Survey On Kernel Level Encryption

Prof. T. R. Shinde[1], Prof. M. U. Sanap[2], Prof. N. M. Karolia[3]

[1,2,3] *Professor, Dept. of Information Technology, Pimpri Chinchwad Polytechnic, Pune, Maharashtra, India*

Abstract - *Most operating system controls access to files through a system of user identification and permissions, which provides an adequate level of security for most purposes. However the persistent nature of disk storage creates a vulnerability that operating system permission cannot redress. Thus dealing with file that must remain confidential, information on the disk must be stored in such a way that even if the integrity of the physical media is compromised, the information stored on it remains safe.*

This project deals with modifications to the Linux Operating System Kernel to provide encrypted file system support. Security to the user data will be provided by embedding encryption/decryption algorithm into device driver. So, when user writes data onto the device, it's device driver will encrypt the data and store onto the storage device (part of RAM). Similarly, while user request for the data from the device, it's device driver will retrieve and decrypt that data and this decrypted data can be viewed by user. Hence, this concept guarantees that no file information will ever be stored on that device in clear format. Thus, even if the security of the disk itself is compromised, no data may be obtained from it without knowledge of the encryption key.

Key Words: Encryption, Data Security, Decryption, RC4 Algorithm, Linux Kernel Level.

1. INTRODUCTION

1.1 BACKGROUND

The LINUX is one of the most powerful and widely used OS in the computer world. It is known for its openness, robustness, and security. The LINUX professionals always love working with this OS and they use this OS for performing various task like programming, administration, networking and other application oriented tasks.

Most operating systems control access to files through a system of user identification and permissions, which provides an adequate level of security for most purposes. However the persistent nature of disk storage creates a vulnerability that operating system permissions cannot redress. Thus when dealing with files that must remain confidential, information on the disk must be stored in such a way that even if the integrity of the physical media is compromised, the information stored on it remains safe.

This project deals with the modifications to the Linux operating system kernel to provide encrypted file system support. Security to the user data will be provided by embedding encryption/decryption algorithm into device driver. So, when user writes data onto the device, its device driver will encrypt the data and store onto storage device (part of RAM). Similarly, while user requests for data from the device, it's device driver will retrieve and decrypt that data and this decrypted data can be viewed by the user. Hence, this concept guarantees that no file information will ever be stored on that device in clear format. Thus, even if the security of the disk itself is compromised, no data may be obtained from it without knowledge of the encryption key.

1.2 RELEVANCE

Many encryption programs exist for Unix-based machines that can provide above mentioned type of security. Most run as command line utilities and work on a file-by file basis, i.e., the user creates a file using one program and then runs the encryption program separately to protect it. However, these utilities are typically awkward to use as they require the time-consuming and tedious ritual of running a separate program twice for every access to a confidential file. Further, command-line encryption utilities suffer from three potential security compromises that stem from human neglect or error. First and most obvious, if a user ever forgets to run the encryption program after accessing a file the data will reside on the disk in a vulnerable form. Second, the user must remember that deleting the unencrypted copy of the file usually won't remove the data from the disk. In most operating systems, deleting a file means its disk space has been marked available for future use. The user needs to remember to take whatever extra steps necessary to actually remove the unencrypted data from the disk. Finally, many applications create temporary files that store unencrypted pieces of your confidential data elsewhere on the disk. The user must make sure these temporary files are wiped off the disk to prevent a security compromise.

Encryption at kernel level substantially improves upon the security of command-line utilities by encrypting an entire file system. This approach has several advantages. Encryption at kernel level transparently encrypts and decrypts all data stored in the file system which ensures no information is ever in clear form .This eliminates the problems of forgetting to encrypt a file when done, dealing with the residual images of deleted files on the disk, and keeping track of temporary files.

1.3 ORGANISAION OF PROJECT REPORT

The rest of this report is structured as follows. We describe literature survey in chapter 2. In chapter 3 we describe the

existing of different file system. In chapter 4 we describe proposed approach. In chapter 5 we specify software requirement specification. In chapter 6 we specified project planning and management in which project mile stones are specified. In chapter 7 we discuss system analysis and design in which the algorithms of the implementation are discussed. In chapter 8 the operating system calls and system commands used in project implementation are described. In chapter 9 results achieved are specified. The chapter also describes the testing plan and the test cases of the project development. Chapter 10 describes the concluding remarks of the project along with future developments are specified. Finally, we conclude with references at the end of the project report.

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.

3. PROPOSED WORK

There have been different approaches used, to solve the file data security problem. Above 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 kernel level encryption 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. There has been a lot of development taken place since the time when MS DOS device driver was used to encrypt the entire partition. Nowadays we have several cryptographic file systems available, which we have briefly described.

