A Performance Evaluation of Logging in XML Databases Using an Xlog File for Trust Based Access Control

Norah Farooqi, Siobhan North
Department of Computer Science
The University of Sheffield
Sheffield, United Kingdom

Abstract

Logging is an important process in databases and is used for recovery and security purposes. Logging in XML databases has rarely been discussed in the literature. In this paper the Xlog file is presented as a dynamic and temporary log file for XML databases. It is used, not for recovery, but to calculate user trust values by recording users' bad transactions and errors. The novelty in this approach is that Xlog relates logging and dynamic access control for XML databases. It is part of a trust based access control approach to XML databases. It supports dynamic access control by tracking user history and is part of a novel approach to solving security issues for XML databases. Experimental work has been performed to test the creation and retrieval processes of Xlog files from a performance perspective and is described here.

1. Introduction

XML databases have become widely used in many applications. Due to the recent increase in their availability, much research has been undertaken to improve their usefulness. XML databases are a relatively new kind of database but, like traditional databases, they require storage strategies and query languages. However, some important areas have not been thoroughly investigated. One of these areas is security in XML databases.

XML databases, like any other databases, can contain a lot of sensitive and personal data. All data, but especially important data need to be protected from unauthorized access either from outsiders or insiders. The traditional access control models focused on outsiders but our trust based access control works for both outsiders and insiders [1, 2, 3]. Trust based access control for XML databases tracks user operations and behavior over time. Therefore, there is a need for logging in XML databases to record users' transactions.

In this paper, we describe an Xlog file that is used to record users' errors and bad transactions. The Xlog file supports the development of access control for XML databases depending on trust. It is designed with a simple structure to be temporary and dynamic and is written in XML to be flexible and easy to understand. The Xlog file's performance has been tested through creation experiments and experiments concerning of the retrieval of data.

The remainder of this paper is categorized as follows. A short literature review of related work appears in section 2. Section 3 explains the contributions of this research and describes the structure of the Xlog file. Section 4 highlights the main features of the Xlog file from several angles. Section 5 discusses the experimental work to evaluate its performance. Finally, section 6 concludes with suggestions for future work and summarizes the conclusion.

2. Related work

The main purpose of logging in normal databases is to record transaction information that is used for recovery when the system crashes and sometimes for concurrency control [4, 5]; it can also be useful for security purposes to track malicious transactions in databases [6]. The main classifications in logging are: undo logging, redo logging, and undo/redo logging, all of which are used mainly to restore data [4, 5]. Logs can be represented as tables in databases or files. Log files can be written in different syntaxes, formats, and languages. Reference [7] suggests that using XML language to create the log file saves both time and space compared to tables.

3. Xlog file

Taking into account the need for logging in XML databases, we introduce the Xlog file for XML databases which unlike conventional log files is focused on security rather than recovery. Thus the Xlog file will:

- Support a secure environment for access control of XML databases.
- Track user operations and behavior by recording and organizing their actions.
- Produce a log file that can be used to calculate a trust value that directly affects user access privileges in a trust based access control model for XML databases.

The structure of the Xlog file for XML databases is shown in Figure 1. It is dynamic and temporary as it is retained only for a certain period of time depending on the organization's policy such as a session, a day, or a week. The Xlog file is written in XML and is processed as a normal XML file. Its structure differs from a normal log file, since it depends on the user identifier instead of time. Using this structure makes capturing user behavior fast and easy. It does not need to record time for each transaction because it is retained for a defined period.

The Xlog file records specific kinds of transactions and errors. Bad transactions are identified by rules defined in the operation policy file that is shown in Figure 2. These rules cover accessing unauthorized nodes or deleting root and parent nodes. Each bad transaction is categorized by its identifier and type. At present, only five basic types of bad transactions are defined but the rules can be easily extended to consider other transaction types depending on the system needs.

Likewise, error rules are defined in the error policy file shown in Figure 3. They focus on accessing nonexistent nodes. All errors are classified by their identifier and type. Like the bad transaction rules they can easily be extended. Furthermore although these rules do not depend on the existence of a schema, a fixed structure for the XML document, they could easily be extended to cover problems that affect the XML file structure when there is a schema.

The Xlog file is a useful tool to calculate trust values for the users by assigning weights for bad transactions and errors. These weights are subsequently used to adjust the users' trust score.

```
<Users>
  <User >
    <ID> 30 </ID>
    <Bad Transaction> 1 </Bad Transaction>
    <Bad Transaction> 4 </Bad Transaction>
    <Error> 1 </Error>
    <Error> 3 </Error>
     …
    </User>
 </Users>
```

Figure1. The Xlog file for XML databases

```
<Bad Transactions>
  <Transaction >
     <ID> 1 </ID>
     <Type> Read unauthorized node </Type >
  </Transaction>
  <Transaction >
     <ID> 2 </ID>
     <Type> Write unauthorized node </Type >
   </Transaction>
   <Transaction >
     <ID> 3 </ID>
     <Type> Delete unauthorized node </Type >
   </Transaction>
   <Transaction >
      <ID> 4 </ID>
      <Type> Delete root node </Type >
    </Transaction>
   <Transaction >
      <ID> 5 </ID>
 <Type> Delete parent node with existing
children</Type >
    </Transaction>
</Bad Transactions>
```

Figure2. The operation policy file.

```
<Errors>
   <Error >
     <ID> 1 </ID>
<Type> Read nonexistent node</Type >
    </Error>
   <Error >
      <ID> 2 </ID>
 <Type> Write nonexistent node </Type >
    </Error>
    <Error >
       <ID> 3 </ID>
  <Type> Delete nonexistent node</Type >
    </Error>
</Errors >
```

Figure3. The error policy file.

