GLOBAL JOURNAL OF ENGINEERING, DESIGN & TECHNOLOGY
(Published By: Global Institute for Research & Education)
www.gifre.org

Enhancing File Security by Integrating Steganography Technique in Linux Kernel

R.Jegadeesan [1], Dr.N.Sankar Ram[2], M.S.Tharani[3]
Ph.D Research Scholar [1], Supervisor[2], P.G.Scholar [3]
Anna University-Chennai (India) [1], Anna University. Chennai (India) [2],
S.R.M University, Chennai (India) [3]
ramjaganjagan@gmail.com [1], n_sankarram@yahoo.com[2],tharani_vig@yahoo.com[3]

ABSTRACT
In today's world securing file data is very important. The proposed Secure File System (SFS) provides file data security using steganographic techniques in a transparent and convenient way. The proposed SFS pushes information hiding services into the Linux kernel space, mounting it between the Virtual File System layer and underlying file system. After SFS is integrated with the Linux operating system (OS), it enables OS to provide File Data Security as its inherent functionality. SFS requires that the user creates a directory and name it with the prefix 'secrt' to store the encrypted file data, such as secrtdir. Any directory on the system with the prefix 'secrt' will basically tells the system that the newly created directory will contain secret data. All files destined to be saved on this directory will be transparently hidden in non-suspicious information on the fly without any user intervention. For hiding we use a steganographic technique in which SFS is fully compatible with all underlying storage file systems. This paper illustrates the design of SFS for Linux which extends the operating system to provide file data security as its inherent functionality.

Categories and Subject Descriptors
D.4.3 [**operating systems**]: File Systems Management-*File organization, Access methods*, Security and Protection

General Terms

Algorithms, Performance, Design, Reliability, Security,

Keywords
Steganography, File data security, Fie system

1. INTRODUCTION

Dr.Abdul Kalam Told to Bill Gates

"In November 2002, Microsoft Chairman Bill Gates met with India's President APJ Abdul Kalam. At this meeting, Kalam advocated for open-source software codes [GNU/Linux]. Among other things, Kalam worries that most people in developing countries such as India can't afford commercial systems and software. "[1] [2]

With this insight from Dr.A.P.J.Abdul Kalam, this project is going to make an additional facility to Linux platform which is the official operating system of Tamilnadu [SuSe Linux], Kerala, Gujarat, Indian Military and ISRO. In today's world, file security is an important factor where each and every record of government or other high-level organizations are stored in the computer database. This information has to be secured properly. In order to protect this highly sensitive information from hackers, terrorist, and other unofficial intruders, it is necessary to build a file security system that cannot be attacked by them. [3]

In computer systems information is stored traditionally in the form of files. File is considered as a basic entity for keeping the information. In UNIX like systems, the concept of file is so important that almost all input/output devices are considered as a file. Therefore the problem of securing data or information on computer systems can be defined as the problem of securing file data. It is a well accepted fact that *securing file data is very important*, in modern computing environment.

There are various approaches available to ensure file data security, such as encryption tools like 'aescrypt' in Linux or integrated encryption application software or disk encrypter. But each one has its own inherent disadvantages, rendering them being less frequently used. These approaches are generally cumbersome and inconvenient to the users. Therefore, there is a need for a mechanism/system which can ensure reliable and efficient file data security in a transparent and convenient manner. [4]

So, we propose our system with highly reliable file security options that is integrated with secured Linux kernel, using steganography with Biometric finger print recognition (can be extended to any biometric recognition such ad Iris, Voice, Face, etc.) as stego key.

2. RELATED WORK
There have been different approaches used, to solve the file data security problem. Most of the solutions provided works in user space. The simple and naive approach used by many people to secure their file data is to use common utilities like 'crypt' or 'aescrypt'. These utilities take the filename and the password as inputs and produce the encrypted file. This type of utility is good for limited use only, as it is very cumbersome and manual. Second approach is integrating encryption engine in application software itself, where each program that is to manipulate

sensitive data has built-in cryptographic facilities. But the disadvantage here is that all application should use the same encryption engine and any change in one will require changes in all. The third approach is to use commercially available disk controllers with embedded encryption hardware that can be used to encipher entire disks or individual file blocks with a specified block. It suffers from the fact that key needs to be shared among users, whose data reside on the disk because entire disk is protected as a single entity. It is good for single user system but for multi-user system the key protecting the data needs to be shared between different users.

