디지털산업정보학회 논문지 제15권 제3호-2019년 9월
http://dx.doi.org/10.17662/ksdim.2019.15.3.101

A Study on Database Access Control using Least-Privilege Account Separation Model

Jang Youngsu*

최소 권한 계정 분리 모델을 이용한 데이터베이스 엑세스 제어 연구

장 영 수

<Abstract>

In addition to enabling access, database accounts play a protective role by defending the database from external attacks. However, because only a single account is used in the database, the account becomes the subject of vulnerability attacks. This common practice is due to the lack of database support, large numbers of users, and row-based database permissions. Therefore if the logic of the application is wrong or vulnerable, there is a risk of exposing the entire database. In this paper, we propose a Least-Privilege Account Separation Model (LPASM) that serves as an information guardian to protect the database from attacks. We separate database accounts depending on the role of application services. This model can protect the database from malicious attacks and prevent damage caused by privilege escalation by an attacker. We classify the account control policies into four categories and propose detailed roles and operating plans for each account.

Key Words : Database Account Separation, Information Guardian, Account Design and Control Policy

Ⅰ. 서론

The Database System (hereinafter called "DBS") should provide a means to ensure certain users or group of users can access selected portions of a database. In addition, the DBS includes database security and authorization subsystems responsible for maintaining the security portion of the database (e.g., specific data files and records)[1]. DBS administration is performed on root or elevated system accounts, which is necessary to protect the database environment by restricting the run authority.

The Dualistic Account Separation Model (DASM)[2] uses only the owner account (a database user who creates objects), which has the authority

* Dept of Smart Software, Korea Polytechnics (Corresponding Author)

to create and modify the schema, as the application service account, which serves the application. However, this model has a vulnerability that can expose sensitive personal information because there is no separate access control constraint for accessing the schema when an application is exposed to external attacks (e.g., blind injection). This common practice is due to lack of database support for connection pooling, large number of users, and row-based database permissions[3]. Moreover, if the logic of the application is wrong or vulnerable, there is a risk of exposing the entire database.

Therefore, the owner account should be kept separate, because it should control the database account to protect the database from a malicious attacker. Database account separation can limit the privileges of the potentially buggy application logic.

In this paper, we propose a least-privilege account separation approach to protect the database against attackers. The contributions of this paper are as follows:

1) A business role-based account separation model that can detect access control vulnerabilities.
2) An implementation of our analysis which defines and designates the roles and permissions of each account with four categories.
3) An evaluation of our approach on real-world relational database. Our approach is able to prevent vulnerabilities before program execution.

The remainder of this paper is organized as follows: Section II reviews related works, Section III details our system model, Section IV describes the implementation, Section V presents our evaluation, and finally, Section Ⅵ presents our conclusions.

Ⅱ. Related works

Colombo and Ferrari[4] focused on isolating web application users from each other. Using their approach, application access is limited according to the identity of the user logged in over Secure Sockets Layer (SSL). The separation is implemented in the application to prevent one user from accessing another user's data. However, our approach enables fine-grained access control for multiple modules of a program, which can work on behalf of one or more users.

Moon and Jeong[5] proposed using a user's credentials to forward commands to the database only if the user is logged in. This includes data stored for the purpose of running the application itself, as well as data that users can access. However, they only consider the use-case of secure application servers running buggy applications. However, our approach is that account separation is performed to prevent one user from accessing another user's data.

Guarnieri *et al.*[6] proposed a method in which each module in a program explicitly recommends the database access required to do the job. This allows programmers and code reviewers to better understand how a given module affects or relies on a particular dataset in the database. However, this method may incur additional overhead due to the need to modify database access control when changing program modules. Our approach is business role-based account separation, which reduces the manual effort required to modify the application.

Ⅲ. System model

The method of enforcing discretionary access control in the DBS is based on the granting and revoking of privileges[7]. Our approach provides selective access to each relation in the database based on specific accounts. Therefore, we can control the privilege to access each individual relation or object in the database.

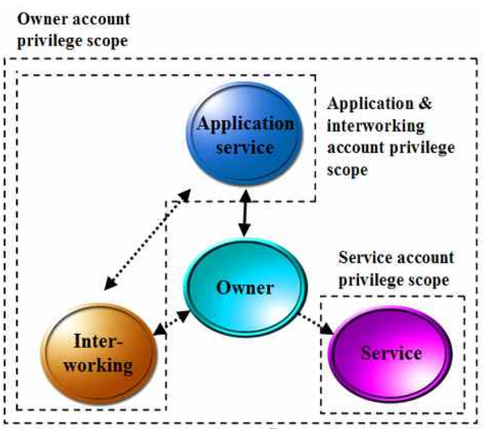


Fig. 1. DBS Least-Privilege Account Separation Model (LPASM).