4. Xlog file features

The purpose of the Xlog file is to record user behavior. The features of this file are discussed in this section. The majority of its advantages appear through applying a simple structure and using the XML language to write the Xlog file. Consequently, the Xlog file adopts the advantages of XML such as flexibility and simplicity [8, 9, 10]. The important features are discussed below.

- Temporary: the Xlog file is created to be used for a certain period depending on the organization's needs and policies. The organization and the system administrator can define how long the Xlog file may exist. The period can be a session, a day, or a week. After using data from the Xlog file to calculate users' trust and update their privileges, the Xlog file is destroyed. This leads to another feature; the Xlog file consumes little storage.
- Dynamic: one of the main features of the Xlog file is that it is dynamic and updated regularly. It reflects misuse as soon as it occurs. This feature is derived from it transient nature. The Xlog file is temporary; it is amended to record each fresh transaction and thus contains all recent data.
- Consumes little storage: as a result of temporary and dynamic structure, the Xlog file contains only recent processes. Furthermore this storage is only retained for a defined period.
- Flexible: the Xlog file gains its flexibility in that it is written in XML which is intrinsically flexible. XML gives the users the freedom to create their own tags according to their needs. Even the Xlog file structure defined in the previous section, can be changed by the administrator and tags can be amended to serve particular needs.
- Simple: the Xlog file is created to be simple and easy to share between different platforms. Through using the XML language, the Xlog file becomes easy to use and understandable for both humans and machines.
- Interrelated with other files: the Xlog file can be related easily and smoothly with the operation policy file and the error policy file. The content refers to other files by using reference identification <ID>. The organization can extend errors and bad operations types in the policy files and these can be automatically related to and recorded in the Xlog file.
- Consistent environment: since the motivation of creating the Xlog file is to serve XML databases' security, the Xlog file is obviously best written in the XML language.

5. Experiments results

In this section, the experiments to evaluate the performance of the Xlog file for XML databases are described. These experiments check the Xlog file performance speed over time from two perspectives. The first is to test the creation process of the Xlog file. The second perspective focuses on evaluating the reading process and retrieval of data. Both perspectives were implemented in three parts depending on the type of processes in the Xlog file. These viewpoints were evaluated in three ways; with errors only, bad transactions only, and a mixture of both errors and bad transactions. Each of these parts was tested in different sizes according to the number of processes in the Xlog file.

The Xlog file experiments were executed on a PC with 2.40 GHz Intel ® Core ™ i5 CPU, 4 GB of main memory, and Windows 7 operating system. The processes of Xlog file creation and retrieval data are run using Java Language (JDK 1.7.0) and NetBeans IDE 7.0.1 platform framework. These experiments were performed with various Xlog files that had different numbers of errors, bad transactions, or both errors and bad transactions.

5.1. Evaluating the creation process of the Xlog file

The main goal of the creation experiments is to measure and evaluate the time required to create the Xlog file and record processes. As mentioned in the previous section, this experiment was executed with three types of Xlog files. The first file type consists of errors. The second file type includes only bad transactions. The third has errors with bad transactions. Then these three Xlog files were tested in several steps depending on the number of recorded processes. The first and second Xlog files start from 10 processes with either errors or bad transaction until they reach 100 processes. In each step, the number of processes is increased by 10. Since the third Xlog file has both errors and bad transactions, the number of process is shared equally between them. This third Xlog file runs from 20 to 200 processes. The number of processes is increased by 10 errors and 10 bad transactions each step.

The results of creating the first Xlog file that contains errors only and the second Xlog file that includes only bad transactions were almost identical. This is because the time consumed for recording the same number of processes in different Xlog files is similar to each other even if the type of process is different. This means the creation of the Xlog file is affected by the number of recorded processes regardless of their type. The required time to create the Xlog file with 10 processes with either error or bad transactions is 122 milliseconds. This time increases regularly by around 50 milliseconds when the number of errors or bad transactions is raised by 10 processes each time. When the number of either errors or bad transactions is 100 processes, the time reaches around 566 milliseconds.

The time consumed to create the third Xlog file is increased by around 100 milliseconds when the number of processes is increased by 20 processes. At the start point, when the number of errors is 10 and the number of bad transactions is 10, the time for creating the Xlog file with these 20 processes is 186 milliseconds. The creation time grows markedly until reaching 1098 milliseconds when there are 200 processes, which are 100 errors and 100 bad transactions. Figure 4 shows the result of creating the Xlog file with both errors and bad transactions.

