SQRM: An Effective Solution to Suspicious Users in Database

Dai Hua
College of Information Science & Technology, Nanjing University of Aeronautics & Astronautics
Nanjing, China
dai_hua@nuaa.edu.cn

Qin Xiaolin
College of Information Science & Technology, Nanjing University of Aeronautics & Astronautics
Nanjing, China
qinxcs@nuaa.edu.cn

Zheng Guineng
College of Information Science & Technology, Nanjing University of Aeronautics & Astronautics
Nanjing, China
redzgn@nuaa.edu.cn

Li Ziyue
College of Information Science & Technology, Nanjing University of Aeronautics & Astronautics
Nanjing, China
yenuo_1108@msn.com

***Abstract*—Since traditional database mechanisms such as identity authentication and access control, can be fooled by authorized but malicious users, to solving the problems, three key techniques namely intrusion detection, damage quarantine and recovery are studied for decades to implement survival database systems. However, these techniques are all built on identification of malicious behaviors, which is much more complex, sluggish and inefficient than the identification of suspicious behaviors because the former need more evidence than the later. This paper proposes an effective security mechanism by focusing suspicious users, namely suspect quarantine and recovery method denoted as SQRM, to increase the attack resistance of databases. It isolates invalid data transparently from trustworthy users to prevent further damage by suspicious users suspected to be malicious, while still maintaining continued availability for their data access operations to minimize loss of productive work in the case of incidents that they are indeed innocent. And when they are proved innocent or malicious, all invalid data caused by them will be concurrently recovered. Using SQRM is sufficiently effective to improve the survivability for database.**

Keywords-database security; survival database; suspicious user quarantine; invalid data recovery

I. INTRODUCTION

Database security, an issue focuses on data confidentiality, integrity and availability [1] has drawn a considerable amount of interest since database was used in data-intensive and security-sensitive applications, such as credit card billing, banking, air traffic control and online stock trading. Traditional database security technologies, such as identity authentication, access controls and encryption concentrate on database confidentiality, which is often powerless for malicious attacks including authorized abusing, hackings, and so on. So many attacks succeeded, which had fooled traditional database protection mechanisms, because in reality not all attacks can be averted at their beginning. Consequently, survival database systems (or attack resistant，or intrusion tolerant, or self healing database systems) [2-5] are of significant concern, which can survive malicious attacks, and provide continuous but maybe degraded service when the damage is being recovered.

To implement survival database systems, three key technologies namely intrusion detection (ID) [6-9], damage quarantine (DQ) [10-14] and damage recovery (DR) [15-19] have been studied for decades. ID detects malicious attacks including malicious users' transactions and operations. DQ isolated all invalid data result from corruption of malicious attacks detected by ID, and ensure invalid data not be accessed by trustworthy users, otherwise it might cause damage spreading [20] (if data x is invalid, operation $y=x+100$ will have damage spread to y). DR repairs all invalid data and improves availability of database. Apparently, ID, DQ and DR are built on the identification of malicious behaviors. Actually, the identification of suspicious behaviors could be more efficient, easier and earlier than identification of malicious behaviors in practical applications, because the latter needs more evidence to investigate. Obviously trustworthy data would be in danger as long as the suspicious behaviors exist because they could be indeed malicious. Therefore if we can control suspicious behaviors immediately after it has been detected, the indeed malicious attacks will be prevented earlier; the scare of damage will be decreased and the recovery of database will be easier and more efficient.

Here, we focus on suspicious users, whose behaviors are suspicious, but still need further investigation and more evidence to finally confirm their uncertain identities innocent or malicious. For example, when an accountant logs on banking system at 2:00 am as user "Jack" who usually works in the daytime, this abnormal logon will make Jack suspicious. The real identity of this Jack is uncertain. Perhaps Jack himself is working overtime involving an urgent task, or this Jack is a malicious hacker who cheated jack's identity. More evidence is needed to make the right judgment. What could we do if we encounter this suspicious Jack? The naive rejection would cause loss of his constructive work if he is indeed Jack himself. On the other hand, the simple permission may cause further damage if he is a hacker. As a result, to handle the above dilemma, necessary measures should be taken toward suspicious users.

