Implementation of Java Virtual Machine in Cloud

Abstract: A Java File Security System (JFSS) [1] has been implemented in this paper. JFSS is a system file which is encrypted. The reports by the Data Loss DB which is an Open Security Foundation organization has reported many instance of breach of data in the past and also at present, hence we are implementing this on cloud so that the data can be safe. Mainly two software engineering approaches are used to evaluate the JFSS, and one of the approach is Line of Code(LOC) which is a metric of size in the product development cycle of a product. The other approach of JFSS is a methodology called User Satisfaction Test which mainly a customer oriented approach. In today's modern world the satisfaction of the customers is the is the most important element to stay in business. Gaining the loyalty and getting repeated business from customers as well as satisfying and delighting our customers with the quality of the product and the service rendered should be our main aim. Improvement of our process and also the prediction of quality of our software is one of the principle goals of satisfying the customers. User Satisfaction Index allows us to calculate this along with the various other parameters that is involved in customer satisfaction. The best way to determine the quality level of our program is by conducting Customer Satisfaction Surveys.

Keyboard: JFSS, Performance, Evaluation, File Security, File System.

1. INTRODUCTION

The operating system provides much fundamental functionality like storage management, Memory management, process management, and the user interface. They also provide many security functionalities, but it always lacks the mechanism for the file security. When our Java File Security System (JFSS) is integrated once with the operating system, it enhances the file security on demand of the user. The file system is the primary focus of access control in an operating system. The flat file systems make poor secure file systems because there is no way to hide the existence of a file from a user. This approach is very convenient, and user friendly. It is developed in the user space and on the Java technology.

2. Free Java

There are given a lot of free java projects developed in core java, servlet, jsp, struts, spring and hibernate technology.

Features of Java point:

1) All the advance java projects can be downloaded in Eclipse, Myeclipse and Netbeans IDE's.
2) Projects have SRS including Objective of the project, Users of the Project and their Role, Functional Requirement, Non-functional requirement etc.

3) Projects have detailed information of How Project Works?, including Snapshots of the project in the document file with full illustration.
4) For Servlet and JSP projects, you need to create table in the database manually

3. Motivation

The file system is a major component of the operating system. It is a complex piece of software with layers below and above it, all affecting the performance of the system. Developing in-kernel file systems is a challenging task, because of many reasons . These are as follows:

i) The kernel code lacks memory protection,
ii) It requires great attention to use of synchronization primitives,
iii) These can be written in C language only,
iv) Debugging is a tedious task,
v) Errors in the developed file systems can require rebooting the system,
vi) Porting of the kernel file systems requires significant changes in the design and implementation, and
vii) These can be mounted only with super-user privileges. But developing of file systems in user space eliminates all the above issues. At the same time, the research in the area of storage and file systems increasing involves the addition of rich functionalities over the underlying, as opposed to designing the low-level file systems directly. On the other end, by developing in user space, the programmer has a wide range of programming.

4. Related Work

So many approaches are used to solve the problem of file data security. Cryptography is used because the underlying storage provider is not trusted to prevent unauthorized access to data. The technique provides us the security and integrity of the stored data. Cryptographic file systems exists in two forms: either as an enhancement within an existing basic file system that uses an underlying block-storage provider, or as a virtual file system that must be mounted over another (virtual or basic) file system. The first approach is called monolithic cryptographic file systems and the second one is called stackable or layered file systems. The first approach is dedicated to the operating system and second one may be applied on any operating system with no or minor changes. But the stackable file systems are slower than the monolithic file systems. A considerable number of file systems have been developed in the past 20 years. A few of them are briefly described here.

SFS is an MSDOS device driver that encrypts an entire partition. Once encrypted, the driver presents a decrypted view of the encrypted data. This provides the convenient abstraction of a file system, but relying on MSDOS is not secure because MSDOS provides none of the protection of a modern OS.

CFS is a cryptographic file system that is implemented as a user-level NFS server. It requires the user to create a directory on the local or remote file system to store encrypted data. The cipher and key are specified when the directory is first created. The CFS daemon is responsible for providing the owner access to the encrypted data via a special attach command. The daemon, after verifying the user ID and key, creates a directory in the mount point directory that acts as an unencrypted window to the user's encrypted data. Once attached, the user accesses the attached directory like any other directory. CFS is a carefully designed, portable file system with a wide choice of built-in ciphers. Its main problem, however, is performance. Because it runs in user mode, it must perform many context switches and data copies between kernel and user space.

TCFS is a cryptographic file system that is implemented as a modified kernel-mode NFS client. Since it is used in conjunction with an NFS server, TCFS works transparently with the remote file system, eliminating the need for specific attach and detach commands. To encrypt data, a user sets an encrypted attribute on directories and files within the NFS mount point. TCFS integrates with the UNIX authentication system in lieu of requiring separate passphrases. It uses a database in /etc/tcfspwdb to store encrypted user and group keys. Group access to encrypted resources is limited to a subset of the members of a given UNIX group, while allowing for a mechanism (called threshold secret sharing) for reconstructing a group key when a member of a group is no longer available. TCFS has several weaknesses that make it less than ideal for deployment. First, the reliance on login passwords as user keys is not safe. Also, storing encryption keys on disk in a key database further reduces security. Finally, TCFS is available only on systems with Linux kernel 2.2.17 or earlier, limiting its availability.