3.1 Least-privilege account separation model

We divide database accounts into four categories based on the business roles. Fig. 1 illustrates our DBS Least-Privilege Account Separation Model (LPASM). The DBS accounts are separated into the owner, application service, interworking, and service accounts. The dotted line arrow in Fig. 1 represents limited data access and the solid line arrow represents normal data access. The application service account and interworking account share data with each other. This signifies data interworking between the application and external systems.

The owner account generates schema from the DBS and has all privileges on the schema. In contrast, general users can only access the access-permitted schema according to the granted access privilege roles. The application service account and interworking account are for internal and external users, respectively. For these accounts, access privileges on the schema are granted by the owner account to give internal and external users access to the database through application programs. The service account is granted limited access to only the database schema and is used for statistical databases, such as the On-Line Analytical Process (OLAP) and census.

3.2 Account management policy

The LPASM manages DBS accounts through 1) resource control and 2) access privilege.

3.2.1 Resource control

As part of the account security domain, system resources can be limited to an account based on permissions for Data Control Language (DCL), Data Definition Language (DDL), and Data Manipulation Language (DML)[8].

Owner accounts can use DCL, DDL, and DML without any restrictions. The objects (e.g., tables, views, procedures, etc.) that other accounts use are generated and managed using DDL. The application service account is used by many users. When this account is granted DML permission by the owner account, access to the tables, synonyms, and views is possible. The system is implemented by setting a buffer size that is impossible to use for batch data processing in order to prevent unauthorized processes[9]. This prevents unauthorized users or persons from accessing the system itself as a whole either to obtain information or to make changes.

The interworking account enables data interchange between the internal database and the external system. This account gains access to the tables, synonyms, and views after being granted the DML privilege by the owner account; however, access to the schema is more restricted than that of the application service account.

The service account is for statistical databases and is granted the most limited DML privilege. Specifically, it retrieves information from the database using the SELECT query.

3.2.2 Access privilege

The LPASM enables the DBS to apply account privileges. For example, an application account in the database may need to be modified to operate with access privileges. Modification of the database account privileges is more likely to lead to maintenance problems. However, the LPASM implementation presents an attractive option from a different viewpoint. Protection from SQL Injection Attacks (SQLIA) is applied to any application that runs on a modified database[9]. This is a distinct advantage and may even be preferable in certain practical situations. In addition, the service account cannot access the objects and relations used by the application service account because it has the minimum access privilege role. On the other hand, the owner account has the

Book

BNO	Title	Price	PNO	ANO
BN01	Abel's Island	₩ 20,000	PN01	AN01
BN02	A Solitary Blue	₩ 23,000	PN02	AN02
BN03	Bluebird	₩ 18,000	PN03	AN01
BN04	Stories	₩ 32,000	PN02	AN03
BN05	Dracula	₩ 18,000	PN03	AN03

Publisher

PNO	Pname	Phone
PN01	Dong-H	02-1381-3429
PN02	Woo-Ri	031-456-9123
PN03	Ha-Nul	02-8417-9813

Author

ANO	Aname	Age
ANO1	Jang	46
ANO2	Lee	71
ANO3	Hong	53

Invoice

INO	Type	BNO	Odate	Inumber	Istate	DNO
1	Magazine	BN01	01/15/2019	201901150001	Received	DN01
2	Novel	BN02	01/19/2019	201901190002	Received	DN02
3	Magazine	BN03	01/30/2019	201901300003	Rejected	DN02
4	Fairy tale book	BN04	02/11/2019	201902110004	Sent	DN03
5	Magazine	BN03	02/12/2019	201902120005	Sent	DN01
6	Fairy tale book	BN05	02/18/2019	201902180006	Sent	DN03
7	Novel	BN01	03/10/2019	201903100007	Rejected	DN03
8	Fairy tale book	BN02	03/12/2019	201903120008	Rejected	DN02
9	Fairy tale book	BN03	03/22/2019	201903220009	Rejected	DN01
10	Magazine	BN05	03/28/2019	201903280010	Sent	DN01

Delivery

DNO	Dname
DN01	K.Expr
DN02	Hala
DN03	Mulang

Fig. 2. Simple bookstore real-world relational database.

maximum access role and can access all objects and relations.

