Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 914305, 8 pages
http://dx.doi.org/10.1155/2015/914305

Research Article

Tracking and Repairing Damaged Healthcare Databases Using the Matrix

Sanaa Kaddoura,[1] Ramzi A. Haraty,[2] Ahmed Zekri,[1] and Mehedi Masud[3]

[1] *Department of Mathematics and Computer Science, Beirut Arab University, Beirut, Lebanon*
[2] *Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon*
[3] *Computer Science Department, Taif University, Taif, Saudi Arabia*

Correspondence should be addressed to Ramzi A. Haraty; rharaty@lau.edu.lb

Received 18 June 2015; Accepted 10 August 2015

Academic Editor: Atsuo Inomata

In a distributed mobile e-health care system, e-health service providers exchange data on the fly in response to user queries without any centralized control. Local databases in e-health service providers might be intercepted during the exchange of data and read by intruders; and malicious transactions may damage data that is highly confidential. In this case any centralized control for securing data cannot be assumed to protect confidential data. Therefore, securing health information from malicious attacks has become a major concern. Although prevention techniques are available, the history of system break-ins guarantees that there is no foolproof technique that totally eliminates security loopholes in a computer system. Hence, efficient damage assessment and recovery techniques are needed. Traditional methods require scanning the entire log from the point of attack to the end which is a slow procedure. In this paper, we present an efficient damage assessment and recovery algorithm to recover the database from malicious transactions. The algorithm is based on data dependency and uses a single matrix. The results of this work prove that our algorithm performs better than the other algorithms in both the damage assessment and the recovery stages.

1. Introduction

Information warfare is one of the most effective weapons used in today's wars. It had many definitions throughout history. In 1980, information warfare was used as a military term [1]. Then, it grew up to affect our life more during the Gulf War in 1991 [1]. Libicki and Fellow [2] compared the definition of information warfare to the process of discovering the nature of an elephant by a blind man. If the blind man touched the elephant's leg, he will think that it is a tree, while if he touched the tail, he will think that it is a rope. The aim of this comparison is to say that information warfare has different meanings and dimensions. It can be defined according to the dimension that the person tackles.

In this research, information warfare is defined as the set of mechanisms used to gain access to private information using information warfare weapons such as computer viruses, Trojan horses, logic bombs, trap doors, and denial of service [3].

Securing data has become a crucial issue during the web era because business applications are based on intensive sharing of information through the internet. Defensive information warfare passes through three phases in order to secure data: prevention, detection, and recovery. The prevention phase tries to prevent an attack on the database. However, there is always a chance for a successful attack on the computer system. As a legitimate transaction reads damaged data to update valid data, damage will spread in the database. When prevention fails, the intrusion detection phase starts identifying the attack. Although there are numerous researches in the intrusion detection area [4], existing detection techniques cannot guarantee that the malicious transaction will be detected immediately. Hence, until the attack is detected, part of the database will be affected. Thus, recovery will be needed.

Recovery is one of the main phases in defensive information warfare because it ensures software tolerance. Upon transaction processing, it reads and manipulates data items which in turn might be read and updated by other transactions. This leads to interdependency between the transactions. Hence, when a malicious transaction is detected by the intrusion detection system, many transactions might be affected after reading malicious data. All the affected transactions must be rolled back to recover the database to its consistent state. There are two issues to be considered in order to have efficient recovery of the database. First, the recovery process must be carried out in the shortest possible time to minimize denial of service. Second, the accuracy of the algorithm is a crucial issue in order to roll back only affected transactions or affected data items. The benign part of the database should stay available to the running applications.

In this paper, we propose new efficient damage assessment and recovery algorithms that use a single matrix to store the dependency between data items in the log file. When an intrusion detection system detects a malicious transaction, the matrix will be used to identify the damaged data items for later recovery. The algorithms do not need to scan the entire log file in order to discover the affected part of the database. This will cut down on damage assessment time. Using only one data structure, the matrix, saves space and computational time because there is no need for graphs or logical operations as other approaches do. All of this increases the efficiency of our approach over previously proposed ones.

The remainder of the paper is organized as follows. Section 2 covers the literature review. Section 3 presents the proposed model. Section 4 discusses the experimental results, and Section 5 concludes the paper.