We have used the Windows XP operating system to design the functionality of file security system. The programming language to be used is the Sun Microsystems Java technology. To design it there is a function design form which has the necessary buttons on it.

The login form, that is used to login with the file security system. After entering the user id and password we are linking to the security execution program. We always need the user registration with the file security system. The registration is done by the program

Administrator who has the only permission to make number of users for the system. He or she will give the username and the password to the user.

That is displayed in Figured 3.1

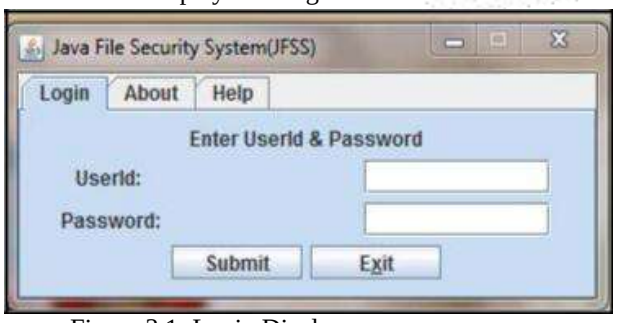

Figure 3.1: Login Display

After this user registration, he or she can login the system and use its functionality. The file encryption, decryption, about and help control form is appeared on the screen. It is shown in the Figure 3.2. It has the option to select the file to which the user wants to encrypt for the security feature. He or she can select any type of file and click on the encrypt button after that the encryption key is saved on the smart card is that is not available then the key is saved on the user specified location.

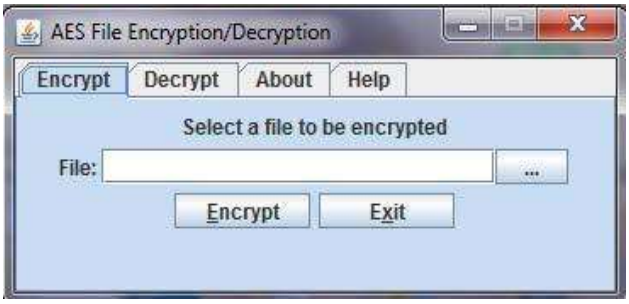

Figure 3.2: Encryption Tab Display

The user may want to decrypt his previously encrypted file to use it. Then he or she have to make two selections one for the file and one for the key especially the encryption key. Then the user will get the message to be successful or unsuccessful decryption. The successful message is shown in the Figure 3.3.

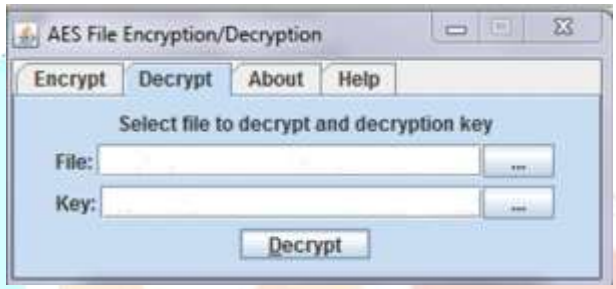

Figure 3.3: Decryption Tab Display

We performed test and evaluation on the proposed file security system for files and the directories. For experiment the computer system was with the configurations as Pentium 4 processor, Windows XP operating system.

The system has been tested for its functioning. In the first login window the user enter his or her use rid and the password. If that is correct then he or she will get a message login successful or not. As in the Figure 4.1 the login is a successful one.

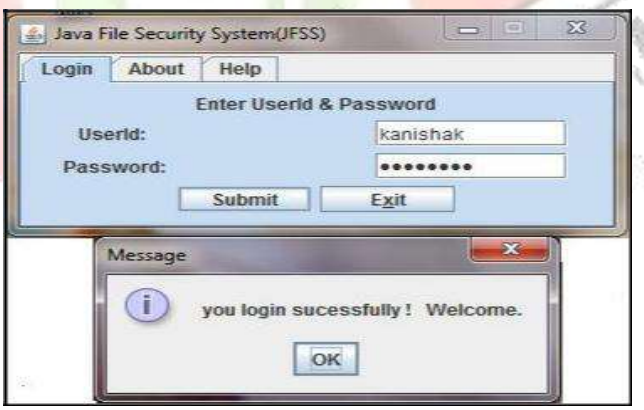

Figure4.2: Login Successful

Encrypts the specified file and save the encryption key to the smart card which is separate location of storage from the encrypt file. It increases security of the data. It shown in figure 4.2.

