International Journal of Security and Its Applications
Vol.8, No.1 (2014), pp.123-132
http://dx.doi.org/10.14257/ijsia.2014.8.1.12

A Reliable File Protection System Based on Transparent Encryption

Jun Liu [1], ShuYu Chen [2*], MingWei Lin [1] and Han Liu [1]

[1]*College of Computer Science, Chongqing University, Chongqing, China*
[2]*College of Software Engineering, Chongqing University, Chongqing, China*
oldcattleliu@cqu.edu.cn, netmobilab@cqu.edu.cn, linmwcs @163.com, visualliu@126.com

Abstract

As data leakage moves to ever more challenging areas, improving the security level of anti-data leakage and lowering overhead to the operating system becomes increasingly important. Therefore, the paper presents a novel double cache file filter driver based on transparent encryption, called as DBFD. In order to boost security of data, that is controlled by the file filter driver. The DBFD overcomes the limitation of double cache in the file system kernel in windows operating system and used the transparent encryption method to protect the data security. To evaluate DBFD, we used Iometer as the measurement tools to measure performance. The simulation results indicate that proposed DBFD has higher security, less overhead to the windows operating system.

Keywords: *File Filter Driver, Transparent Encryption, Double Cache, Data Leakage*

1. Introduction

At present, we have many urgent things to deal with in data security, one of which is the data leakage. Data leakage is the one of the main problems of data security for most enterprises. The problem of data security does not only come from hacker Intrusion, Trojan horse programs and other security threats from the internet. In fact more security threats come from enterprise itself because of lack of security experience. A lot of important documents are stored in a sensitive position directly, such as desktop and my documents folder in windows operating system.

There are many technologies about the solutions of the information security. Intrusion detection, firewalls and private networks and are traditional methods in information security. Generally speaking, intrusion detection systems, which are used in networks, are to monitor some malicious or abnormal behavior [1, 2]. Firewalls are controlled link systems aimed to protect threats and attacks from the internet and network-based [3]. Private networks, commonly used for home, office, and enterprise, are local area networks that private IP address space [4]. But these methods are difficult to prevent data leakage because they are suitable for dealing with network and malicious code attack. Sand-box security model is another important security mechanism, which can execute untrusted or unknown applications. Sand-box can prevent these attacks from being successful by watching the code for adherence to a specific policy [5]. But there is an obvious defect in sand-box mechanism, which has a large overhead for security checks.

This paper introduces a reliable protection system, which can control the access of files and documents by specific applications. The protection system controls data leakage by transparent encryption technologies. The processes of encryption and decryption are executed in file systems using file filter driver and are completely transparent to users. In order to

ISSN: 1738-9976 IJSIA

improve efficiency of data checks, the system creates double caches in file system and breaks the single buffer limit of Windows operating system kernel buffer manager. The system must ensure the reliability intercepted the read and write operations of files and documents through the analysis and research of the reading and writing mode accessing by applications.

The paper is organized as follows: Section 2 presents the related work for data leakage and the reliable protection system is discussed in Section 3; Section 4 shows the experimental results. Finally, in section 5 final remarks and a conclusion are presented.

2. Related Work

The innovative idea of control file access has stimulated much research in this area. An excellent review of control data leakage can be found in Ref. [6]. In the following, we will briefly summarize anti-data leakage works, encompassing very recent research.

Transparent encryption technologies based on file system filter drivers are widely used in preventing data leakage. The encrypting file system (EFS) using transparent encryption technology first showed up in the Windows operating system. Although the technology has the advantages of high safety, simple, easy to use and closely relation to the operating system, there are still some problems in the practical application, mainly reflected in the following two aspects. Firstly, EFS only supports NTFS file system, does not support FAT file system. Secondly, EFS can only encrypt files and folders, and does not support in file type or the specified program file encryption [12]. Aiguo HE and Tomohiro Ohdaira proposed FP (File Protector) [6] using file hook driver to monitor only the file access in Microsoft Windows environment. At the same time FP draws lessons from the sand-box security model based function. The main purpose is to improve the execution time of application programs running under the sand-box security mode [6]. But the main disadvantage of FP is that the overhead to the operating system is higher.