2. Literature Review

Damage assessment and recovery algorithms have been of great interest to researchers in the last two decades. There is always an intrusion detection system that detects the malicious transactions. Then the recovery process starts, where all the transactions from the malicious one to the end of the log file should be assessed and classified into benign or affected transaction. There are two approaches to do so: data dependency and transaction dependency. Using the data dependency approach, affected data items will be checked if they were updated by an operation of a malicious transaction, while using transaction dependency approach, a transaction will be considered as affected if it reads a value that was updated by a malicious transaction.

One of the previous approaches was the user isolation approach [5]. This approach assumes that all the users are malicious users until a certain period of time passes. Then, the actions of the user will be identified as either malicious and will be aborted or nonmalicious and will be committed.

Clustering the log file is also one of the suggested approaches. In [6], the authors proposed segmenting the log files into clusters. However, the size of the cluster was a weakness because two different clusters may contain two dependent transactions. This issue was solved in [7]. The authors proposed segmenting the log file into clusters and further subclusters. The clusters were subclustered according to two different approaches: number of data items or space occupied.

In [8], the authors suggested an agent-based approach, where each agent is responsible for a set of data items. The agents keep the version of the data item to help identify whether it is affected or not.

Other approaches suggested using matrices in recovery. In [9, 10], the authors used a matrix that saves transaction dependencies. After each committed transaction, a new row is added to the matrix. The cells of the matrix hold 0 value in case there is no dependency, 1 in case of dependency, a transaction ID in case there is a transaction that reads a data item that was last modified by another transaction, and finally a negative transaction ID if a data item was modified by a set of transactions. In the last case a complementary array was needed to save the list of transactions. The static allocation of the matrix will cause a problem if the transactions and data items grow in size. The difference between our work and the work presented in [9, 10] is the fact that we will use only one matrix without any complementary structures which will save memory and execution time. In addition, we will use data dependency instead of transaction dependency, which locates more precisely the affected data items.

The authors of [11] suggested using column dependency. The attributes of the database are referred to as columns. The transaction takes into consideration the columns that were read or modified by any operation of the transaction. This approach decreased recovery time and showed that as the number of malicious transactions increases, the number of inconsistencies after reexecution will increase too.

Damage assessment and recovery for distributed databases were suggested in [12]. The proposed approach keeps a log file at each site, where there are two processes that are responsible for damage assessment and repair: a local DAR manager and a local DAR executor. Scanning the log file is the responsibility of the local DAR executor that will identify affected subtransactions and clean them. The damage assessment and recovery algorithms work in parallel, which in turn reduce execution time.

In [13], the authors suggested a recovery approach for real time database systems based on transaction fusion. Upon recovery, the malicious and the valuable affected transactions are fused into new transactions based on their dependencies and time limit. Transaction fusion reduced the number of transactions which in turn reduced recovery time.

The Retro system was suggested in [14]. It is a self-healing system based on transactional dependency and repairs the database from intrusion at the operating system level. The model performs selective recovery by undoing only suspect transactions in order to reduce the effect of those transactions as much as possible.

In [15], the authors propose a damage assessment model that deals with new intertransaction dependencies: phantom dependency, pseudoidentity dependency, domain integrity dependency, and reference integrity dependency. They describe their robust damage assessment approach and show that in some cases the recovery is incomplete unless the new proposed dependencies are considered. In [16], the ITDM prototype model for damage assessment implements the damage assessment model described in [15] using SQL rewriting. The SQL statements that are sent to the DBMS were modified in order to obtain the read/write dependencies. They showed that exploiting the suggested intertransaction dependencies of [15] leads to more consistency in the database.

The Before Images approach was suggested in [17]. The Before Image table is a table that is similar to the stored table in the database but without constraints. These tables are inaccessible by the users. This approach keeps track of the deleted data items. When a data item is deleted or updated in the initial table, a copy of the old record will be added to the Before Image table that refers to the initial table. To control the size of the Before Image table, a time window was suggested for the deleted data items. When these items are out of date, they will be deleted from the Before Image table. The performance of the algorithm shows degradation due to the queries performed on the Before Image table.

The authors of [18] propose a new approach that is based on transaction reverse dependency log file. The transaction reverse dependency log file is used to generate the Undo Transaction Set (UTS). The authors present a recursive algorithm for generating the UTS and another algorithm that uses a stack. Then, they showed that the stack performs better than the recursive one because recursive functions cause a lot of overhead whenever the transaction number increases.