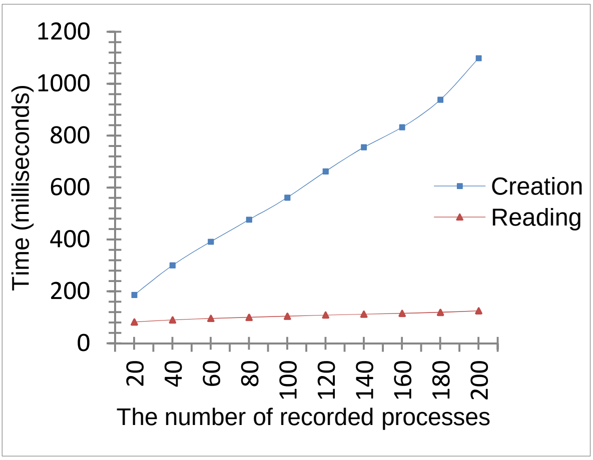

Figure4. The required time for the Xlog file with both errors and bad transactions.

5.2. Evaluating the reading process of the Xlog file

The retrieval experiments were used to evaluate the performance of reading the Xlog files. These experiments were tested with the same procedures as the creation experiments. The experiments were executed by using the same test factors, which are the process types and the number of processes.

The time required for reading the Xlog files that contained either only errors or only bad transactions is similar. The time starts increases from 79 milliseconds to read 10 processes up to 103 milliseconds to read 100 processes. The time is increased by around four milliseconds when the number of errors or bad transaction is increased by 10 processes. The consumed time to read the third Xlog file type, which has both errors and bad transactions, is increased by around six milliseconds when the number of processes grows by 20 processes. The time to access the third Xlog file starts from 82 when the number of processes is 20 and ends by 125 milliseconds when the number of processes is 200. Figure 4 explains the results of reading and accessing the Xlog file when half of the processes are errors and the other are bad transactions.

The results of creating and reading the Xlog file experiments explain some points. In general, the time consumed for the reading process is always less than the time for creation. The creation and retrieval processes are not affected by the type of processes, whether they are errors or bad transactions. Both experiments are affected markedly by changing the number of processes in each step.

6. Conclusion and future works

The Xlog file consists of a dynamic and temporary log file for XML databases. This approach focuses on using logging for security issues related to access control. The XML log file is a part of trust based access control for XML databases. It is used to evaluate user behavior by recording user transactions and errors. The rules for bad transactions and errors are defined and can be extended according to need.

The performance of the Xlog file is evaluated during both creation and retrieval. The experiments were executed with errors only, bad transactions only or both. These three ways were run with several sizes of file. The experimental results show that the reading process is faster than the creation process. Both creation and reading processes are affected by the increment in the number of processes in the Xlog file. The maximum number of errors and bad transactions reaches 200 processes. Due to temporary and dynamic features of the Xlog file, there is no need to go higher.

The logging in XML databases is important to provide a secure environment. This topic needs further investigation and more work. Logging could be extended to cover the traditional features of the log file that is used for recovery purposes in relational databases to XML databases.

7. Acknowledgements

This paper is supported by Umm Al Qura University on behalf of the Higher Ministry of Education of Saudi Arabia.

8. References

[1] N. Farooqi and S. North, “Logging in XML Databases: Xlog File for Trust Based Access Control”, World Congress on Internet Security (WorldCIS-2012), IEEE Xplore, Ontario, Canada, 2012, PP. 174-175.

[2] N. Farooqi and S. North, “Trust Based Access Control for XML Databases”, The 6th International Conference for Internet Technology and Secured Transactions (ICITST),IEEE Xplore, Abu Dhabi, UAE, 2011, PP. 764-765.

[3] N. Farooqi and S. North, “Developing a Dynamic Trust Based Access Control Model for XML Databases”, Department of Computer Science Research Memoranda CS-11-09, University of Sheffield, UK, 2011.

[4] H. Molina, J. Ullman and J. Widom, Database Systems The Complete Book, 2nd ed, Pearson International Edition, USA, 2009.

[5] R. Elmasri and S. Navathe, Fundamentals of Database Systems, 5th ed, Pearson International Edition, USA, 2007.

[6] F. Etho, K. Takahashi, Y. Hori and K. Sakurai, “Study of Log File Dispersion Management Method”, 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT), IEEE Computer Society, Seoul, Korea,2010. pp. 371-374.

[7] F. Wang, X. Zhou and C. Zaniolo, “Using XML to Build Efficient Transaction-Time Temporal Database Systems on Relational Databases”, The 22nd International Conference on Data Engineering (ICDE), IEEE Computer Society, Atlanta, Georgia, 2006, pp.131-134.

[8] E,Ray, learning XML, O’REILLY.

[9] W3School, “Introduction to XML”, http://www.w3schools.com/xml (8-9-2012).

[10] W3C, “What is XML?”, http://www.w3.org/standards/xml/core (9-9-2012).