Sand-box security model is extremely useful for secure execution of untrusted applications, Many security systems based on sand-box model so far provide security by intercepting system calls invoked by applications and controlling their execution [7] [8]. There are many platforms for control file access under sand-box. Safe-Tcl [9] introduces padded cells. The padded cells are separated environments that can identify the trust and distrust application programs. Safe-Tcl is written in the Tcl scripting language [9]. Tcl scripting language has Tcl Interpreter. The important thing is that the process of interpreter will take some time [10]. MAPbox [11] is a classification mechanism based on sand-boxes by retaining the ease of use of application-class-specific. However, these mechanisms are difficult for computer users to manger [6], because they are so complex and increase the memory consumption to operating system.

Access control list (ACL) is another important guard against data leakage mechanism. ACL can control the security behavior associated with a given (protected) object of some sort. [13]. ACL based on managing access control policy was proposed in Ref [14, 15]. However, it is very difficult for computer novices to manage the control policy [6], because the control policy is not easy to computer novices. Another disadvantage of ACL is that the consumption to operating system of ACL is more than the mechanism based on file system filter drivers.

3. The Design and Implementation of DBFD

An efficient preventing data leakage model must focus on reducing the check over access cost and controlling the way to read important or sensitive data. To achieve these goals, we propose a novel double cache file filter driver based on transparent encryption policy, which is called DBFD, for data leakage.

The key of the DBFD is the introduction of double cache file system filter driver based on transparent encryption. In this model, all the important and sensitive data must be encrypted. There is a file filter driver in DBFD. The file filter driver is manger to buffer for every file. In general a driver only allocates a buffer in windows operating system kernel. In order to improve the efficiency of access file, the DBFD file filter driver overcomes a buffer in windows operating system and creates two buffers for a file in windows kernel. Authorization application accesses the encryption files by specific buffer which contains the plain text of encryption files. The applications that do not have the proper authorization access encryption files by another buffer which contains the ciphertext of encryption files. That is the encryption files will not be decrypted when unauthorized applications access encryption files. The two buffers and the process of encryption and decryption are managed by file filter driver.

3.1. Creating Double Cache

The buffer manager in windows operating system must maintain the information of each file buffer. This information is maintained by cache bitmap. The buffer manager allocates a structure of shared cache bitmap to save the file and associated with the file information. The structure of shared cache bitmap is allocated in the file buffer by a file system driver or network redirection request initialization. It is important to share cache bitmap which is unique for each file, because buffer manager allocate the private cache map when the file buffer for the first time established. The buffer manager uses the structure to manage file information, such as prefetching control and access information. It is important that these information and structure are maintained by buffer manager in windows operating system.

In order to create the two buffers for a file in windows operation system kernel, DBFD proposed the method of copy interface. The method of copy interface is that file filter driver copy the contents of file buffers to user buffers or copy the contents of user buffers to file buffers. At the same time, file filter driver need map a specific virtual address range to physically address composed of one or more pages. Fig. 1 illustrates the process of using double cache to read the file.

As shown in Figure 1, there is a cache control block (CCB) in order to query or scheduling cache in DBFD. The CCB records file name, file object and cache pointer .etc information corresponding to the buffer. So that DBFD can convenient query, modify and scheduling buffer.

3.2. Authorization and Unauthorized Process

Only authorization processes can access encryption files in DBFD. Unauthorized processes can not access encryption files. The authorization processes are that applications, which can access important or sensitive data for user or company. These files created by these applications are encrypted. The process of encryption and decryption is transparent to authorization processes, because filter driver deal with the operation of encryption and automatic.

The identification of authorization is through processes name. There is a structure, which is called EPprocess, in operating system kernel. The structure includes many processes information. Processes name is one of the important parameter in the structure. DBFD introduced MD5 algorithm to prevent fake authorization process by rename processes, because every file has a unique value no matter what type of file. Figure 2 describes the whole process of judgment in detail.