In [19], the authors propose that each application needs its own type of recovery based on its nature. The authors suggested a recovery algorithm for a banking system by exploiting the deposit/withdrawal transactions. They reduced recovery time by incorporating the semantics of the transactions. The limitation of this approach is the fact that it cannot be applied to any database system since it is not generic.

Fuzzy dependencies were suggested in [20]. The suggested approach consists of "Fuzzy Value Generator" that interacts with the database and the "fuzzy dependency storage". The advantage of this approach is the fact that it does not require intensive search for transactional dependencies in the log file. This boosts the recovery process. However, the disadvantage of this approach is the lack of accuracy.

3. The Proposed Model

The proposed model is based on data dependency and uses a matrix to store the dependencies between data items. Data dependency is considered to be more accurate than transaction dependency. The data dependency approach identifies the affected data items by each operation of the transactions and only deals with these data items upon recovery. However, transaction dependency rolls back all the affected transactions regardless of the operations that compose the transaction. The proposed algorithm uses a single matrix for identifying the affected data items during the damage assessment process, which saves memory and reduces recovery time.

3.1. Assumptions. The proposed model is based on the following assumptions. First, the algorithm will receive the set of malicious transactions from an intrusion detection system. Second, the history generated by the database management systems is rigorously serializable as serializability theory provides correctness. Third, there is a sequential log file that saves all the read/write operations of the committed transactions. This log file is inaccessible by users and will be used upon recovery. Fourth, the transactions have sequential IDs that are incremented on the arrival of new transaction. This means that when transaction T_2 commits, then the only transaction that has committed before T_2 is T_1. Fifth, the order of the operations is the same as the history. Finally, we assume that the transaction log stores all the operations of the committed transactions and stores for each write operation the value of the data item before being updated.

3.2. Definitions

Definition 1. A write operation $w_i[x]$ of a transaction T_i is dependent on a read $r_i[y]$ operation of T_i, if $w_i[x]$ is computed using the value obtained from $r_i[y]$ [21].

Definition 2. A blind write is when a transaction T_i writes data item x without reading the previous values of x [22].

Definition 3. A write operation $w_i[x]$ of a transaction is dependent on a set of data items I, if $x = f(I)$; that is, the values of data items in I are used in calculating the new value of x. If $x \notin I$, the operation is called a blind write. In this case, if the previous value of x (before this write operation) is damaged and none of the data items in I are damaged, then the value of x will be refreshed after this write operation [21].

Definition 4. A data value v_1 is dependent on data value v_2, if the write operation that wrote v_1 was dependent on a read operation on v_2. Note that v_1 and v_2 may be two different versions of the same data item [23].

Definition 5. A transaction management mechanism guarantees rigorousness if it guarantees strictness, and no data item is written until the transaction that previously read it either commits or aborts [14].

Definition 6. A transaction T_j is said to be dependent upon another transaction T_i, if there exists a data item x such that T_i is the last committed transaction to update x before T_j reads x. The dependency relationship is denoted by $T_i \rightarrow T_j$. Since the schedule is assumed to be strictly serializable, there will not be any active transaction writing x between T_i updating x and T_j reading x [24].

Definition 7. A write operation is called a valid write if the value is written by a benign transaction and is independent of any contaminated data [25].

3.3. The Damage Assessment Algorithm. The proposed damage assessment algorithm is based on data dependency. The idea of this approach is to connect the data items together to show the dependency between them. Then, the algorithm will detect the directly affected data item(s) by the malicious transaction. After that, the algorithm will be able to find any data items that are reachable by the malicious data items. Those data items are called affected data items. Data items that are not reachable by affected data will be considered clean and will be available to the user upon the recovery process.

A two-dimensional matrix will keep track of the dependencies between the data items. This matrix will directly point out the affected data items without any need to scan the whole matrix or the log file. Only one matrix will be used to represent the dependencies. No other data structures or logical operations are needed in the algorithm.

Initially, the matrix is empty. One of the transaction's operations is commot., a row will be added to the matrix to represent this transaction and the data that this transaction used to write the new values of the data items are added as columns. Although data dependencies are the key of the algorithm, saving the transactions will help accurately identify the affected data items. The columns in the matrix represent the data items used in the operation to write a new value. Each cell in the matrix holds the data item that was written by a given transaction.