Ⅳ. Implementation

4.1 Simple relational database test-suite

Fig. 2 shows a simple bookstore real-world relational database. Each book and delivery can have one or more invoices, and each publisher and author can have one or more books. The “INO” column shows the order serial number. Fig. 2 is processed as follows:

(1) Relational Table schema: 5 tables
- {*Book, Publisher, Author, Invoice, Delivery*}

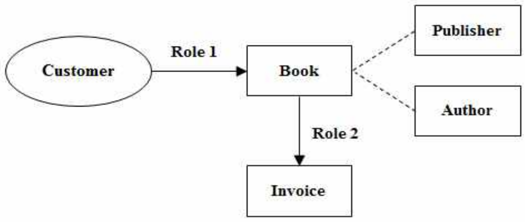
(2) Explicit referential integrity relation:
- {*Invoice.BID, Book.BID*}

(3) Implicit referential integrity relation:
- ({*Publisher.PNO, Book.PNO*}, {*Author.ANO, Book.ANO*}, {*Invoice.DNO, Delivery.DNO*})

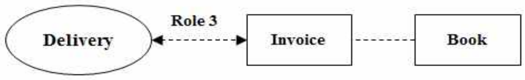
(4) Table primary key columns:
- ({*Book.BID*}, {*Published.PNO*}, {*Author.ANO*}, {*Invoice.INO*}, {*Delivery.DNO*})

4.2 Definition of business roles

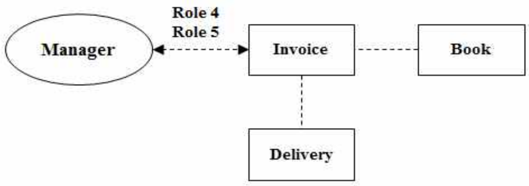
The real-world business roles for a simple bookstore are illustrated in Fig. 3 (line arrow: normal data access, dotted line: limited data access). The customer can select books to order from the bookstore application (Role1), place an order for the books (Role2). The delivery service delivers the ordered books to the customer (Role3). The manager can check the list of orders and deliveries according to the order date (Role4, Role5) and manage the bookstore application (Role6). Therefore, the customer, delivery personnel, and manager can



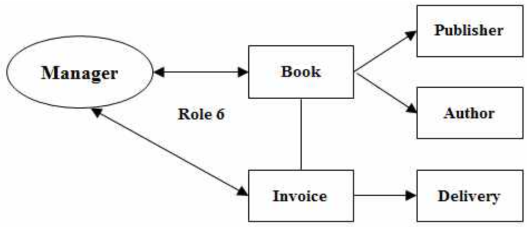
(a) customer subject.



(b) delivery service subject.



(c) manager subject (i).



(d) mamager subject (ii).

Fig. 3. Subjects to use a database table

be defined as subjects who can use the database tables. Our simple real-world bookstore business roles are summarized in Table 1.

4.3 Least-privilege account separation

We separate the simple real-world bookstore database accounts based on business roles as follows. *TM* represents a set of transactions from each table:

Table 1. Simple real-world bookstore business roles

Business roles	Description
Role 1	Customer user orders books in bookstore application.
Role 2	The books selected by the customer user are inserted in the INVOICE table.
Role 3	Delivery user service refers to ordered books through an external system.
Role 4	Manager user may check the order list of books by Odate.
Role 5	Manager user may check the order list of delivery by Odate or Inumber.
Role 6	Manager user manages bookstore application.

(1) Owner account: Manager
- Related table list: {Book, Invoice, Publisher, Author, Delivery}
- Normal data access table list: {Book, Invoice, Publisher, Author, Delivery}
- Referential data access table list: None
- Limited data access table list: None
- Shared target table list: {Book, Invoice}
  - ✔ TM(Book) = {Read, Insert, Update, Delete}
  - ✔ TM(Invoice) = {Read, Insert, Update, Delete}

(2) Application service account: Customer
- Related table list: {Book, Invoice, Publisher, Author}
- Normal data access table list: {Book, Invoice}
- Referential data access table list: {Publisher, Author}
- Limited data access table list: None
- Shared target table list: {Book, Invoice}
  - ✔ TM(Book) = {Read}
  - ✔ TM(Invoice) = {Read, Insert}

(3) Interworking account: Delivery
- Related table list: {Book, Invoice}
- Normal data access table list: None
- Referential data access table list: {Book}
- Limited data access table list: {Invoice}
- Shared target table list: {Invoice}
  - ✔ TM(Invoice) = {Read}

(4) Service account: Manager
- Related table list: {Book, Invoice, Publisher, Author, Delivery}
- Normal data access table list: None
- Referential data access table list: {Book, Publisher, Author, Delivery}
- Limited data access table list: {Invoice}
- Shared target table list: {Invoice}
  - ✔ TM(Invoice) = {Read}

Table 2. Results of the evaluation using the empirical test-suite

Application	Size (LOC)	Modifications	LPASM	
			Access privilege prevention	Vulnerability detection
JForum	219	91	Success	N/A
Drupal	518	32	N/A	Success
WordPress	1,092	40	N/A	Success

Ⅴ. Evaluation

5.1 Empirical evaluation of the test-suite

To evaluate the LPASM verification, we used three empirical test-suites. Our target test-suite is a small relational database with a functionality that can be separated into relatively dependent modules. The following describes the code bases of these projects: JForum, Drupal, and WordPress. The following describes the code bases of these projects:

(1) JForum is a Java open source program to message board system that runs several forums. Designed as a separate set of modules, it is a good candidate for account separation. We retrofit JForum's "*posts*" module ver.2.1.8 to work with LPASM. It is the most privileged module that requires full access to the database tables.