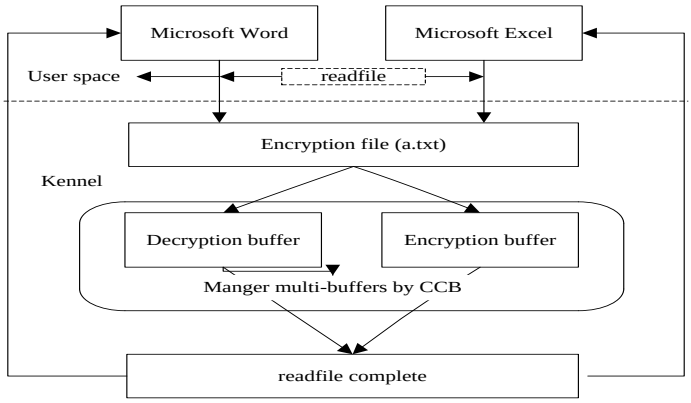


Figure 1. Cache Control Block Maintain two Caches

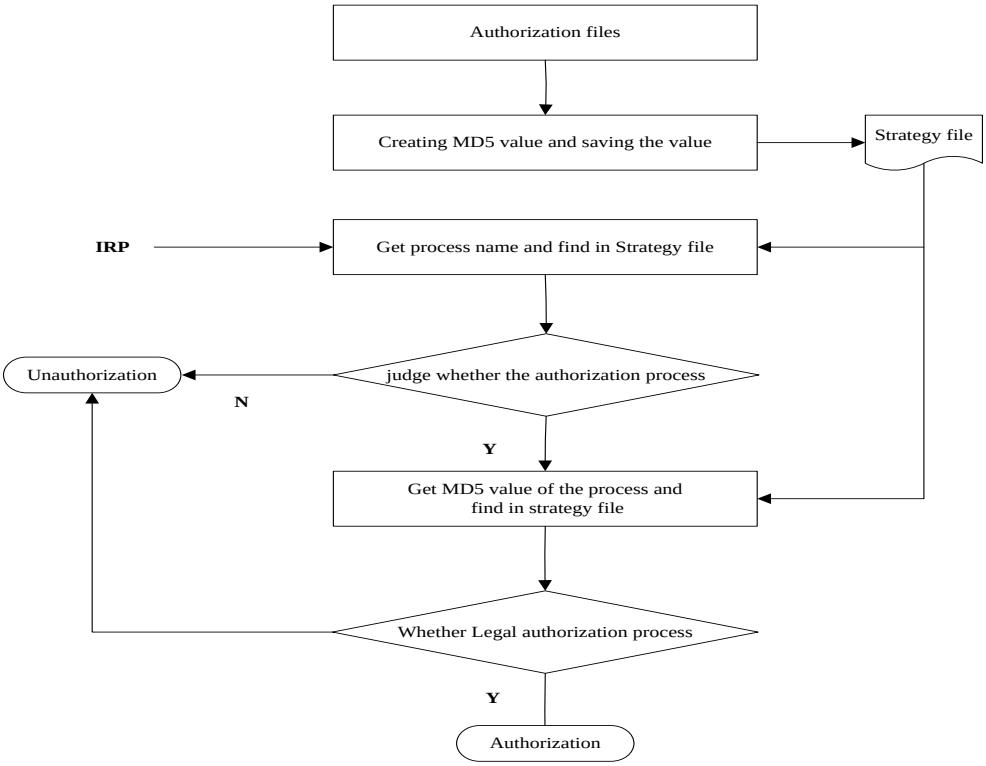


Figure 2. The Flow Chart of Allowed-process Judgement

3.3. Encrypted File Identifier

There are mainly two methods to store the encryption flags in file filter driver. For one thing, the encryption flags are stored in the end of the file. For another, the encryption flags are placed at the beginning of the file. Figure 3 shows three status of the encryption files. Figure 3 (a) presents the initial status. Figure 3 (b) presents the status that the encryption flags are located in the end of the file. Assume that the original size of the file is P. The encryption flags size is P. The encryption flags size is defined as (L). Then the encrypted file length is ($P+L$). Although the file size in Figure 3(b) and Figure 3(c) are equal, there is a distinct difference between Figure 3(b) and Figure 3(c). In case of Figure 3(b), the encryption file size

is ($P+L$), when user open the file properties to view file size in windows operating system. In case of Figure 3(c), the encryption file size is (P) under the same conditions. Obviously, saving the encryption flags at the beginning of the file is a good method, which is used in DBFD.