Another feature that increases the efficiency of the proposed algorithm is adding the first column to the matrix, which is BW column or blind write column. It does not correspond to a specific data item; instead, it will be used if the transaction blindly wrote an item without reading any other data item. The importance of this column will be revealed if an affected data item was blindly written after being affected and then it will become clean. This helps in reducing the number of affected data items which in turn reduces recovery time. Table 1 shows the matrix that will be generated after the commitment of the transactions as follows.

Sample Transactions Set. Sample transactions set is as follows:

T_1: C := D.

T_2: D := D + 2.

T_3: A := B + 1.

T_4: B := C.

T_5: E := C + 3.

T_6: E := 3.

T_7: X := E + 5.

T_8: D := E + B.

T_9: Y := B.

In [26], the authors used multiple matrices for damage assessment in order to discover dependencies which require logical operations between the matrices. However, in our approach, the single matrix reduces the damage assessment time.

Assume that T_1 is the malicious transaction. T_1 writes the data item C. Thus, C is responsible for spreading the damage. We will go directly to the column labeled C in the matrix to see the data items that are reachable by C. In this column, B and E are affected data items. Thus, B and E will spread the damage more. However, C was used in writing B and E in transactions T_4 and T_5. Since T_4 commits after T_1, then, the activity at T_4 will be checked. There is no need to look at any transaction that precedes T_4 since damage spreading started at T_4. Data item A was written after reading B in T_3 but it will not be considered affected because T_3 committed before T_4. When T_3 committed, data item B was clean.

Table 1: The dependency matrix.

	Data items used in the transactions				
	BW	D	B	C	E
T_1		C			
T_2		D			
T_3			A		
T_4				B	
T_5				E	
T_6	E				
T_7					X
T_8			D		D
T_9			Y		

Considering the data item B, we will move to the transactions T_i, where $i > 4$, in column B to detect the data items that were affected by B. In this case, we have D and Y. D and Y will be added to the affected data items set. Considering the data item E, we will move on to the transactions T_i, where $i > 5$, in column E to detect the data items that were affected by E. The result should be X and D. However, we have an entry for E in the first column of BW; thus, E was blindly written before being used in writing X and D. Therefore, E became clean before being used in transaction T_7. Thus, X and D are clean because they were written after refreshing the value of E. In this example, we ended up having C, B, D, and Y as affected data items that should be recovered. The damage assessment algorithm is summarized in Algorithm 1. Assume that the dependency matrix is called M and there is n-transactions. The BW column has an index 0.

3.4. The Recovery Algorithm. During recovery, any executing or new operation of a transaction should be prevented from accessing malicious and affected data items. Only affected operations of the transactions will be reexecuted. The other part of the database will be available. The operations of malicious transactions will be undone. The recovery algorithm is summarized in Algorithm 2.

3.5. Check Points. The matrix may grow and the data inside it may become obsolete. Thus, to save memory, the above algorithm requires a checkpoint to get rid of the matrix at a specific time interval after which we suspect that the data is clean and the malicious transactions were detected by the intrusion detection system. This time interval should not be too short in order not to need to go back to previous check points and reread the log file. Additionally, it should not be too long so that the size of the matrix and the log file can be controlled. However, the intrusion detection system may detect a malicious transaction after the check point. In this case, the dependency matrix has to be reconstructed.

```
Receive a set of malicious transactions $S$
While there is unprocessed transaction in $S$
    Select the minimum unprocessed transaction id $T_i$ in $S$
    Identify the data items set D written by $T_i$
    Add D to the affected set
    Associate with each data item the affecting transaction index $i$
    For each unprocessed data item in the affected set
        Find the index $k$ for the item
        For $j = m + 1$ to $n$ // $m$ is the transaction ID that affected $k$
            If $M[j][0] = k$ // the affected item is blindly written
                Remove $k$ from the affected
                Move to another data item in the affected set
            Else If $M[j][k]$ is not Null // Null means empty cell
                    Add $M[j][k]$ to the affected set
```

Algorithm 1: The damage assessment algorithm.

```
Receive the set of malicious and affected data items
For each affected data item
    Retrieve the operation from the log file
    Update the value by the old value before the operation
    Update the database
For each malicious data item
    Retrieve the operation from the log file
    Delete the operation
```

Algorithm 2: The recovery algorithm.