So we have seen that each one of the approaches described above; has its own inherent disadvantages, rendering them less frequently used. These approaches are generally cumbersome and inconvenient to the users. Therefore, there is a need for a mechanism/system which can ensure reliable and efficient file data security in a transparent and convenient manner. We focused on this issue and proposed SFS that solves the file data security problem. We considered various places where this mechanism/system can be placed to fulfill its requirement in the best possible way. The considered places include user space, device layer level, and kernel space. We are of the opinion that the file data security should be provided as a functionality of operating system, therefore we have decided to push the encryption services into the Linux kernel space mounted beneath the virtual file system.

2.1 Cryptographic File System (CFS)

Cryptographic File System was developed by Matt Blaze, to provide a transparent UNIX file system interface to directory hierarchies that are automatically encrypted with user supplied keys. CFS is implemented as a user level NFS server. User needs to create an encrypted directory and assign its key required for cryptographic transformations, when the directory is created for the first time. In order to use an encrypted directory, CFS daemon requires the user to attach the encrypted directory to a special directory '/crypt'. This attach basically creates a mapping between the encrypted directory and mount point (directory) in the '/crypt'. This way the actual encrypted data resides in the encrypted directory and the mapping provides a window to access these encrypted file in clear text form to the authenticated user. CFS uses Data Encryption Standard (DES) to encrypt file data. The CFS prototype is implemented entirely at user level, communicating with the UNIX kernel via the NFS interface. Its main disadvantage is that it runs in user mode, thus requires many context switches and data copies from user space to kernel space. [5]

2.2 Transparent Cryptographic File System (TCFS)

TCFS works as a layer under the VFS (Virtual File system) layer, making it completely transparent to the applications. The security is guaranteed by means of the DES algorithm. TCFS is implemented as a NFS distributed file system. The TCFS daemon handles the RPC generated by the kernel. RPC relative to read, write etc have been extended to perform security operations. Each time a new file handler is created, the extended attribute 'secure' is tested. If the file is secure, then all successive read and write operation will be filtered through the encryption/decryption layer. In TCFS for file encryption, each user is associated with a file system key and all files of a user are encrypted using this key. This key is encrypted with user's login password and is stored in a database in '/etc/tcfspwdb'. This dependability of user key on login password is one of the major disadvantages of TCFS. Also we are of the opinion that storing encryption key on the same disk containing data reduces security. [6]

3. PROPOSED SECURE FILE SYSTEM

3.1 Design Goals

We have designed Secure File System (SFS) with the aim that file data security should be provided as one of the primary functionality of the kernel. We have extended the kernel to provide file data security using steganographic techniques as one of its functionality. The hiding / retrieving of file data are performed transparently, making it convenient for the users.

The proposed SFS is designed with the following primary objectives:

- Security: Confidentiality of data is ensured by use of strong steganographic technique. The files are hidden on the fly and then saved to the disk or sent on the network.
- Strong Access Control: We have also used Stego-key as fingerprint, to control the access of the file. This approach enhances the security of file by avoiding unwarranted access.
- Transparent Performance: The secret message containing files should behave no different from some other files.
- Convenience: The system should be convenient to users.

3.2 Design of the Proposed System

The proposed Secure File System is designed to provide the above mentioned goals. Figure 1 shows the normal flow of control in standard file system. VFS shown in figure 1 has namely two main functions. [7]

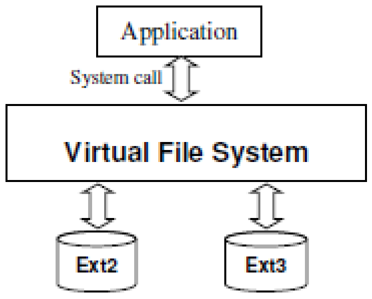

Figure 1: Control flow in standard file system

First, to handle the file system related system calls like open, close, read, write etc. Secondly it provides a uniform interface to