 ISBN:978-1-61208-115-1

In order to solve problems of suspicious users，**S**uspect **Q**uarantine and **R**ecovery **M**ethod (SQRM) is presented by us in this paper. SQRM has two phases of work: Suspicious User Quarantine Phase (SUQ-Phase) and Invalid Data Recovery Phase (IDR-Phase). As shown in Figure 1, user *s* was trustworthy before it was detected suspicious at time t_1, proved innocent or malicious at time t_2, and invalid data recovery was accomplished at time t_3. SUQ-Phase starts from t_1 to t_2, while IDR-Phase originates from t_2 to t_3. The key points of SQRM are as follows:

- In SUQ-Phase, *s* will be quarantined immediately once *s* is detected suspicious, but instead of being stopped arbitrarily, *s* will be able to continue its work. An extra value of data is provided to *s* for its data accessing. Meanwhile, the invalid data caused by *s* will be access denied by trustworthy users to prevent damage spreading since it is recovered.
- In IDR-Phase, when *s* is proved innocent or malicious, all the extra value of data caused by *s* will be identified. If *s* is proved malicious, the extra value of data caused by *s* will be incorrect and discarded directly, but if *s* is proved innocent, it will be identified as correct and written back into the database. All the invalid data caused by *s* will be recovered at last.

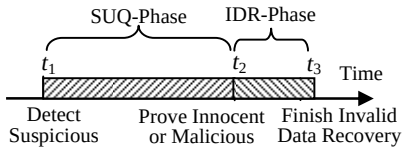

Figure 1. SQRM workflow

There is an evaluation criterion of judging the strategy of handling suspicious users: No Leakage of Invalid Data (NLID). NLID requires that invalid data should be isolated from trustworthy users, which means that they would not access any invalid data, so the damage spreading will be prevented. Meanwhile all invalid data will be recovered trustworthy, and the integrity and correctness of database will be assured.

To satisfy the NLID criterion and make sure suspicious users working under quarantine, we present a data model of SQRM firstly, which characterizes the value types of data items maintained by trustworthy and suspicious users. Then we provide the user operation isolation algorithm and on-the-fly invalid data recovery algorithm based on the data model, the former algorithm will not only isolate all invalid data from trustworthy users to prevent damage spreading, but also provide extra value of data items to suspicious users to continue their work, while the later will recover all the invalid data in IDR-Phase of suspicious users.

The rest of the paper is organized as follows. Section 2 discusses the related works. In Section 3, we give the theoretical model and algorithms of SQRM. Finally, Section 4 summarizes what we have done and future work of this paper.

II. RELATED WORKS

Since current research mostly relies on ID, DQ and DR method to solve malicious attacks to implement survival database systems. Only a few studies of suspicious users have been proposed. In current research of suspicious uses, Liu et al. proposed a data attack isolation system (DAIS) using data versions [21-23]. The main point of DAIS includes two steps as following: In step 1, once a user *s* is detected suspicious, it will be isolated from other users according to isolation protocol, and suspicious version data will be created and maintained by *s* when *s* performs updates in database. Meanwhile trustworthy users can not access any suspicious version data. In step 2, when *s* is proved malicious, all the suspicious version data maintained by *s* will be discarded. And when *s* is proved innocent, to resolve the conflicts between trustworthy and innocent transactions statically or dynamically, precedence graph of transactions will be created. Because the acyclic precedence graph means no conflict of transactions and consistence of database, if cycles appear in precedence graph, related committed transactions (trustworthy or innocent) incurring cycles will be backed out to break cycles thus guaranteeing precedence graph acyclic (Back out a transaction means to restore every data item updated by it to the latest value before updates). After precedence graph is established, the suspicious version data (which is indeed trustworthy) maintained by the innocent user *s* will be adopted to replace the corresponding trustworthy version data (which are indeed invalid). Till now the processing of suspicious user *s* is accomplished.