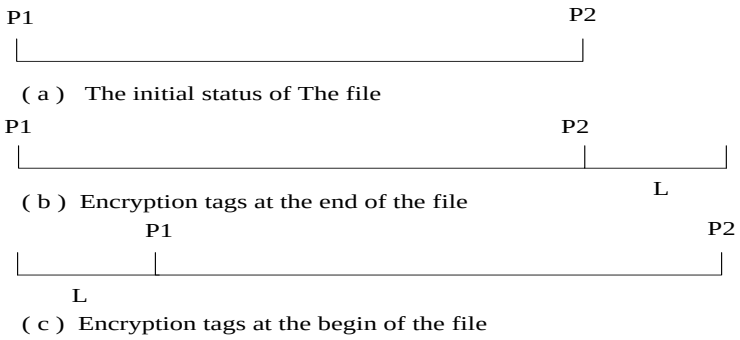


Figure 3. The Storage of Encryption Tags

The read and write operations will recalculate because of saving the encryption flags at the beginning of the file. As shown in Figure 4, assume that data will be written from position *Pw1* to *Pw2*, the actual writing position is ($L+Pw1$). The size of writing is ($Pw2-Pw1$).

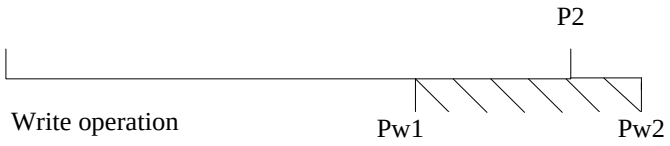


Figure 4. Write Operation

Figure 5 illustrates the encryption identification process of generation and judgment. Generally speaking, IRP_MJ_WRITE and IRP_MJ_SET_INFORMATION command are dispatched by I/O manager, and the encryption flags are added when file filter driver received the IRP_MJ_WRITE command. The file size is handled in IRP_MJ_SET_INFORMATION command.

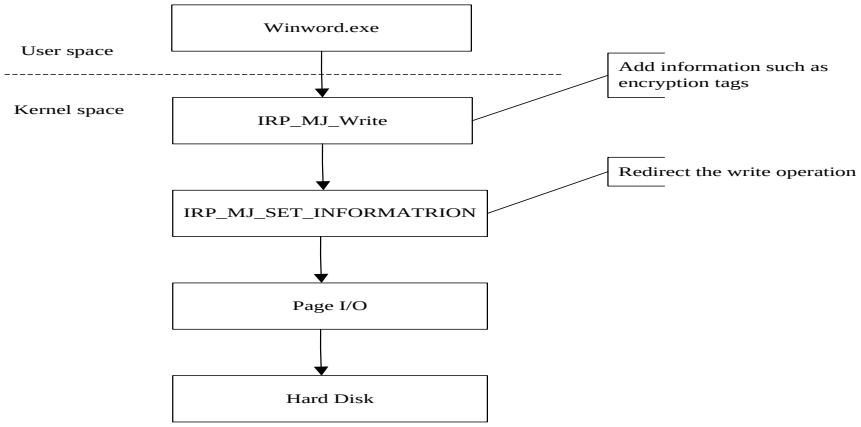


Figure 5. Process of Adding Encryption Flag and Changing File Length

3.4. Encryption Algorithm

In DBFD, the file contents are protected by security features based on AES256 symmetric encryption algorithm, because asymmetric algorithm has slower speed in encryption than symmetric algorithm. Each file encryption keys generated randomly is stored at the beginning of the file. The beginning of the file is named encryption flags area. The encryption flags area size is defined as (L) in DBFD. The value of (L) is 4KB that 4KB is aligned unit when the buffer writes data to hard disk. Obviously, the 4KB data is very important. So, the 4KB data is encrypted using RSA asymmetric encryption algorithm, because asymmetric algorithm has higher security in encryption. Encryption keys including public key and private key are generated using random function during system initialization. The private key is stored in USB devices, and the public key is stored in the system configuration file. The private key stored in USB devices should be encrypted again using symmetric encryption algorithm in which key is created by MD5 algorithm, because the private key is very important. Figure 6 describes the detail relationship with the encryption keys.