It will be time consuming if we reread the log file and reconstruct the matrix. To solve this issue, we will keep a compressed structure of the dependency matrix that will help in reconstructing the matrix without going back to the log file.

Since the matrix is a sparse matrix, the condensed storage technique that we used is the Condensed Row Storage (CRS) [26]. The CRS format makes no assumption about the sparsity structure of the matrix and does not store any unnecessary element of the matrix. Assuming we have an $M \times N$ sparse matrix $A = [aij]$, containing nonzero (NZ) elements, the CRS format is constructed as follows:

(i) One-dimensional vector AN holds all the nonzero values of matrix A.

(ii) One-dimensional vector AJ which has an equal length to AN holds the column number of each element (starting from 1).

(iii) One-dimensional vector AI stores the locations in the AN vector that start a row.

Thus, the CRS of the matrix in Table 1 will be as follows:

$$AN = [\text{C} \;\; \text{D} \;\; \text{A} \;\; \text{B} \;\; \text{E} \;\; \text{E} \;\; \text{X} \;\; \text{D} \;\; \text{D} \;\; \text{Y}].$$

$$AJ = [2 \;\; 2 \;\; 3 \;\; 4 \;\; 4 \;\; 1 \;\; 5 \;\; 3 \;\; 5 \;\; 3].$$

$$AI = [1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6 \;\; 7 \;\; 8 \;\; 10].$$

These vectors will only be used in case we needed to reconstruct the matrix after a check point. Since we assume that the log file is rigorous and serializable, we will only build the matrix starting from the malicious transaction. The transactions that precede the malicious one do not need to be recovered. The vectors will be refreshed at each new check point to hold the values of the new matrix.

The CRS will only be built for one check point backward. If in rare cases the intrusion detection system detected an attacker before the check point of the CRS (worst case scenario), then the log file will be used and the dependency matrix will be rebuilt.

3.6. Example. Consider a database for health care management system. We will only consider the process of keeping patients records in the health care system. It contains information about the following:

(i) Doctor (DrID, DrName, DrSpecialization).

(ii) Patient (PID, PName, PGender).

(iii) Disease (DID, DName).

(iv) PatientRecord (PID, DrID, DID).

(v) PatientBillItems (PBID, PID, Nitems, cost).

(vi) PatientBill (BID, PID, Nitems $*$ cost).

Consider the following insert transactions in the database:

(i) T_1 = Doctor (“1”, “Mike”, “Allergist”).

(ii) T_2 = Patient (“5”, “Hana”, “F”).

(iii) T_3 = Disease (“11”, “eye allergy”).

(iv) T_4 = PatientRecord (“5”, “1”, 11).

(v) T_5 = PatientBillItems (3, 5, 6, 150).

(vi) T_6 = PatientBill (2, 5, 900).

The dependency matrix M will hold the dependencies of the above committed transactions, as shown in Table 2. The transactions T_1 to T_3 do not depend on any other transactions. They are insert transactions; hence, they are blindly written and will be stored in the first column of the matrix BW. T_4 and T_5 are insert transactions but since

Table 2: Sample matrix.

	BW	5	1	11	6	150
T_1	1					
T_1	Mike					
T_1	Allergist					
T_2	5					
T_2	Hana					
T_2	F					
T_3	11					
T_3	Eye allergy					
T_4		5	1	11		
T_5	3	5				
T_5	6					
T_5	150					
T_6	2					
T_6		5			900	900

they read values from other transactions and put them in another table the data items will be considered dependent. T_4 reads from T_1, T_2, and T_3. T_5 reads from T_2. T_6 is an insert transaction that reads from T_5 and writes a new value.

Consider the case where damage assessment algorithm receives from the intrusion detection system that T_5 is a malicious transaction. T_5 writes 3, 5, 6, and 150. After that T_6 reads 5, 6, and 150 to write 5 and 900. Then, from the columns of the matrix, the data items 5 and 900 are affected and will be added to the affected data set. The operation of Nitems $*$ cost in T_6 should be rolled back and reexecuted. T_5 operations must be deleted.