However, DAIS still has shortcomings in damage spreading. We illustrate it by giving an example as follows:

Example 1: Suppose user *s* is detected suspicious at time t_1, proved innocent at time t_2, and user *u* is always trustworthy. During time interval [t_1, t_2], *s* executes transactions T_{s1} and T_{s2} while *u* executes T_{u1}, T_{u2} and T_{u3}. Details of these transactions are shown in Figure 2. We denote trustworthy version data as $x[T]$, and suspicious version data as $x[s]$ maintained by *s*.

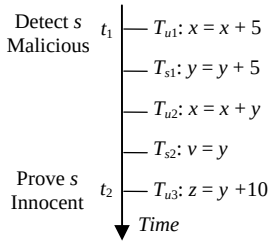

Figure 2. Operations and History of Transactions

In [t_1, t_2], *s* is suspicious. According to DAIS, when 5 transactions are finished, $x[T]$ and $z[T]$ will be updated while $y[s]$ and $v[s]$ are created ($x[T]$=21, $z[T]$=18, $y[s]$=13 and $v[s]$=13 as shown in TABLE I). When *s* is proved innocent at t_2, an acyclic precedence graph *G* of committed transactions will be created as shown in Figure 3. Obviously there is no conflict of transactions. Since the proved

 ISBN:978-1-61208-115-1

innocence of *s* indicates that T_{s1} and T_{s2} are trustworthy, *y*[*s*] and *v*[*s*] are actually trustworthy too, they will be adopted to replace *y*[*T*] and *v*[*T*] (*y*[*s*]=13 replace *y*[*T*]=8 and *v*[*s*]=13 replace *v*[*T*]=8, then remove *y*[*s*] and *v*[*s*] as shown in TABLE I). At this moment we are certain about that *y*[*s*] and *v*[*s*] should be written back into *y*[*T*] and *v*[*T*]. But when *s* is executing its operations with suspicious identity in [t_1, t_2], y[*s*] and v[*s*] could also be discarded if *s* is proved malicious. Therefore *y*[*T*] and *v*[*T*] are invalid since *y*[*s*] and *v*[*s*] are created till they are replaced by *y*[*s*] and *v*[*s*], but these invalid data is not isolated from trustworthy users in DAIS. If they are accessed by trustworthy users, leakage of invalid data might cause damage spreading (In example 1, *y*[*T*]=8 and *v*[*T*]=8 are invalid since T_{s1} and T_{s2} commit till they are replaced with *y*[*s*]=13 and *v*[*s*]=13, but the trustworthy transactions T_{u2} and T_{u3} both read the invalid *y*[*T*]=8, when T_{u2} and T_{u3} commit, *x*[*T*] and *z*[*T*] will be infected invalid too.) According to the above discussion, we can see that DAIS can not satisfy NLID criterion.

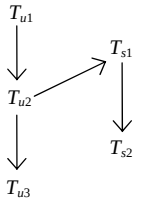

Figure 3. Precedence Graph G

TABLE I. VALUE OF DATA VERSIONS

Data Item	***x***		***y***		***z***		***v***	
Data Versions	***x*[*T*]**	***x*[*s*]**	***y*[*T*]**	***y*[*s*]**	***z*[*T*]**	***z*[*s*]**	***v*[*T*]**	***v*[*s*]**
5 Transactions Finished	21	—	8	13	18	—	8	13
After Data Version Replacement	21	—	13	DEL	18	—	13	DEL

a. "DEL" means data deletion; b. "—" means data not exists; c. *x* = y = *z* =*v* = 8 at beginning

III. SQRM METHOD

We assume that all operations of users are trustworthy when the users are trustworthy, suspicious when the users are suspicious and malicious when the users are malicious. The identity transition diagram of user is shown in Figure 4. A trustworthy user can be detected suspicious, and a suspicious user can be proved innocent (trustworthy) or malicious. Furthermore, because suspect detecting method is not the purpose of this paper, we assume that detection of suspicious users is accurate and prompt (In fact, suspicious detection could be simple and efficient in practical applications. For example, when a user logs on system from an unknown address or an abnormal time, this user could be suspicious).