4. ARCHITECTURAL VIEW

The architecture of the proposed system for LINUX can be show in the figure below.

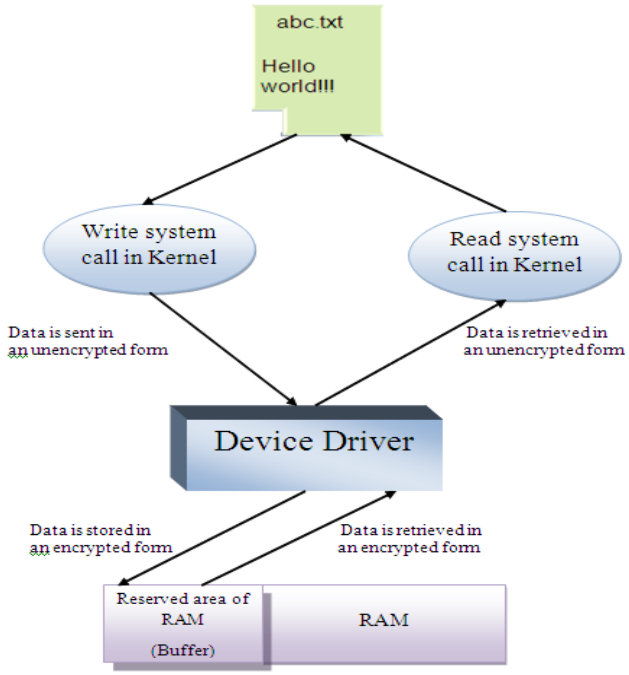

Fig -1: Proposed Architecture for LINUX

System working flow is given below:

1. A device driver for the character device including in the functionality of cryptography needs to be registered in to the kernel.

2. When the device driver is registered in to the kernel, it is allocated a Major and a Minor number with the help of which device file needs to be completed.

3. As the users don't have direct access to the device driver, a user program is needed so that user can interact with the device driver.

4.When some data is to be written on to the device, it send to the device driver by passing the data as parameter to the system call being invoked from the user program.

5.Then the device driver encrypts the same and writes the Cipher text on to the device.

6.When the data is to be retrieved from the device, then a system call is invoked using the user program.

7. Then the system call makes a request to the device driver to read the data from the device which in turn retrieves the data from the device, decrypts it and gives the plain text to the system call so that it can be presented to the user.

In this way the user is not aware of how the cryptographic transformations take place on the data while writing and reading the same to and from the device.

5. ALGORITHM RC4

RC4 is a stream cipher having variable key-size stream with byte oriented operations.

The algorithm is based on the use of random permutations. The RC4 algorithm is remarkably simple.

A variable length key of from 1 to 256(8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0],S[1],S[2],.....,S[255].

At all times, S contains a permutation of all 8-bit numbers from 0 to 255. For Encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion. As each value of k is generated, the entries in S are once again permuted.

6. CONCLUSIONS

The first level implementation results show that data encryption is a powerful tool through which data security can be obtained. The results achieved shows that the data is encrypted before storing data onto the device and decrypted after retrieving and before displaying data to the user with the help of key provided by the user. Hence, confidentiality of data is maintained and the disadvantages of conventional operating system are overcome.

The project has dealt with designing of character device driver that encrypts and decrypts data. The project can further be extended to handle block devices. Also, the resulting overhead caused because of encryption and decryption is not considered in the project, it can be considered for future enhancement.

REFERENCES

[1] Understanding the Linux kernel, Daniel P. Bovet, Marco Cesati, October 2000.

[2] Kernel Level Implementation of an Encrypting File System, Brian. K.Dewey, May 1996.

[3] Operating System Principles, Peter Baer Galvin, Abharam Silberschatz, Greg Gagne, March 2008.

[4] Linux Kernel Development, Robert Love, January 2005.

[5] Beginning Linux Programming, Neil Matthew, Richard Stones, March 2004.

BIOGRAPHIES

Prof. T. R. Shinde B.E.(Computer Engg.) From PCCOE,Pune in 2011. She is currently working as Asst. Prof. in Information Technology Dept, PCP, Pune. She has published research paper in International Journals and conferences. Her Interest areas include Data Mining, Microprocessor and Computer Security.

Prof. M. U. Sanap M.E.(Computer Engg.) From SKNCOE, Pune in 2016. He is currently working as Asst. Prof. in Information Technology Dept, PCP, Pune. He has published various research papers in International Journals and conferences. His Interest areas include Data Mining, Big Data Analytics and Computer Security.

Prof. N.M. Karolia MBA (IT) From SBPIM, Pune in 2017. She is currently working as Asst. Prof. in Information Technology Dept, PCP, Pune. She has done research in Agile Software Development. Her Interest areas include OOP and Computer Security.