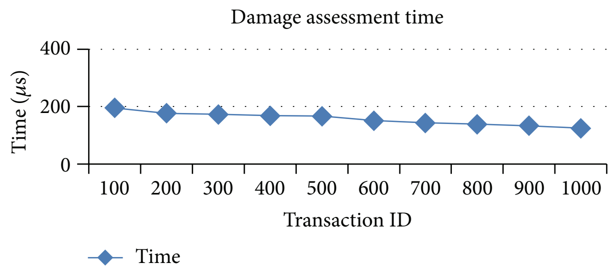

Figure 1: Performance time of damage assessment algorithm.

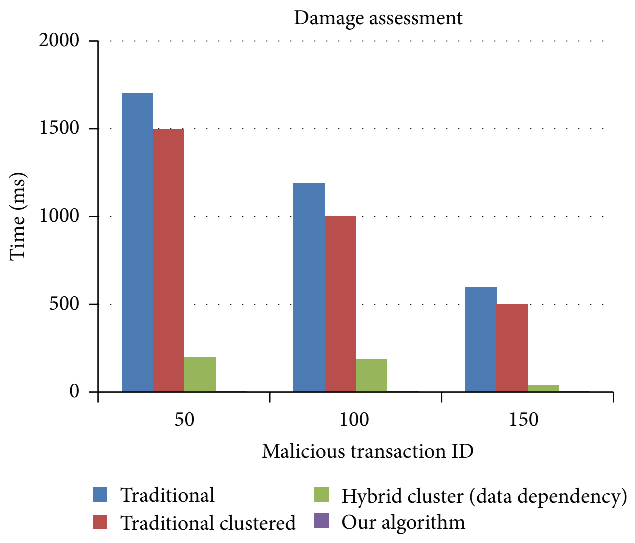

Figure 2: Comparison of different damage assessment algorithms.

4. Experimental Results

In order to test the performance of the algorithm, we created a simulated system. This system creates random transactions. When a transaction is committed, the log file will be updated. The algorithm requires a dynamic dependency matrix that is also updated after each committed transaction. The matrix will be available during the damage assessment procedure. The log file is required to be available during the recovery process. The system generates random transactions and accordingly creates the log file. The system is connected to SQL server 2012. The Northwind database was used during the experiments [27]. The simulation system is implemented using Java programming language (JDK 1.6) on NetBeans IDE 6.8. The computer system used is as follows: the processor is Intel Core i5 CPU 2410 M at 2.3 GHz with 6 GB RAM.

4.1. The Damage Assessment Algorithm Performance Analysis. Figure 1 shows the time needed by the damage assessment algorithm to discover the affected data items. x-axis shows the transaction ID of the malicious transaction and y-axis shows the time in microseconds. As shown in the figure, as the transaction ID increases, the time required for damage assessment decreases. This is one of the advantages of the rigorous serializability of the log file. As the transaction ID increases, there is no need to go back to the log file and check the previous transactions. The log file contains 1080 records and each record refers to a transaction. When the malicious transaction ID is 500, the algorithm has to traverse only 580 transactions, whereas when the malicious transaction ID is 1000, the algorithm has to traverse 80 transactions only. This decreases damage assessment time and in turn reduces the system denial of service time.

We compared our algorithm to three other ones: traditional algorithm, traditional clustered algorithm, and hybrid cluster algorithm. All the algorithms were tested in the same simulation system that contains 200 transactions. Figure 2 shows this comparison. As shown in the figure, our algorithm performs faster than the hybrid cluster one by 2218 times. This comparison shows that our algorithm performs the best among the four comparisons. This is due to the fact that our algorithm is based on matrices which allow the algorithm m to go directly to a specific index. In addition, using only one data structure saved time because no logical operators are needed. In addition, our algorithm does not need to refer to the log file during the damage assessment stage.

4.2. The Recovery Algorithm Performance Analysis. The recovery process starts when the damage assessment process stops. The output of the damage assessment stage is a set of

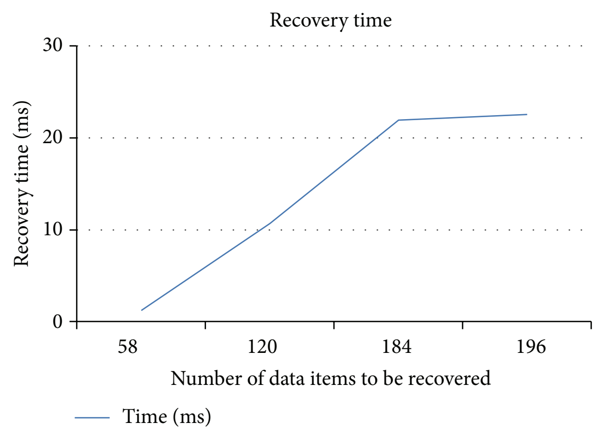

FIGURE 3: Performance of recovery algorithm.