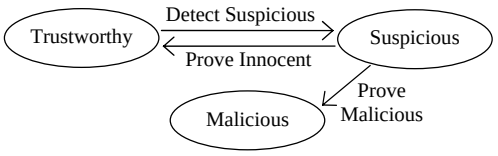

Figure 4. Identity transition diagram of users

In this section, we will formally describe the theoretical concepts and algorithms of SQRM, including data modal, user operation quarantine algorithm and invalid data recovery algorithm. The key points of SQRM are to isolate invalid data from trustworthy users to prevent damage spreading, provide the quarantined extra value of data items to suspicious users for catching results of their work instead of stopping them arbitrarily, and recover the invalid data as soon as possible.

A. Data Model

A database system could be seen as a set of data items, we denote it as $DB=\{x_1, x_2 \ldots x_n\}$. There are two value types of data item as shown in definition 1.

Definition 1. Each data item $x_i \in DB$ could have two value types:

- Normal Type value (NT-value). It is maintained by all trustworthy users and denoted as x_i^N. We use $DB^N = \{x_i^N \ldots x_j^N\}$ to represent NT-value set.
- Quarantined Type Value (QT-value). It is maintained only by suspicious user who created it and denoted as x_i^Q. The user maintaining x_i^Q is denoted as *owner*(x_i^Q). We use $DB^Q = \{x_i^Q \ldots x_k^Q\}$ to represent QT-value set.

NT-value and QT-value of a data item are transparent to users. If a user submits to accessing a data item x_i successfully, only one value (x_i^N or x_i^Q) of x_i will be accessed. Data accessing measures follow user operation isolation algorithm in Section 3.2. When database is initialed to start service, only NT-value of data items exists, and all data items are trustworthy so as to be able of being accessed by trustworthy users at that time. We give the definition of trustworthy data as follows.

Definition 2. Trustworthy Data. For a data item $x_i \in DB$, if its NT-value x_i^N exists and QT-value x_i^Q does not exist, x_i is a trustworthy data. We use $\Re$ to represent the trustworthy data set.

$$\Re = \{x_i \mid x_i \in DB \wedge \exists x_i^N \in DB^N \wedge \neg \exists x_i^Q (x_i^Q \in DB^Q)\} \quad (1)$$

However, when suspicious user emerges, invalid data could be produced because of suspicious activities in SUQ-Phase. And when suspicious user is proved innocent or malicious, invalid data will be recovered to be trustworthy finally in IDR-Phase. We will give definitions of invalid data and discuss it in next sections.

B. User Operation Isolation Algorithm

Once a suspicious user is detected, it should be quarantined efficiently, and make sure that suspicious users will be able to continue their work in isolation instead of

being stopped arbitrarily, meanwhile the invalid data will not be accessed by trustworthy users to prevent damage spreading. To achieve this goal, data accessing operations of users should be controlled as shown in User Operation Isolation Algorithm (UOIA). Here, *read*(x_i^Q/x_i^N), *write*(x_i^Q/x_i^N) and *create*(x_i^Q/x_i^N) are the read, write and create operations on NT-value or QT-value of x.

