2009 Fourth International Conference on Computer Sciences and Convergence Information Technology

A Framework for a File View Model in Intranets

Yong Ai
Computer School
Wuhan University
Wuhan, China
aiywhu@gmail.com

Hongbin Dong
State Key Lab of Software Engineering
Wuhan University
Wuhan, China
hbdong@whu.edu.cn

Yiwen Liang
Computer School
Wuhan University
Wuhan, China
ywliang@whu.edu.cn

R.I. McKay
Structural Complexity Laboratory
Seoul National University
South Korea
rim@cse.snu.ac.kr

***Abstract*—Today, a new security problem is arising in intranets. The threats from inside an organization account for a rapidly increasing proportion of losses. The DAC (Discretionary Access Control) model, which is the primary access control mechanism in most intranets, is main responsibility for this state of affairs. Users can make a duplicate of a confidential document for which they only have read authorization. They can then grant access rights the replication to others who did not previously have authorization. This transformation of authorizations would result in the contents being divulged to unauthorized users. This paper proposes a concept of "File View" to solve this security problem in intranets. First, the paper proposes a hierarchy of file views which are used to structure availability of reference to database views. However there are some challenges in extending this proposal to file systems because of differences between the two. The paper proposes a framework for a file view model to solve these challenges. Finally, under three assumptions, it discusses access control and proposes read and write process algorithms for secured access in this framework.**

***Keywords*-File View; Mapping Mechanism; Information Security; Access Control; Access Tracing**

I. Introduction

Today, the greatest proportion of organizational information resides in digital documents. The security policies in most enterprises have largely evolved to prevent attack from the extranet. But according to the 2007 CSI/FBI Computer Crime and Security Survey, almost 64% of respondents believe that insider threats account for a non-negligible proportion of their overall cyber losses [1]. Attacks coming from inside an organization are becoming an important new issue. Therefore it is important to strengthen the security of intranets of enterprises and organizations. The confidential information needs to be protected to the greatest extent possible.

At present, file security is increasingly important in the intranet. Most intranets adopt some form of access control. [2]. The access control model which is mostly used in intranets results in poor security. The operating systems used in most intranets belong to class (C2) [3] of the TCSEC (Trusted Computer System Evaluation Criteria) – Unix/Linux and variants of Microsoft Windows. In this TCSEC class, the access control model is DAC (Discretionary Access Control) [4, 5]. The central idea of DAC is that the owner of an object – who is usually its creator – has discretionary authority over who else can access that object [5, 6]. There is a security problem in DAC model, in that it is unable to enforce information flow controls [7]. For example, Alice has created an object and granted read authorization to Bob. In this circumstance, Bob can create a replica of Alice's object, which is legal in DAC. Since Bob created this replica, he can grant authorizations on this replica to others who did not previously have those authorizations. This may result in the contents being divulged to unauthorized users. The change of authorizations through information flow is a serious security problem in intranets.

There are two methods to solve this problem. The first is to use an EDMS (Enterprise Document Management System) [8] to manage confidential documents. But in an EDMS, documents are accessed and manipulated through a database management system. This mode is inconvenient for exchanging documents in an intranet. The other is to change the operating system enterprises to a more secure one in TCSEC class B1 [3] , such as SELinux (Security-Enhanced Linux) [9]. SELinux is an implementation of flexible and fine-grained nondiscretionary access controls in the Linux kernel [10]. In TCSEC class B1, the access control model is DAC plus MAC (Mandatory Access Control) [3]. The Mandatory Access Control policy is expressed in terms of security labels attached to subjects (usually the users) and objects [11]. There are two principles – read down and write up – which should be maintained by comparing a subject's clearance and the security level of the object being accessed [5]. But when the subject's clearance and the security level of the object are at the same level, no security principle is defined. Generally, DAC is used as access control model at the same level, leading to a recurrence of the security problem mentioned above.