(2) Drupal is a open source content management program written in PHP. Version 5.10 of the "*Brilliant Gallery*" plugin could allow an attacker to retrieve the administrative password of a Drupal-based website.

(3) WordPress is a open source content management program based on PHP and MySQL database. "*GNU Commerce*" plugin of WordPress ver.2.7 has a SQLIA that allows an attacker to access the use account database.

Table 2 summarizes the results of evaluation using the open-source test suite. The application size is given in the LOC. The third column shows how many lines of code were added and altered. We evaluated JForum for "Access privilege prevention" and Drupal and WordPress for "Vulnerability detection", respectively. The final two columns shows that Access privilege prevention and Vulnerability detection result by our LPASM. Table 4 presents an example of a SQLIA for WordPress. It is possible with the SQLIA through {$sod} in the *order by* clause on line 9.

```
 1.  if (!$sst) {
 2.      if ($board['bo_sort_field']) {
 3.          ...
 4.      }
 5.  } else {
 6.      ...
 7.  }
 8.  if ($sst) {
 9.     $sql_order = " order by {$sst} {$sod}";
10.  }
```

Fig. 4. Example of SQLIA for order by clause of WordPress

Ⅵ. Conclusion

This paper proposes a database LPASM that can be applied to extend the security in most database systems. We separate database accounts according to the business role to protect the database from attacks and prevent damage caused by the elevation of privileges by an attacker. In addition, we separate the account management policies into four categories and proposed detailed roles and operational plans for each account. Our approach provides evidence that it is possible to successfully design retrofitting techniques that guarantee security in legacy applications and eliminate well-known attacks.

References

[1] S.M. Groomer and U.S. Murthy, "Continuous auditing of database applications: An embedded audit module approach," In Continuous Auditing: Theory and Application, 2018, pp.105-124.

[2] Separation of system resources guideline, UC Berkely Information Security Office, https://security.berkeley.edu/separation-system-resources-guideline

[3] M. Malik and T. Patel, "Database security attacks and control methods," International Journal of Information, Vol.6, 2016, pp.175-183.

[4] P. Colombo and E. Ferrari, "Enforcement of purpose based access control within relational database management systems," IEEE Transactions on Knowledge and Data Engineering, Vol.26, No.11, 2014, pp.2703-2716.

[5] SU. Moon and YJ. Jeong, "System and method for authentication," DC: U.S. Patent and Trademark Office, 2016.

[6] M. Guarnieri, S. Marinovic, and D. Basin, D. "Strong and provably secure database access control," In 2016 IEEE Euro S&P, 2016, pp.163-178.

[7] CK. Wee and R. Nayak, R. "A novel database exploitation detection and privilege control system using data mining," In Modern Approaches for Intelligent Information and Database Systems, 2018, pp.505-516.

[8] N. Batra and H. Aggarwal, "Autonomous multilevel policy based security configuration in distributed database," International Journal of Computer Science Issues(IJCSI), Vol.9, No.6, 2012, pp.170-176.

[9] YS. Jang and JY. Choi, "Detecting SQL injection attacks using query result size," Comput. Sec., Vol. 44, 2014, pp.104-118.

[10] JS. Park and CS. Kim, "Research trends analysis of big data: focused on the topic modeling," The korea society of digital industry and information management, Vol.15, No.1, 2019, pp.1-7.

[11] ES. Cho, SY. Min, SH. Kim, and BG. Kim, "Development of extracting system for meaning subject related social topic using deep learning," The korea society of digital industry and information management, Vol.14, No.4, 2019, pp.35-45.

■ 저자소개 ■

장 영 수
Jang Youngsu

2017년 12월~현재
한국폴리텍대학
스마트소프트웨어학과 조교수
2019년 8월 고려대학교 컴퓨터학과(공학박사)
2011년 2월 고려대학교
소프트웨어학과(공학석사)

관심분야 : 정보보안, 보안코딩,
시큐어프로그래밍
E-mail : jyskkh@naver.com

논문접수일 : 2019년 8월 20일
수 정 일 : 2019년 9월 8일
게재확정일 : 2019년 9월 11일