Algorithm 1: UOIA Pseudo Code

Input: User s submit read or write operation on data item x_i
Output: Result of operation (TRUE or FALSE)
Steps:

```
1: IF s is trustworthy THEN
2:   IF ¬∃ x_i^Q(x_i^Q∈DB^Q) ∧ ∃ x_i^N(x_i^N∈DB^N) THEN
3:     execute read(x_i^N) or write(x_i^N);
4:     RETURN TRUE; // user operation succeed
5:   END IF
6: ELSE IF s is suspicious THEN
7:   IF s submit read on x_i THEN
8:     IF ∃ x_i^Q(x_i^Q∈DB^Q ∧ owner(x_i^Q) = s) THEN
9:       execute read(x_i^Q);
10:      RETURN TRUE;
11:    ELSE IF ¬∃ x_i^Q(x_i^Q∈DB^Q) ∧ ∃ x_i^N(x_i^N∈DB^N) THEN
12:      execute read(x_i^N);
13:      RETURN TRUE;
14:    END IF
15:  ELSE IF s submit write on x_i THEN
16:    IF ∃ x_i^Q(x_i^Q∈DB^Q ∧ owner(x_i^Q) = s) THEN
17:      execute write(x_i^Q);
18:      RETURN TRUE;
19:    ELSE IF ¬∃ x_i^Q(x_i^Q∈DB^Q) ∧ ∃ x_i^N(x_i^N∈DB^N) THEN
20:      execute create(x_i^Q); //create QT-value
21:      set owner(x_i^Q) = s; // set ownership of QT-value
21:      execute write (x_i^Q);
22:      RETURN TRUE;
23:    END IF
24:  END IF
25: END IF
26: RETURN FALSE; // user operation failed
```

We can see that in UOIA: a) When a trustworthy user s wants to read or write data item x_i, only if x_i^N exists and x_i^Q does not exist (which means that x_i is trustworthy), the read or write operation on x_i^N will be executed, otherwise it will fail. b) When suspicious user s wants to read x_i, if x^Q owned by s exists, x_i^Q will be returned to s, while if x_i^Q does not exist and x_i^N exists, x_i^N will be returned to s. c) When suspicious user s wants to write x_i, if x_i^Q owned by s exists, the write operation on x_i^Q will be executed, while if x_i^Q doesn't exist and x_i^N exists, x_i^Q will be created, and the write operation on x_i^Q will be executed in the end. Note that the algorithm is based on strict two-phase-locking (2PL) [24] concurrency control protocol with data item locking granularity.

Data accessing of suspicious users could cause trustworthy data to be invalid. For a trustworthy data x, if a suspicious user s submits write operation (UPDATE) on it, according to UOIA, the QT-value x_i^Q owned by s will be created, so x_i^N and x_i^Q will both exist. Due to the suspicious identity of s, x_i^Q is also suspicious. If s is indeed malicious, x_i^Q should be discarded since it is incorrect value of x_i and x_i^N will be identified as correct. Reversely, if s is indeed trustworthy, x_i^Q is actually the correct value of x_i reversely, while x_i^N is turned to be incorrect and should be replaced with x_i^Q. Obviously, the correct value of data item x_i is undetermined, either x_i^N or x_i^Q is correct. So accessing x_i^N or x_i^Q by trustworthy users could harm trustworthy data and cause damage spreading, leading this kind of data items invalid. Particularly, if s submits a new data creation operation (INSERT) successfully, a particular data item will be created with only QT-value existence, which renders situation similar to above: Suppose that data item x_i with only x_i^Q existence is created by s, if s is indeed trustworthy, x_i will be also trustworthy, but if s is indeed malicious, x_i should be non-existent, so this kind of data item is also uncertain and invalid. Therefore, to isolate invalid data like x_i from trustworthy users is essential to prevent damage spreading. Here we give the definition of invalid data as follows.

Definition 3. Invalid Data: For a data item $x_i \in DB$, if the QT-value x_i^Q exists, x_i is an invalid data. We denote invalid data set of database as $\Im$, and invalid data set caused by suspicious user s as $IDS(s)$.

$$\Im = \{x_i \mid x_i \in DB \wedge \exists x_i^Q (x_i^Q \in DB^Q)\} \quad (2)$$

$$IDS(s) = \{x_i \mid x_i \in DB \wedge \exists x_i^Q (x_i^Q \in DB^Q \wedge owner(x_i^Q) = s)\} \quad (3)$$

Since invalid data is caused by suspicious users, if we use $S=\{s_1, s_2, \ldots, s_m\}$ to represent all suspicious users, we can get an equation about invalid data set as follows.

$$\Im = \bigcup_{s_i \in S} IDS(s_i) \quad (4)$$

Lemma 1. $DB = \Re \cup \Im$

Following the definition of $\Im$ and $\Re$, it is easy to see that Lemma 1 is true.

