' is a user-space, FUSE-based, stackable and virtual file system which encrypts the files in a specific directory and decrypts them on demand. It is an effort to give an extra security to the user against offline attacks. It converts the file name (having printable characters) to encoded file name which may have non-printable characters, thus making it difficult for the attacker to access the encrypted file.

In future we plan to handle integrity of encrypted files in a more efficient way. Also, the security of this file system hangs on one thread – '.cifrarobj'. If it gets deleted, there is no way to recover the encrypted passphrase and so the encrypted files, as the encrypted passphrase depends upon the time of creation and process id. Therefore, regular backup of encrypted files and '.cifrarobj' is required to be taken.

7. REFERENCES

1. **FOR STANDARD:** National Bureau of Standards, Data Encryption Standard, U.S. Department of Commerce, FIPS Publication 46, Jan 1977

2. **FOR WEBSITE:** Roberta Bragg. "*The Encrypting File System*". http://technet.microsoft.com/en-us/library/cc700811.aspx

3. **FOR BOOK:** Avi Silberschatz, Peter Galvin, Greg Gagne: "*Operating System Concepts*", John Wiley and Sons, Inc, Sixth Edition

4. **FOR CONFERENCE:** HweeHwa Pang, Kian-Lee Tan and Xuan Zhou. "*StegFS: A Steganographic File System*", IEEE International Conference on Data Engineering, Mar 2003

5. **FOR STANDARD:** National Bureau of Standards, Data Encryption Standard, U.S. Department of Commerce, FIPS Publication 197, Nov 2001

6. **FOR STANDARD:** Xuejia Lai and James Massey. "*A proposal for a New Block Encryption Standard*", 1990

7. **FOR BOOK:** William Stallings. "*Operating Systems: Internals and Design Principles*", Prentice Hall, Fifth Edition

8. **FOR CONFERENCE:** Ion Badulescu and Erez Zadok. "*A Stackable File system Interface for Linux*", LinuxExpo Conference Proceedings in 1990

9. **FOR WEBSITE: "***ReiserFS - Using Resier4 with Linux*", http://www.ibm.com/developerworks/aix/library/au-unix-reiserFS/

10. **FOR CONFERENCE:** Matt Blaze. "*CFS – A Cryptographic File System for UNIX*", First ACM Conference on Computer and Communications Security, 1993

11. **FOR REPORT:** Erez Zadok, Ion Badulescu and Alex Shender. "*CryptFS: A Stackable Vnode Level Encryption File System*", Technical Report CUCS-021-98, June 1998

12. **FOR WEBSITE:** "*EncryptFS: A Versatile Cryptographic File System for Linux*", pompone.cs.ucsb.edu/~wei/EncryptFS.pdf

13. **FOR WEBSITE:** Michael Halcrow. "*eCryptFS An Enterprise Class Cryptographic file system for Linux*", http://ecryptfs.sourceforge.net/ecryptfs.pdf

14. **FOR PRESENTATION:** Valient Gough. "*EncFS*", Libre Software Meeting, France, 2005. http://www.arg0.net/encfsintro

15. **FOR WEBSITE: "***FUSE operations*", http://www.soe.ucsc.edu/~aneeman/FUSE_how-to.html

16. **FOR WEBSITE: "***FUSE Documentation*", http://www.prism.uvsq.fr

17. **FOR WEBSITE: "***Introduction to FUSE and Working of FUSE*", http://fuse.sourceforge.net/

18. **FOR WEBSITE: "***Implementation of Simple File System Using FUSE*", http://fuse.sourceforge.net/helloworld .html

19. **FOR CONFERENCE:** Antti Kantee and Alistair Crooks. "*ReFUSE: Userspace FUSE implementation using puffs*", EuroBSDCon 2007, 2007