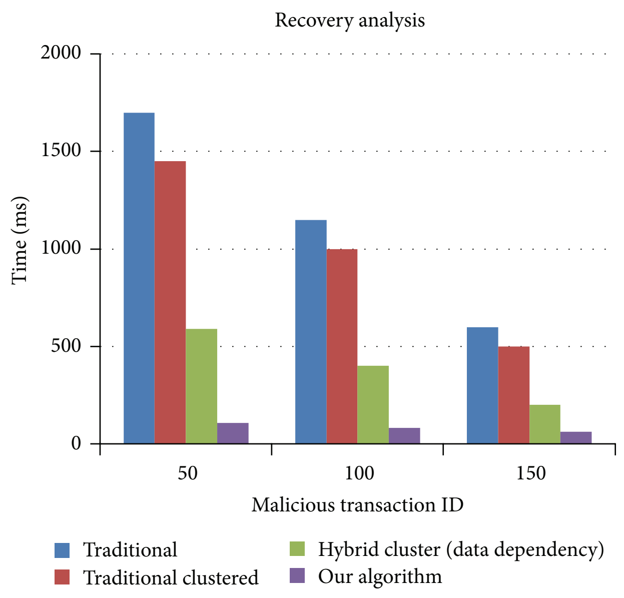

FIGURE 4: Comparison of recovery performance in different approaches.

affected data items to be recovered. Figure 3 shows the time needed to recover the affected data items. As revealed in the chart, as the number of data items increases, the recovery time increases. The database consisted of 500 transactions. Figure 4 shows the performance of different approaches in the recovery stage. The chart shows the efficiency of our algorithm. Our algorithm will refer to the log file records using the index of the transaction. Indexing helps a lot in saving the recovery time.

5. Conclusion

In this paper, we presented a new efficient approach of damage assessment and recovery in databases. Our approach is based on data dependency because it is more accurate than transaction dependency. The dependencies between data items are stored in a dynamic matrix that will grow as new transactions commit and new data items are updated. Using only one data structure in the damage assessment process saves memory and processing time. Being able to directly access the affected data items that are all present in one column reduces processing overhead. All the aforementioned properties of the approach make it efficient and reduce recovery time which reduces denial of service of the database. The algorithms were implemented and compared to other approaches.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was partially funded by the Lebanese American University, Beirut, Lebanon.

References

[1] W. Hutchinson, "Information warfare and deception," *Informing Science*, vol. 9, pp. 213–223, 2006.

[2] M. Libicki and S. Fellow, *What is Information Warfare*, United States Government Printing, 1995.

[3] R. Haeni, *Information Warfare an Introduction*, The George Washington University, Washington, DC, USA, 1997.

[4] T. F. Lunt, "A survey of intrusion detection techniques," *Computers & Security*, vol. 12, no. 4, pp. 405–418, 1993.

[5] H. Dai, X. Qin, G. Zheng, and Z. Li, "SQRM: an effective solution to suspicious users in database," in *Proceedings of the 3rd International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA '11)*, St. Maarten, The Netherlands Antilles, January 2011.

[6] R. A. Haraty and A. Zeitunlian, "Damage assessment and recovery from malicious transactions using data dependency," *ISESCO Journal of Science and Technology*, vol. 3, no. 4, 2007.

[7] R. A. Haraty and H. Mohsen, "Efficient damage assessment and recovery using fast mapping," in *Proceedings of the International Conference of Advanced Computer Science and Information Technology (ACSIT '14)*, June 2014.

[8] K. Kurra, B. Panda, W. Li, and Y. Hu, "An agent based approach to perform damage assessment and recovery efficiently after a cyber attack to ensure E-government database security," in *Proceedings of the 48th Hawaii International Conference on System Sciences (HICSS '15)*, pp. 2272–2279, January 2015.