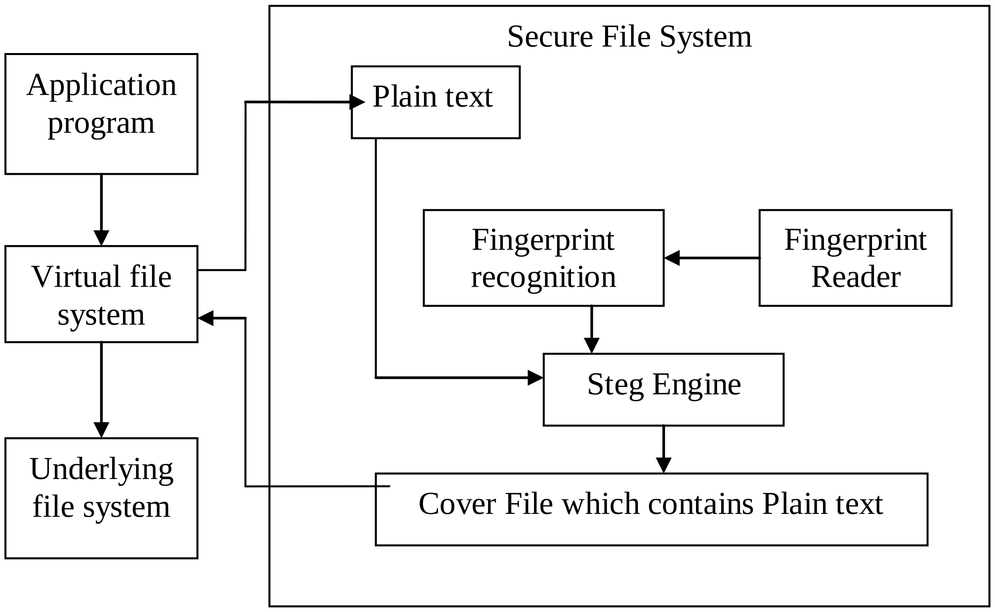

Figure 2: Secure File System

actual file systems like ext2, ext3, FAT, etc, by acting as a switch. In our design, we have taken the control flow from VFS layer based on some condition and rerouted it to proposed SFS.The condition that is checked is the location where the file is destined to be saved. If the location is the directory starting with prefix word 'secrt' (e.g., secrtdir) then we take the control flow to SFS layer. Figure 2 shows SFS layer is mounted beneath the VFS and interacts closely with it. VFS and proposed SFS functions in kernel space, therefore a user cannot access them directly. The above condition will be checked and executed by kernel.

Mounting SFS beneath VFS is the idle place because then we can efficiently use the kernel infrastructure and deviate only where it is required. In this way, it achieves one major advantage that, it provides uniform interface to all the application and the underlying file system. This means SFS transparently handles the data without being bothered of, from which application the data is coming. Therefore SFS is compatible and works with all the applications. The user can use the steganographic strength provided by SFS with any and every application.

3.3 Architecture of Secure File System:

Figure 2 shows the architecture of Secure File System in detail. SFS mainly has four components

- Fingerprint recognition
- Steg Engine
- File Extractor

3.3.1 Fingerprint recognition:

Fingerprint recognition or fingerprint authentication refers to the automated method of verifying a match between two human fingerprints. Fingerprints are one of many forms of biometrics used to identify an individual and verify their identity.

Pattern based algorithms compare the basic fingerprint patterns (arch, whorl, and loop) between a previously stored template and a candidate fingerprint. This requires that the images be aligned in the same orientation. To do this, the algorithm finds a central point in the fingerprint image and centers on that. In a pattern-based algorithm, the template contains the type, size, and orientation of patterns within the aligned fingerprint image. The candidate fingerprint image is graphically compared with the template to determine the degree to which they match. [8]

If the Fingerprint Recognition is successful, the reader sends the acceptance to the Steg-Engine to start hiding process. In the opposite side, during the extraction of the file, once again the fingerprint of the user is verified with the stored fingerprint associated to the owner of that file.

3.3.2 Steg Engine

Hiding A File:

When an existing file in the file system has to be hidden steg-Engine asks for the filename and the password. The filename is encrypted with the password and a signature is formed. Our ‘inode’ structure will look as shown. It may modify it later for more security

```
Struct steg_hidden_file
{
        Char signature[EXT2_NAME_LEN];
        Struct ext2_inode inode;
}
```

The whole structure will be encrypted and stored. The procedure for hiding will be as follows and the figure illustrate the whole concept.

A random block number will be generated from the filename and password. An example for such a generation would be to take the add and subtract alternate squares of characters from the filename and password and then multiply these values. Then it is divided by the total number of blocks and the remainder is taken as the random block where the steg-inode will be placed. But if this block is already allocated then the next block will be taken and this will go on until an unused block is found.

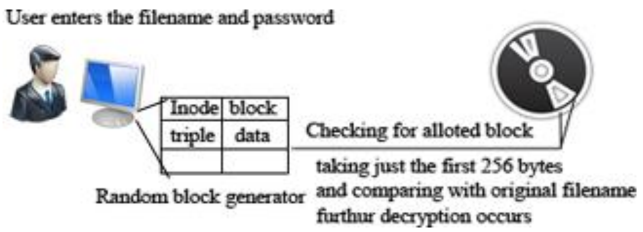
User enters the filename and password
Inode triple
block data
Checking for alloted block
taking just the first 256 bytes and comparing with original filename furthur decryption occurs
Random block generator

Figure 3: Steg- Engine which hides data

Then random blocks are generated for each block that is needed by the file and stored in the appropriate indexes of the inode upto triple indirection. The bitmap will be set for each of the used blocks. But no other changes to the filesystem are made. The superblocks and the group descriptors and the inodes remain as it was before. The number of free blocks will not be altered anywhere. Note that apart from setting the bitmaps, no change in any of the inodes, directories, group descriptors etc are made.