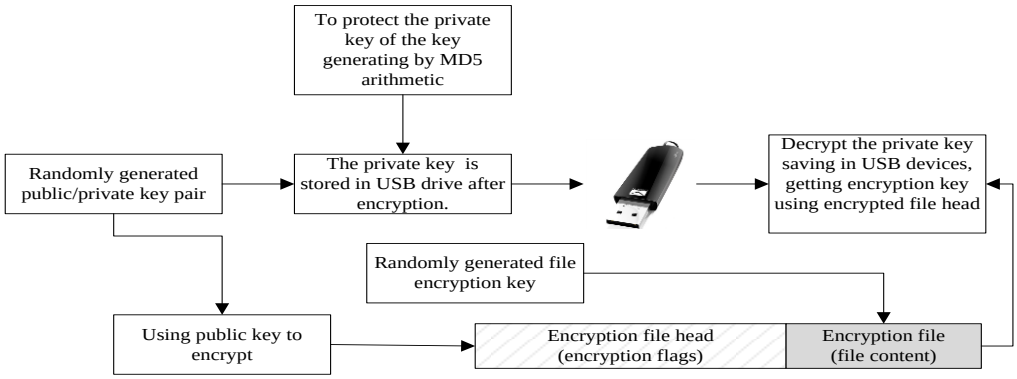
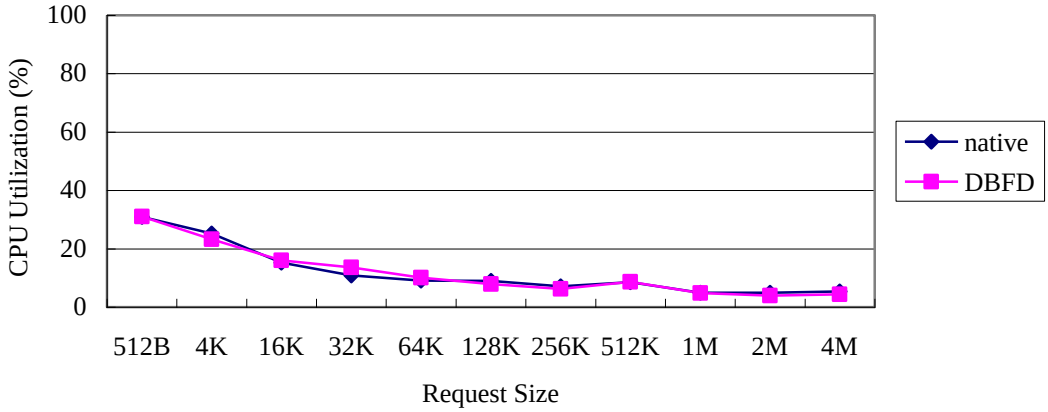