[9] R. A. Haraty and M. Zbib, "A matrix-based damage assessment and recovery algorithm," in *Proceedings of the 14th International Conference on Innovations for Community Services (I4CS '14)*, pp. 22–27, June 2014.

[10] R. A. Haraty, M. Zbib, and M. Masud, "Data damage assessment and recovery algorithm from malicious attacks in healthcare data sharing systems," *Peer-to-Peer Networking and Applications*, 2015.

[11] A. Chakraborty, A. K. Majumdar, and S. Sural, "A column dependency-based approach for static and dynamic recovery of databases from malicious transactions," *International Journal of Information Security*, vol. 9, no. 1, pp. 51–67, 2010.

[12] P. Liu and M. Yu, "Damage assessment and repair in attack resilient distributed database systems," *Computer Standards and Interfaces*, vol. 33, no. 1, pp. 96–107, 2011.

[13] C. Chen, Q. Liu, Y. Liu, and G. Shen, "A recovery approach for real-time database based on transaction fusion," *Lecture Notes in Electrical Engineering*, vol. 124, no. 1, pp. 473–479, 2012.

[14] T. Kim, X. Wang, N. Zeldovich, and M. Kaashoek, "Intrusion recovery using selective re-execution," in *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10)*, 2010.

[15] G. Fu, H. Zhu, and Y. Li, "A robust damage assessment model for corrupted database systems," in *Information Systems Security: 5th International Conference, ICISS 2009 Kolkata, India, December 14–18, 2009 Proceedings*, vol. 5905 of *Lecture Notes in Computer Science*, pp. 237–251, Springer, Berlin, Germany, 2009.

[16] G. Fu, H. Zhang, and X. Liu, "Efficient damage propagation detection for compromised database systems," *Journal of Network & Information Security*, vol. 4, no. 1, pp. 1–13, 2013.

[17] M. Xie, H. Zhu, Y. Feng, and G. Hu, "Tracking and repairing damaged databases using before image table," in *Proceedings of the Japan-China Joint Workshop on Frontier of Computer Science and Technology (FCST '08)*, pp. 36–41, IEEE, Nagasaki, Japan, December 2008.

[18] X. Xia, Q. Ji, and J. Le, "Research on transaction dependency mechanism of self-healing database system," in *Proceedings of the International Conference on Systems and Informatics (ICSAI '12)*, pp. 2357–2360, May 2012.

[19] U. Rao and D. Patel, "Incorporation of application specific information for recovery in database from malicious transactions," *Information Security Journal: A Global Perspective*, vol. 33, no. 1, pp. 35–45, 2013.

[20] B. Panda and Y. Zuo, "Fuzzy dependency and its applications in damage assessment and recovery," in *Proceedings of the 5th Annual IEEE SMC Information Assurance Workshop*, pp. 350–357, IEEE, June 2004.

[21] B. Panda and S. Tripathy, "Data dependency based logging for defensive information warfare," in *Proceedings of the ACM Symposium on Applied Computing (SAC '00)*, pp. 361–365, March 2000.

[22] X. Qin, J. Sun, and J. Zheng, "Data dependency based recovery approaches in survival database systems," in *Proceedings of the 7th International Conference on Computational Science (ICCS '07)*, vol. 2, pp. 1131–1138, 2007.

[23] S. Tripathy and B. Panda, "Post-intrusion recovery using data dependency approach," in *Proceedings of the IEEE Workshop on Information Assurance and Security*, pp. 156–160, IEEE, West Point, NY, USA, June 2001.

[24] B. Panda and R. Yalamanchili, "Transaction fusion in the wake of information warfare," in *Proceedings of the ACM Symposium on Applied Computing, Special Track on Database Systems*, 2001.

[25] B. Panda and J. Zhou, "Database damage assessment using a matrix based approach: an intrusion response system," in *Proceedings of the 7th International Database Engineering and Applications Symposium (IDEAS '03)*, pp. 336–341, July 2003.

[26] J. J. Dvorský and M. Krátký, "Multi-dimensional sparse matrix storage," in *Proceedings of the Annual International Workshop on DAtabases, TExts, Specifications and Objects (DATESO '04)*, 2004.

[27] Microsoft Corporation, "Northwind and pubs Sample Databases for SQL Server 2000," 2015, http://www.microsoft.com/en-us/download/details.aspx?id=23654.