Thus these two methods do not accord well with the environment in most enterprise intranets. To satisfy the requirements of an intranet, a solution is needed which

1) uses DAC as access control model
2) uses file systems rather than databases

The idea of a database view is nevertheless a useful perspective Database views are used for two purposes: data

978-0-7695-3896-9/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCIT.2009.63

protection and user convenience [12], corresponding to our purposes. Thus we propose, for a similar concept in file systems, the term "File View". The "File View" approach is intended to improve the DAC model when either used alone, or used in conjunction with the MAC model, to provide security control within a single security level. This new mechanism for intranet file systems can realize three new functions:

1) **Viewing Flexibility**
   It results in greater flexibility for users viewing documents. Different users can access different parts of the same document as appropriate, rather than gaining access at the whole document level. Different users might have access to different content from the same document.
2) **Greater Security**
   It provides greater security when users access the documents. The access to information being exchanged between computers can be controlled. Even if users obtain a duplicate of a file view for which they do not have authorizations, the authorizations of the source file view will be validated while they access the duplicate, so they will not be able to perform any operation they could not perform on the source. This can prevent information from leaking out unintentionally while being transferred in an intranet.
3) **Access History**
   The access history – who has tried to access confidential and sensitive information – can be recorded. These records can help a manager to locate hidden security problems in an intranet, and reduce security risks as far as possible.

The hierarchy of file views uses the structure of database views for reference. In the rest of this paper, a framework to solve these challenges is proposed. Under three assumptions, we define structures for security attributes and access histories, for verifying authorization and tracing the access history. Finally, we propose two algorithms for access control in file views.

II. Related Work

The concept of an enterprise document management system was proposed in 1991 [8]. The stored objects in an EDMS are organized, accessed and manipulated through a database management system. An EDMS is designed to securely manage the production and delivery of high-value documents[13]. But all documents must be exchanged through the database. This can be inconvenient to users in an intranet.

The traditional access control models most commonly used are DAC[5, 6], MAC[11] and RBAC(Role-Based Access Control Model)[14]. Today, most operating system use DAC as the access control model. The DAC model, is unable to enforce information flow controls[7]. This security problem can not be easily avoided in intranets.

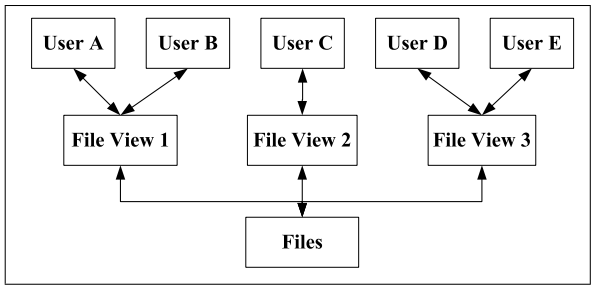

Figure 1. The hiberarchy of file views [24]

There are some studies using an XML(*eXtensible Markup Language*) [15] view which is semi-structured. The work on querying and updating on XML views [16–21] forms a useful background for file views. But in a file system, there is no access language corresponding to XPath (XML Path Language) [22] and XUpdate (XML Update Language) [23]. The files in a file system have no common structure. It would be infeasible to transform all unstructured files to XML file before transferring them. So a similar mechanism for unstructured files to that for semi-structured files is needed.

III. File View

A. Preliminaries

In database theory, a view is a stored query, composed of the result set of a query script. Database views can be used for two purposes: data protection and user convenience [12]. Considering the characteristics of database views, and the security requirements for operating system in an intranet [24],we propose the term "File View" as seen in Fig. 1, using the mechanism of database views for reference.

B. Main Challenges

The differences between database views and file views arise from the storage format. A database view is structured, its storage format is strictly controlled by rules. But the file view based on an operating system is unstructured, its storage format is only loosely controlled. Because of specific issues in operating system, there are some challenges we need to consider in using the structure of database views for reference.

1) When users query data from a database view, it is not executed in the database client but actually executed in the database server. There is no similar server in an intranet operating system for exchanging documents.
2) The database view is a dynamic virtual table, computed or collated from data in a database. All the data in views is actually stored in the database server. But there is no similar server in operating systems. The whole content has to be attached to the file view while it is being exchanged through the intranet. Yet the

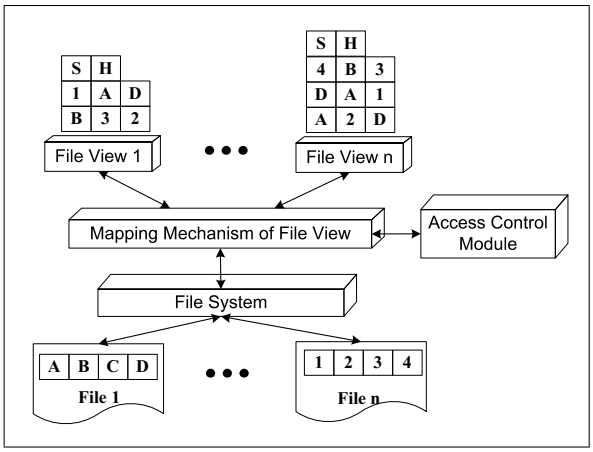

Figure 2. The File View Framework

content of the file view should not be accessible in an illegal manner.