Figure 6. Relationship of the Encryption Keys

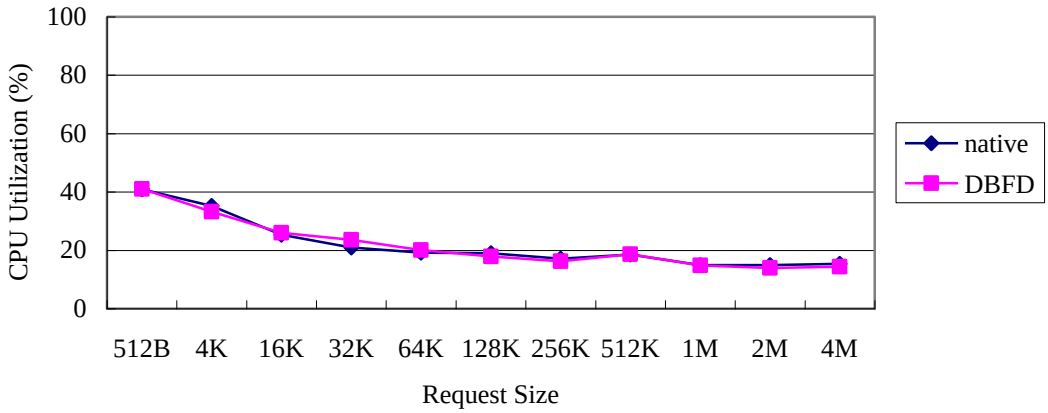
4. Experimental Evaluations

We have been implemented the prototype of DBFD in order to investigate the performance of our proposed DBFD system, and the prototype is tested in a test PC, which is configured as: AMD Athlon(tm) Dual Core processor 2.20 GHz, 2000 MB RAM, Windows XP with service pack 3 and a 500 GB Seagate. The overall performance of DBFD has been compared with that in a native environment which has no DBFD prototype, and the results of experiment show that the DBFD has less overhead to operating system.

In order to evaluate the performance consumption of windows operating system, we used Iometer as the measurement tools, which is an I/O subsystem measurement and measures performance of storage system such as Disk. Testing may be conducted by changing three parameters: percentage of read (%read), percentage of randomness (%rand) and percentage of write (%write). In the experiments, we take two kinds of combination way to test, which are (100%read, 0%write, 100%rand) and (0%read, 100%write, 100%Sequential), respectively. The file size of the read and write operations increase from 512byte to 4Mbs. As show in Fig. 7, DBFD has little overhead to windows operating system, because test results in DBFD are similar with local environment.



(a) (100%read, 0%write, 100%rand)



(b) (100%read, 0%write, 100%rand)

Figure 7. CPU Utilization between Native Environment and DBFD

DBFD should affect the response time of file read and write to a certain degree, that file system encrypt the data automatic. There fore copy multiple files and large files are tested in order to access the response time in copy file operations. As shown in Figure 8, the response time in DBFD is close to native environment when the files size increases from 50MB to 4GB.

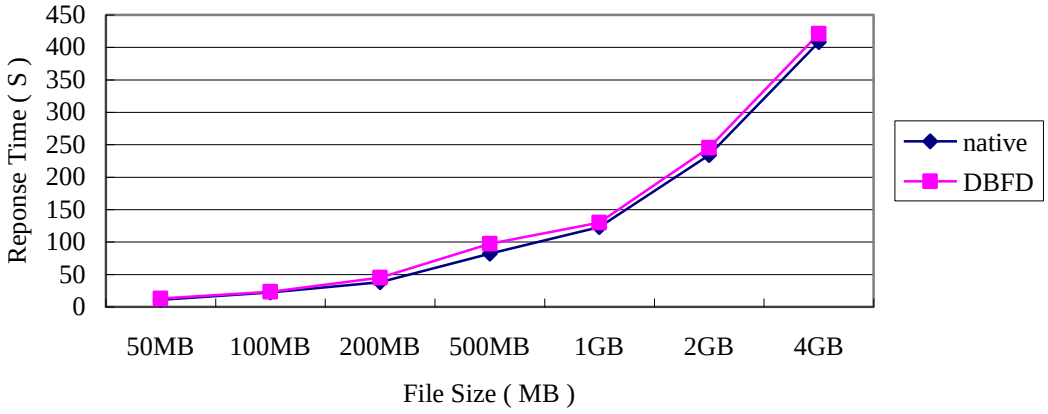


Figure 8. Response Time for Different File Size in Copy File Operations

5. Conclusion

In this paper, we propose a double cache file filter driver layer based on transparent encryption, which is called DBFD, to prevent data leakage. DBFD introduces double cache in file filter driver in order to improve the performance of read and write operations. There is better security in DBFD, because the file content uses AES256 algorithm and encryption keys use RSA algorithm. We conducted a series of experiments and obtained encouraging results.

Acknowledgements

We are grateful to the editors and anonymous reviewers for their valuable comments on this paper.

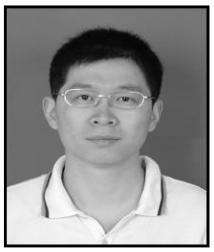
The work of this paper is supported by National Natural Science Foundation of China (Grant No. 61272399) and Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20110191110038).