Lemma 2. UOIA can ensure all invalid data of $\Im$ will be isolated from trustworthy users.

Proof: *(Sketch)* According to UOIA procedures, for each data item $x_i \in DB$, only if x_i^N exists and x_i^Q does not exist, which means that x_i is trustworthy, x_i can be read or written by trustworthy users. But if x_i^Q exists, the access to data item x_i by any trustworthy users will fail. So Lemma 2 holds.

As known from Lemma 2, all invalid data will be isolated from trustworthy users to prevent damage spreading, and QT-value of data items will be provided to continue the work of suspicious users in isolation. Therefore, work of SUQ-Phase can be accomplished by following UOIA.

C. On-the-fly Invalid Data Recovery Algorithm

Once a suspicious user is proved innocent or malicious, for each invalid data x_i caused by it, the correct value of x_i will be identified, and the IDR-Phase will start. The key points of the recovery measures are as follows: If s is proved innocent, the QT-value of invalid data owned by s is correct to be written back into the corresponding NT-value, if s is proved malicious, the QT-value of invalid data owned by s is incorrect and should be deleted. To implement the above measures, we give an On-the-fly Invalid Data Recovery Algorithm (OIDRA) as shown in Algorithm 2. Here,

delete(x_i^Q/x_i^N) represent delete operation on NT-value or QT-value of data item x_i.

In OIDRA, once a suspicious user *s* is proved innocent or trustworthy, the performing operations of *s* will be canceled and *s* will be stopped from data accessing to begin invalid data recovery. Then, if *s* is proved innocent, all QT-value of invalid data owned by *s* is correct, they will be adopted to replace the corresponding NT-value (if NT-value does not exist, it will be created firstly). After that the data accessing authority of *s* will be resumed. If *s* is proved malicious, all QT-value owned by *s* will be deleted. When above procedures finished, all QT-value owned by *s* will be dropped, and all invalid data of *IDS*(*s*) will be recovered to end the IDR-Phase of *s*. Therefore the work of IDR-Phase can be fulfilled by OIDRA.

Algorithm 2: OIDRA Pseudo Code

Input: a signal that *s* is proved innocent or malicious
Output: Recovery of invalid data
Steps:
1: Cancel all performing operations of *s*, and stop *s* from accessing database;
2: **IF** *s* is proved innocent **THEN**
3: **FOR EACH** $x_i^Q \in DB^Q \wedge owner(x_i^Q) = s$
4: **IF** $\neg\exists\, x_i^N (x_i^N \in DB^N)$ **THEN**
5: execute *create*(x_i^N)**;**
6: **END IF**
7: set $x_i^N = x_i^Q$;
8: execute *delete*(x_i^Q);
9: **END FOR**
10: Resume data accessing authority for *s*;
11: **ELSE IF** *s* is proved malicious **THEN**
12: **FOR EACH** $x_i^Q \in DB^Q \wedge owner(x_i^Q) = s$
13: execute *delete*(x_i^Q);
14: **END FOR**
15: **END IF**

Furthermore, since all invalid data is isolated from trustworthy users as known from Lemma 2, OIDRA can performs concurrently with other operations, so OIDRA is an on-the-fly algorithm.

Lemma 3. OIDRA can ensure that all invalid data of $\Im$ will be recovered in the IDR-Phase of suspicious users.

Proof: *(Sketch)* According to OIDRA procedures, once a suspicious user *s* is proved innocent or malicious, all invalid data of *IDS*(*s*) caused by *s* will be recovered. Since every suspicious user will be proved innocent or malicious finally, each invalid data caused by suspicious users will be recovered in the end. While all invalid data caused by all suspicious users is just the invalid data set $\Im$ as known from definition 2, so Lemma 3 can be satisfied.