At the end of all this the original file is destructively deleted if it resides in an ext2/3 partition.

3.3.3 File Extractor

The process of File Extractor is to unhide the file to view. This is explained as follows. The user will be asked to enter the filename and the password and the random block will be generated using the above method. Now each block from this random block is checked to see whether it is allocated or not. Once it encounters an allocated block it decrypts the signature i.e. by taking just the first 256 bytes ad it is compared with the original filename. If they mach then that is the correct inode. Then the remaining blocks of the file are got by decrypting the inode.

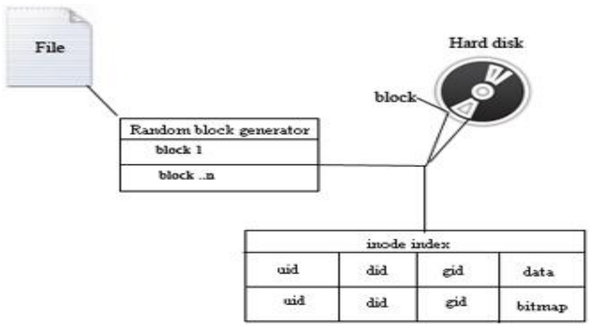

Figure 4. Unhiding of data

Figure 4 shows the File Extractor process. There will also be 2 modes of unhiding the file.

Permanent Unhiding:-In this mode the hidden file will be mounted as a non hidden file in the file system ie. A normal file. This means that it is no longer hidden unless the hiding process is repeated again.

Temporary Unhiding:-In this mode the hidden file will be temporarily made visible for a period of time after which it will return back to its original hidden state without leaving any trace. It will not be mounted anywhere on the hard disk but only in main memory (may be only a part of the file). This means that the file can be made visible before unhiding only if you have the filename and password.

4 SFS OPERATION

In this section we will describe the sequence of events which takes place while the file is being created or written to the disk and also while the file is being modified or read from the disk.

4.1 File Creation

For working with confidential data or files the user needs to enter the secure session by Fingerprint recognition. Then he needs to create a directory with prefix 'secrt' e.g. secrtdir, which will house all the files containing confidential data. After these preliminary requirements are met, the user is free to use any application to create his file. For example, the user may use KWrite utility to create a text file. When the user is finished with the file, he needs to save the file in his newly created directory (secrtdir) which will be housing all such files. This is an important step because SFS will get activated for all the files saved in directory starting with word 'secrt'. The save command issued from the application activates the system call sys_write to write the application data on the disk. It is known that all file system related system calls are handled by VFS, so the control reaches to the VFS. Here the location where file is being saved will be checked, and if the name of the directory is prefixed with 'secrt' is true then the control flow will be transferred to the SFS layer. As shown in the figure 2, Steg Engine will hide the file data. Generated cover file is saved on the disk.

4.2 File Access

For accessing confidential data or files the user needs to enter the secure session by Fingerprint recognition. Now user can open the file with some application, for example KWrite for opening or accessing the text file. The file data will be displayed to the user in plain form if he was given authorization by the owner at the time of file creation else, the file contents will not be displayed (i.e., access denied to unauthorized user). This is done with the help of File Extractor as explained before.

5 IMPLEMENTATION:

SFS layer interacts closely with the virtual file system (VFS) layer. VFS handles all system calls related to file system.

Whenever any file system related system call comes to VFS, we check the location of file being saved. If the file saved, is in a directory which is prefixed by the word 'secrt' (meaning data to be encrypted) the data of the file will be passed through the SFS layer. We transfer the control from VFS to our SFS. The algorithm for which is shown below:

```
vfs_write
{
if (datatobeEncrypted == TRUE)
{
        sfs_write
        {
                call getFingerprint(); /* Get Fingerprint as stego key */
                call stegEngine(); /* Hides the data */
        }
}
file->f_op->write(); /* write function of underlying file system */
}
```

The sfs_write function calls the function getFingerprint function to get the fingerprint of the user to use it as stego-key. Then Steg-Engine hides the data as explained before.