3) The security attributes of tables and views in database view are stored in the data dictionary. There is no similar storage mechanism in operating systems.
4) All the processes executed on tables and views in a database can be logged, while the process executed on files in operating systems are not.

C. The File View Framework

To resolve the challenges mentioned above, the security attributes and process history should be stored in the file view as part of the data. The framework for file views of [24] can be improved as shown in Fig. 2.

As seen in figure 2, the logical files in a file system are divided into logical blocks. This structure makes the access modes for logical files in a file system more flexible. It is possible to grant authorizations on parts of a file rather than as a whole.

Several logical blocks can be recombined as one file through the file view mechanism. For example, the content of "File View 1" in figure 2 comes from several logical files through "File 1", "File n" and so on. In addition to the content, the file view contains two more logical blocks, marked "S" and "H". The logical block marked "S" holds security attributes. It stores the information about access authorizations to logical blocks, which are used for verifying authorizations in the "Access Control Module" of the framework. The other, marked "H", holds the access history. It records processes affecting logical files in the file system for verification, whenever they are stored in computers or transferred though the intranet.

In this file view framework, the challenges mentioned above can be resolved as follows:

1) There is a distributed module (marked "Access Control Module" in figure 2) in every file system in the intranet. It is used to verify the access authorizations for logical blocks in file views when users try to access them.
2) As shown in the framework, the logical blocks from basic files are combined in one file view. The content of these logical blocks are actually stored in the file views when they are exchanged in the intranet.
3) The logical block marked "S" in the framework stores security attributes. It incorporates the information about access authorizations for logical blocks. This information is used for authority inspection, to decide whether blocks in the file view are available to users.
4) The logical block marked "H" in the framework stores process histories. It is used to record the information about processes affecting file views, to provide access tracing.

D. Key Issues

There are some important similarities, but also differences, between database views and file views.

1) **Similarities**
   As in a database view, the mapping mechanism of the file view is multi-layered, which means that the content of a file view can be queried from other file views. As a result, the file view is both flexible and secure.
2) **Differences**
   The database environment is closed. The data in a view is queried from relations through the data dictionary. But the file system is an open environment, logical files can be exchanged flexibly through computers and networks. There is no similar mechanism to the data dictionary in a file system, so that authorizations have to be attached to the file view when exchanging information in an intranet.

Considering the differences between database and file system, there are five issues that need to be solved in basing the structure of reference in a file system on that of databases:

1) What information should be included in the security attributes of the file view?
2) How should we realize the access control based on the multi-layered mapping mechanism?
3) File views are based on unstructured documents. The logical blocks could be defined at many different levels, from one word, to a sentence or a paragraph. How to define the size of the logical block in the file view is a key question.
4) The security attribute of the file view is used to protect the content in the file view. It is crucial that it cannot be modified or destroyed maliciously. How should this information in a file view be protected?
5) The operating systems in an intranet may be heterogeneous. How can we deal uniformly with file views

in different operating system?

E. Assumptions

In this paper, we make some basic assumptions for convenience; these assumptions can be relaxed at the cost of greater complexity

1) Each logical block is a single document (file) in the file system.
2) The security attribute in a file view is securely stored, and cannot be modified or destroyed maliciously.
3) All operating systems in the intranet are isomorphic.

IV. Access Control in the Framework

The access control for file views has two parts: access control on the document containing the file view, and access control on the content of the file view. On the one hand, access control for the document focuses on whether users have the authorizations to access the file view at all. On the other hand, access control for the content focuses on whether the users have the authorizations to access the content of the original files to which these blocks referred.

In the operating system mostly used in enterprises, the authorizations of logical files are usually described as read(R), write(W) and execute(X). The authorizations of the documents of file views are the same as those of basic files. When users access a file view, the authorizations should be validated just as when they access a file in the file system. Users can change the documents of file views through the operating system file system. Thus this paper focusses on access control for the logical blocks of a file view.

For verifying the authorizations and recording the access history, the security attribute and structure of the access history need to be defined first.

A. Security Attribute

The security attribute of the file view is used to store the information about access authorizations for logical blocks. It is used to verify the authorizations when a user tries to process the file view. It should include two parts: the security attribute for the file view and the security attribute for the logical blocks in it.

1) The security attribute for the file view includes the owner's identification, and the access authorizations for the document (generally using an access control list).
2) The security attribute for the logical blocks includes the count of blocks, access authorizations, and the source files which these blocks refer to.