Lemma 4. SQRM can satisfy NLID criterion.

Proof: *(Sketch)* According to Lemma 2 and Lemma 3, we can see that all invalid data caused by suspicious users will be isolated from trustworthy users, so damage spreading will be prevented, and all invalid data will be recovered in the IDR-Phase of suspicious users. As a result, the NLID criterion can be satisfied in SQRM.

Therefore, the suspect quarantine and recovery method we proposed in this paper is an effective security mechanism, which can make sure that further damage by damage spreading will be confined and all invalid data will be recovered.

IV. CONCLUSIONS

In this paper, we presented an effective security mechanism, namely suspect quarantine and recovery method denoted as SQRM, to increase the attack resistance of database vulnerable to suspicious users. We develop a suspicious user quarantine scheme in SQRM that isolates invalid data from trustworthy users to protect databases from any further damage caused by damage spreading, and provides the ability of working continually to suspicious users instead of stopping them arbitrary. At the same time, we propose an on-the-fly invalid data recovery method to repair all the invalid data caused by suspicious users when they are proved innocent or malicious.

There are some future works for SQRM. First, since transactions, which consist of access operations, will be suspended or aborted even if only one invalid data accessed, we will investigate the effect of SQRM on transaction success rate. Second, the availability of database when dealing with the quarantine of suspicious users will be studied. Furthermore, we will concentrate on construction of survival database system which is able to solve the problem of suspicious users by SQRM.

ACKNOWLEDGMENT

Our work is supported by the National Natural Science Foundation of China (60673127), the National 863 High Technology Research and Development Program of China (2007AA01Z404) and the Jiangsu Province Science & Technology Pillar Program (BE2008135).

REFERENCES

[1] E. Bertino and R. Sandhu, "Database Security-Concepts, Approaches, and Challenges," IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 1, pp. 2-19, Jan.-Mar. 2005, doi:10.1109/TDSC.2005.9.

[2] P. Ammann, S. Jajodia, and C. D. McCollum, "Surviving information warfare attacks on databases," Proc. 1997 IEEE Symp. Security and Privacy, IEEE Press, May. 1997, pp. 164-174, doi:10.1109/SECPRI.1997.601331.

[3] P. Liu, "Architectures for Intrusion Tolerant Database Systems," Proc. 18th Annual Computer Security Applications Conference, IEEE Press, Dec. 2002, pp. 311-320, doi:10.1109/CSAC. 2002.1176303.

[4] T. Chiueh and D. Pilania, "Design, implementation, and evaluation of a repairable database management system," Proc. 21th Int. Conference on Data Engineering, IEEE Press, Apr. 2005, pp. 1024–1035, doi:10.1109/ICDE.2005.49.

[5] P. Liu and J.W. Jing, "The Design and Implementation of a Self-Healing Database System," Journal of Intelligent Information Systems, vol. 23, Nov. 2004, pp. 247-269, doi:10.1023/B:JIIS.0000047394.02444.8d.

[6] Y. Hu and B. Panda, "A data mining approach for database intrusion detection," Proc. 2004 ACM Symp. Applied Computing, ACM Press, Mar. 2004, pp. 711-716, doi:10.1145/967900.968048.

[7] M. Vieira and H. Madeira, "Detection of Malicious Transactions in DBMS," Proc. 11th IEEE Int. Symp. Pacific Rim Dependable Computing, IEEE Press, Dec. 2005, pp. 350-357, doi:10.1109/PRDC.2005.31.

[8] J. Fonseca, M. Vieira, and H. Madeira, "Integrated Intrusion Detection in Databases," Proc. 3rd Latin-American Symp. Dependable

 ISBN:978-1-61208-115-1

Computing, Springer-Verlag Press, Sep. 2007, pp. 198-211, doi: 10.1007/978-3-540-75294-3_15.

[9] A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," The VLDB Journal, vol. 17, Aug. 2008, pp. 1063-1077, doi:10.1007/s00778-007-0051-4.