Similarly, while reading/accessing the file data from the disk the following operations are performed:

```
vfs_read
{
        file->f_op->read();       /* read function of underlying file system */
        if (datatobeDecrypted == TRUE)
        {
                sfs_read
                {
                        call fileheaderextractor(); /* Extract the file by reverse process of steganographic algorithm */
                        call getFingerprint();      /* Retrieve KEY from user*/
                        call stegengine();          /* Extract the file data */
                }
        }
}
```

The vfs_read function read the attributes of the file. If it is a secret data identified by its directory, then the control is forwarded to the sfs-read function. The file Extractor then extracts the hidden data using Fingerprint of user as stego-key.

6 CONCLUSION

Our main contribution is in designing and building a Secure File System that was developed with the express goal of enhancing file data security in Linux kernel. The main objective is to provide data security with user convenience. This has been done by implementing SFS in kernel space and enabling steganographic strength on the files on the fly and in a transparent way. We have seen that implementing SFS in kernel enables the operating system to provide file data security as one of its inherent functionality. SFS is very convenient to user as it performs the hiding and extraction transparently and even all system administration tasks like backup, etc are having the same common interface. The scheme guarantees an end to end protection leading to a secure computing environment. We achieved high security by including support for biometric security, designing a strong access control mechanism using fingerprint based access and session entry for accessing confidential data. We achieved high performance by designing SFS to run in kernel.

7 ACKNOWLEDGMENTS

Our thanks to our college faculty for successful finishing of this paper.

8 REFERENCES

[1] Article: Take on Gates, Kalam tells Indian techies available at <http://osdir.com/ml/fsf.india.fsf-friends/2003-05/msg00081.html> last access: 15.09.2009

[2] Article: Take on Gates, Kalam tells Indian techies available at

<http://timesofindia.indiatimes.com/cms.dll/html/uncomp/articleshow?msid=47797903> last access : 15.09.2009

[3] Article : ELCOT's Success Story of Suse Linux Migration Available at <http://goinggnu.wordpress.com/2007/12/26/elcots-success-story-of-suse-linux-migration/>

[4] Aescrypt for linux <http://www.aescrypt.com/linux_aes_crypt.html > last access: 15.09.2009

[5] Matt Blaze AT&T Bell Laboratories. "A Cryptographic File System for Unix" First ACM Conference on Communications and Computing Security, Fairfax, VA, November 3-5, 1993.

[6] Ermelindo Mauriello "TCFS: Transparent yptographic File System" Available at < http://www.tcfs.unisa.it/ >

Last access: 15.09.2009

[7] Braam. Linux Virtual File System available at <http://www.coda.cs.cmu.edu/doc/talks/linuxvfs/sld007.htm Last access: 15.09.2009

[8] Fingerprint recognition Available at < www.biometrics.gov/Documents/FingerprintRec.pdf>

Last access: 15.09.2009

[9] Varun Suresh , Shibin.K , Anoop.S , Vivek.K.P. Magikfs – The Steganographic Filesystem On Linux Available at < http://magikfs.sourceforge.net/>

Last access: 15.09.2009

R.Jegadeesan has registered Ph.D (I&CE) Research in the Anna University Chennai.India on wireles networks and computer Networks 2011 onwards and he has received the B.E.and M.E in computer science and Engineering in Anna University Chennai. India.

During his undergraduate and post graduate studies, he spent time at Anna University Chennai India,From 2000 to 2007, he has contributed at many Universities for visiting faculty for his research area like Wireless networks, Computer Networks, wireless sensor networks. He is currently working with the proposal work in Anna University Chennai. India. He is fellow of IEEE Member for last two years and Computer society of india (CSI)last three years and and RedHAT Linux Instructor and related memberships. in addition that he has Published many National and international level Conferences and Journals, Seminars, and Workshops.

Dr.N. Sankar Ram is the Professor,Computer Science and Engineering, working in a self financing Engineering college affiliated to Anna University,Chennai.. He completed his under graduation and post graduation from Madurai Kamaraj University and Doctorate from Anna University Chennai.

His research areas of expertise are Software Architecture and Computer Networks. He has 16 years of teaching experience and he has published many research papers in International/National journals and conferences. He is a approved supervisor under Anna University of Technology, Chennai and guiding Ph.D and M.E research scholars in his area. Being a Principal Investigator got a funded project to a tune of Rs. 9 Lakhs from AICTE under RPS scheme in the year 2011.

THARANI SUNDARAM has registered UG and PG Chennai.India on wireless sensor networks and computer Networks 2010 and 2014 onwards in computer science and Engineering in Anna University Chennai. And SRM Chennai India.