As discussed above, the structure of the security attribute could be defined as in definition 1.

Definition 1: The security attribute SA is a five-tuple defined as $(owner, fvACL, count, lbACL, SourceFile)$. $lbACL$ is a set of access control lists, one for each logical block. $SourceFile$ is a set of pointers identifying the source files these logical blocks refer to.

B. Access History

Once confidential information is stolen from a standard file system, there is no access history for tracing. To overcome this, the processes affecting files and file views should be recorded. These histories are used to reinforce the security of the file system. They are used for tracing the transfer paths of files and locating the hidden trouble in the intranet to reduce the risks as much as possible.

The information recorded in the access history should include the manipulator's identification, access time, the type of process executed (generally would include read, write, execute and re-grant) and the result of this access.

The data structure for access history could be defined as in definition 2

Definition 2: The Access History AH is a four-tuple defined as $(user, time, type, result)$. These four tuples contain the most important information about the process.

C. Read Processes on a File View

The authorizations of the logical blocks in a file view depend on the authorizations of the original file cited. When users try to read the content of a file view, they can only read the content of original files for which they have read authorization. For example, suppose there is a file view in which the content refers to two logical files are created separately by Bob and Charles. Alice has read authorization for the file created by Bob but does not have read authorization for the file created by Charles. In this way, Alice could only read the content of logical blocks which refer to the file create by Bob.

When users try to access a file view, there are two steps. First, the authorizations to the document of file view should be verified. Second, the authorizations to all the logical blocks in the file view should be verified. The authorizations of the logical blocks are related to the original files in file system which are cited in the file view. The algorithm for the read process on a file view is given in algorithm 1.

Algorithm 1 Read Process on File View

```
if has read authorizations of file view then
  i ⇐ 0
  while i < block_count do
    lb ⇐ current logical blocks
    if has read authorizations of lb then
      accept read process on lb
    else
      deny read process on lb
    end if
    i ⇐ i + 1
  end while
end if
update access history in file view
```

Algorithm 1 has two important advantages:

1) It can prevent the leakage of information seen in the DAC model. In the example in section 1, Alice has created an object and grants read access to Bob. In the framework of file view, Bob can copy Alice's object into a replica created by himself. But when someone else gets the replica, the authorizations of logical blocks are still verified. He could only get the document containing the file view, but not the content in it. This overcomes the security problem in the DAC model.
2) It can protect the confidential information in the file view even if an outsider with a malicious purpose obtains it (for example, by using a trojan horse or some other hacking tool). As in the previous example, even if they can obtain the file view document, they cannot access the content in it. The content authorizations will still be verified whenever he tries to access it.

D. Write Processes on File a View

When users try to write the content of a file view, they intend to write the content in logical blocks. In a database, the data in views is computed from relations, so that modification of the data in a view should be translated into modification of relational tables. As in databases, modification of the content in file views needs to be transformed into modifications of the logical files in the file system.

The write processes can be classified into modify, delete and add.

1) Modify
   When users modify the content of the logical blocks of a file view, the modified content in the file view should be modified in the logical files which are cited in the file views.
2) Delete
   When users delete the content of logical blocks from a file view, they may want to delete the content permanently or just delete the citation in the file view. In the first case, the deletion should deletion the content of the logical file; in the second case, the delete process on the file view should just delete the citation in the file view but leave the logical files untouched. Considering the first case would affect other file view cited, this paper suggest to only delete the citation in the file view.
3) Add
   When users add new content to the file view, the new content cannot be simply stored in the file view. This would result in a state where some logical blocks in the file view do not reference a file. The new content should be stored as a new logical file; then a new logical block can be created automatically in the file view to refer to the new file.

The algorithm is shown in algorithm 2.

Algorithm 2 Write Process on File View

```
if has write authorizations of file view then
  i ⇐ 0
  while i < block_count do
    lb ⇐ current logical blocks
    if has write authorizations of lb then
      check the type of write process on lb
      if modify process then
        write lb into file view
        update content in corresponding source file
      else if delete process then
        delete lb from file view
        update security attribute of file view
      else if add process then
        create new file N
        N ⇐ lb
        create reference of N
        update security attribute of file view
      end if
    else
      deny process on lb
    end if
    i ⇐ i + 1
  end while
end if
update access history of file view
```

V. Conclusion and Future Work

In this paper, we proposed the concept of "File View" to solve some security problems arising in intranets. First, we proposed a hierarchy which uses a structure modeled on database views. There are some challenges in using this hiberarchy in file systems, because of the differing environments between database and file systems. Then, we proposes a framework based on the file view model to overcome these challenges. Finally, under three assumptions, we discuss the access control mechanism and propose read and write process algorithms for this framework.