References

[1] G. Pannell and H. Ashman, "Anomaly Detection over User Profiles for Intrusion Detection", Proceedings of the 8th Australian Information Security Management Conference, Perth, Australia, **(2010)** November 30-December 2.
[2] X. Huang, X. Wang and S. Zhu, "Research on firewall system for confidential network", Advanced Materials Research, vol. 433-440, **(2012)**, pp. 4279-4283.
[3] R. Oppliger, "Internet security: Firewalls and beyond", Communications of the ACM, vol. 40, no. 5, **(1997)** May, pp. 92-102.
[4] http://en.wikipedia.org/wiki/Private_networks.
[5] T. Khatiwala, R. Swaminathan and V. N. Venkatakrishnan, "Data Sandboxing: A Technique for Enforcing Confidentiality Policies", Proceedings of 22nd Annual Computer Security Applications Conference, Miami Beach, Florida, **(2006)** December 11-15.
[6] H. E. Aiguo and T. Ohdaira, "A Case Study: File Access Privacy Control Using Filter Hook Driver", Proceedings of IEEE International Symposium on Parallel and Distributed Processing with Applications, Chendu, China, **(2009)** August 9-12.
[7] V. Prevelakis and D. Spinellis, "Sandboxing Applications," Proceedings of the USENIX Technical Annual Conference, Boston, USA, **(2001)** June 25-30.
[8] D. W. Goldberg, R. Thomas and E. Brewer, "A secure environment for untrusted helper applications confining the Wily Hacker", Proceedings of the 6th conference on USENIX Security Symposium, California, USA, **(1996)** July 22-25.

[9] J. K. Ousterhout, J. Y. Levy and B. B. Welch, "The Safe-Tcl Security Model", Proceedings of the USENIX Annual Technical Conference, Orleans, USA, **(1998)** June 15-19.
[10] B. Welch, "Practical Programming in Tcl and Tk", Prentice-Hall, ISBN 0-13-182007-9, **(1995)**.
[11] A. Acharya and M. Raje, "MAPBox: Using Parameterized Behavior Classes to Confine Untrusted Applications", Proceedings of the 9th USENIX Security Symposium, Denver, USA, **(2000)** August 14-17.
[12] http://technet.microsoft.com/en-us/library/cc700811.aspx.
[13] http://technet.microsoft.com/zh-cn/library/ff538821.
[14] K. Kida, H. Sakamoto, H. Shimazu and H. Tarumi, "A Proposal of File Access Control Software Agent Toward Using P2P File Sharing System in Safety", IPSJ Journal, vol. 48, no. 1, **(2007)**, pp. 200-212.
[15] S. Kai, M. Arai, Y. Nagai, S. Tomida and S. Tezuka, "Proposal of File Access Control Scheme for Client Protecting Information Assets from Illegal Programs", IPSJ Journal, vol. 46, no. 8, **(2005)**, pp. 1912-1922.

Authors



Jun Liu, received his B.S. degree in Southwest University, P. R. China, at 2001, and M.S. degree in Chongqing University, P. R. China, at 2009. Currently he is a Ph.D. candidate in College of Computer Science, at Chongqing University. His current interests include flash memory, information security, Linux Kernel and big data analytics.

ShuYu Chen, he received his Ph.D. degree in Chongqing University, P. R. China, at 2001. Currently, he is a professor of College of Software Engineering at Chongqing University. His research interests include embedded Linux system, distributed systems, cloud computing, etc. He has published over 120 journal and conference papers in related research areas during recent years.

MingWei Lin, received his B.S. degree in Chongqing University, P. R. China, at 2009. He is currently a Ph.D. candidate in Chongqing University. He is invited as the reviewer by *Journal of Systems and Software*, as well as *Computers and Electrical Engineering*. His current interests include flash memory, Linux Kernel, information security and wireless sensor network.

Han Liu, received his B.S. degree in Chongqing University, P. R. China, at 2002, and M.S. degree in Chongqing University, P. R. China, at 2010. His current interests include date backup and recovery, information security, Linux Kernel.