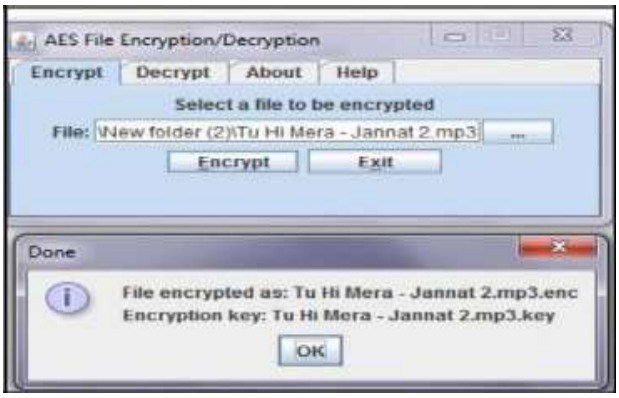

Figure4.2: Encryption Tab Display

This is the Figure 4.3 which shows the decryption process of the system. It has two file selection buttons on it. One file selection button for the specified encrypted file to which the user is going to decrypt. Another one is for the key selection of the specified file. Because every file has its own independent key to encrypt or to decrypt it.

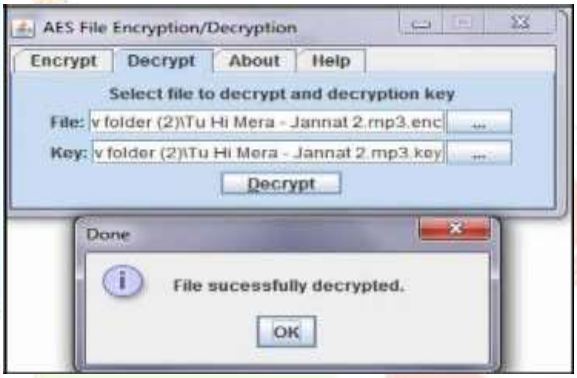

Figure4.3: Decryption Tab Display

We have seen the file's look how it will behave after the encryption. The system is highly secure that we can cont delete the encrypted file and also can't change data which shows the integrity. The encrypted file's view is shown in the Figure 4.4.

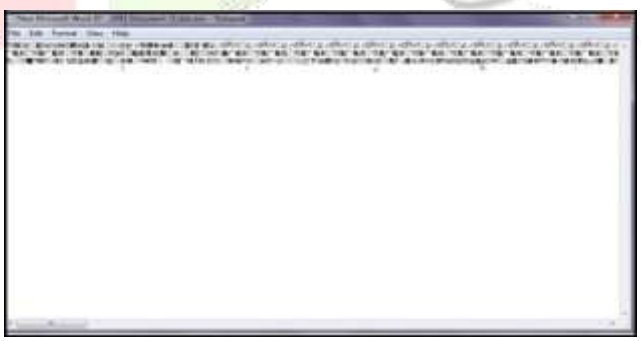

Figure 4.4: Display screen of an encrypted File

5. Conclusion

Our main contribution is in designing and building a Java File Security System (JFSS) that was developed with the express goal of enhancing file data security. The main objective is to provide data security with user convenience. This has been done by implementing JFSS in user-space and enabling encryption and decryption of the files on-the-demand and in a transparent way. We have seen that implementing JFSS in user-space enables the operating system to provide file data security as one of its extensible

functionality. JFSS is cryptographically enforced and uses public-private pair key to control the access of a file. Using public cryptography makes it more reliable, secure and in a way provides user authentication also. JFSS is very convenient to user. The scheme guarantees an end-to-end protection leading to a secure computing environment. JFSS overcomes one of the major drawbacks of the TCFS, which uses one key per user for encrypting all files. We believe that storing data and the key on the same disk and using login password to encrypt the key has several vulnerabilities. JFSS uses randomly generated key for a file and each file is

Encrypted with a different key. JFSS stores the private key on a smart card thus separating it from data it protects. But this forces the scheme to be used on a system which has a card reader. We achieved high security by including support for the Rijndael (AES), designing a strong access control mechanism using public cryptography and session entry for accessing confidential data. We achieved high performance by designing JFSS to run in user-space and implementing it on the Java platform the best choice. We achieved ease of use by providing encryption and authentication that is transparent to users and process.

7. REFERENCES

[1] Brijender Kahanwal, T. P. Singh, and R. K. Tuteja. International Journal of Computer Science & Technology, Vol. 2, Issue 3, pages 25-29, September 2011

[2] M. Blaze, “A Cryptographic File System for UNIX”, in ACM Conference on Computer and Communications Security, pages 9-16, 1993

.

[3] FUSE-File System in User Space, http://fuse.sourceforge.net/.

[4]G. Cattaneo and G. Persiano. “Design and Implementation of a Transparent Cryptographic File System for UNIX”, In Proceedings of the AnnualUSENIX Technical Conference, REENIX Track, pages 245-252, June 2001

.

[5]B. Kahanwal, T. P. Singh, and R. K. Tuteja. “A Windows Based Java File Security System (JFSS)". International Journal of Computer Science & Technology (2011),Vol. 2, No. 3, pp. 25-29.

[6]B. Kahanwal, T. P. Singh, and R. K. Tuteja, “JavaFile Security System (JFSS) Evaluation Using Software Engineering Approaches”, International Journal of Advanced Research in Computer Science & Software Engineering (2012), Vol. 2, No. 1, pp. 132-137.