There are some other current and future directions of work on file view mentioned before in the key issues in this framework:

For the future, we are working on extending the file view mechanism defined in this paper in the following directions:

1) The size of logical blocks in logical file should be defined more flexibly.
2) The security attribute in the file view needs to be protected from malicious modification or destruction.
3) The operating systems in an intranet may be heterogeneous. The file view mechanism needs to be extended to a variety of different operating systems in a single intranet.

ACKNOWLEDGMENT

This work was supported by Research Grant No.60573038 and No.90204011 from National Natural Science Foundation of China. This work was supported by the Research Project No.A1420080183 from Ministry of Education of People's Republic of China.

REFERENCES

[1] R. Richardson, "CSI Computer Crime and Security Survey," *Computer Security Institute*, 2007.

[2] H. Zhang, J. Diao, and Q. Wen, "Secure Files Management System in Intranet," in *Internet Computing in Science and Engineering, 2008. ICICSE'08. International Conference on*, 2008, pp. 306–311.

[3] D. Latham, "Department of Defense Trusted Computer System Evaluation Criteria," *Department of Defense*, 1985.

[4] B. Lampson, "Protection," *ACM SIGOPS Operating Systems Review*, vol. 8, no. 1, pp. 18–24, 1974.

[5] R. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, 1994.

[6] ——, "Authentication, access control, and intrusion detection," *The Computer Science and Engineering Handbook*, vol. 1, pp. 929–1, 1997.

[7] R. Sandhu and Q. Munawer, "How to do discretionary access control using roles," in *Proceedings of the third ACM workshop on Role-based access control*. ACM New York, NY, USA, 1998, pp. 47–54.

[8] R. SMITH and M. T MENDELSSOHN, "DOCUMENT MANAGEMENT AND PRODUCTION SYSTEM," *No.: WO/1991/008538*, 1991.

[9] P. Loscocco and S. Smalley, "Integrating flexible support for security policies into the Linux operating system," in *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference table of contents*. USENIX Association Berkeley, CA, USA, 2001, pp. 29–42.

[10] S. Smalley, C. Vance, and W. Salamon, "Implementing SELinux as a Linux security module," *NAI Labs Report# 01*, vol. 43, 2001.

[11] R. Sandhu, "Lattice-based access control models," *Computer*, vol. 26, no. 11, pp. 9–19, 1993.

[12] E. Bertino, "A View Mechanism for Object-Oriented Databases," in *Proceedings of the 3rd International Conference on Extending Database Technology: Advances in Database Technology*. Springer, 1992, pp. 136–151.

[13] R. Perry and R. Lancaster, "Enterprise Content Management: Expected Evolution or Vendor Positioning?" *Yankee Group Report*, 2002.

[14] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[15] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML) 1.0," *W3C recommendation*, vol. 6, 2000.

[16] W. Fan, C. Chan, and M. Garofalakis, "Secure XML querying with security views," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM New York, NY, USA, 2004, pp. 587–598.

[17] V. Braganholo, S. Davidson, and C. Heuser, "From XML view updates to relational view updates: old solutions to a new problem," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 276–287.

[18] W. Ni and T. Ling, "Update XML data by using graphical languages," in *ACM International Conference Proceeding Series; Vol. 334*. Australian Computer Society, Inc. Darlinghurst, Australia, Australia, 2007, pp. 209–214.

[19] G. Cong, "Query and Update Through XML Views," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 4777, p. 81, 2007.

[20] B. Choi, G. Cong, W. Fan, and S. Viglas, "Updating Recursive XML Views of Relations," *Complexity*, vol. 2, no. 2, p. 2, 2007.

[21] E. Damiani, M. Fansi, A. Gabillon, and S. Marrara, "Securely Updating XML," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 4694, p. 1098, 2007.

[22] J. Clark, S. DeRose, *et al.*, "XML path language (XPath) version 1.0," *W3C recommendation*, vol. 16, p. 1999, 1999.

[23] A. Laux and L. Martin, "XUpdateŁXML Update Language," *XML: DB Working Draft*, pp. 09–14, 2000.

[24] Y. Liang, Y. Ai, H. Dong, and T. Li, "File View: Secure Model in Intranet," in *2009 International Conference on Networks and Digital Society*, 2009, pp. 198–201.