[10] P. Liu and S. Jajodia, "Multi-phase Damage Confinement in Database Systems for Intrusion Tolerance," Proc. 14th IEEE Computer Security Foundations Workshop, IEEE Press, Jun. 2001, pp. 191-205, doi:10.1109/CSFW.2001.930146.

[11] K. Bai and P. Liu, "A light weighted damage tracking quarantine and recovery scheme for mission-critical database systems," Proc. 17th ACM Conference on Information and knowledge management, ACM Press, Oct. 2008, pp. 1403-1404, doi:10.1145/1458082.1458302.

[12] K. Bai, M. Yu, and P. Liu, "TRACE: Zero-Down-Time Database Damage Tracking, Quarantine, and Cleansing with Negligible Run-Time Overhead," Proc 13th European Symp. Research in Computer Security, Springer-Verlag Press, Oct. 2008, pp. 161-176, doi:10.1007/978-3-540-88313-5_11.

[13] M. Yu, W. Zang, and P. Liu, "Database Isolation and Filtering against Data Corruption Attacks," Proc 23rd Annual Computer Security Application Conference, IEEE Press, Dec. 2007, pp. 97-106, doi:10.1109/ACSAC.2007.18.

[14] K. Bai and P. Liu, "A data damage tracking quarantine and recovery (DTQR) scheme for mission-critical database systems," Proc. 12th Int. Conference Extending Database Technology, ACM Press, Mar. 2009, pp. 720-731, doi:10.1145/1516360.1516443.

[15] P. Liu, P. Ammann and S. Jajodia, "Rewriting Histories: Recovering From Malicious Transactions," Distributed and Parallel Databases, vol. 8, Jan. 2000, pp. 7-40, doi:10.1023/A:1008731200105.

[16] P. Ammann, S. Jajodia, and P. Liu, "Recovery from Malicious Transactions," IEEE Transactions Knowledge and Data Engineering, vol. 14, Sep. 2002, pp. 1167-1185, doi:10.1109/TKDE.2002.1033782.

[17] T. Chiueh and S. Bajpai, "Accurate and efficient inter-transaction dependency tracking," Proc. 24th Int. Conf. Data Engineering, IEEE Press, Apr. 2008, pp. 1209-1218, doi: 10.1109/ICDE.2008. 4497530.

[18] D. Lomet, Z. Vagena, and R. Barga, "Recovery from "bad" user transactions," Proc. 2006 ACM SIGMOD Int. Conference Management of Data, ACM Press, Jun. 2006, pp. 337–346, doi:10.1145/1142473.1142512.

[19] R. Yalamanchili and B. Panda, "Transaction Fusion: A Model for Data Recovery from Information Attacks," Journal of Intelligent Information Systems, vol. 23, Nov. 2004, pp. 225-245, doi:10.1023/B:JIIS.0000047393.99078.c4.

[20] K. Bai and P. Liu, "Towards Database Firewall: Mining the Damage Spreading Patterns," Proc. 22nd Annual Computer Security Applications Conference IEEE Press, Dec. 2006, pp. 449-462, doi:10.1109/ACSAC.2006.52.

[21] S. Jajodia, P. Liu, and C. D. McCollum, "Application-Level Isolation to Cope With Malicious Database Users," Proc. 14th Annual Computer Security Applications Conference, IEEE Press, Dec. 1998, pp. 73-82, doi:10.1109/CSAC.1998.738580.

[22] P. Liu, "DAIS: A Real-Time Data Attack Isolation System for Commercial Database Applications," Proc. 17th Annual Computer Security Applications Conference, IEEE Press, Dec. 2001, pp. 219-229, doi:10.1109/ACSAC.2001.991538.

[23] P. Liu, H. Wang, and L.Q. Li, "Real-time data attack isolation for commercial database applications," Journal of Network and Computer Applications, vol 29, Nov. 2006, pp. 294-320, doi:10.1016/j.jnca. 2005.03.001.

[24] P. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987.

 ISBN:978-1-61208-115-1