Information Warfare: Fighting Back Efficiently Through the Matrix

Ramzi A. Haraty
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
rharaty@lau.edu.lb

Sanaa Kaddoura, Ahmed Zekri
Department of Mathematics and Computer Science
Beirut Arab University
Beirut, Lebanon
sana.kadoura@hotmail.com; ahmedzekri@yahoo.com

Abstract—**Intrusion detection systems can detect a malicious transaction in a database. However, sometimes this process takes time and the detection occurs after the transaction commits. Databases cannot take any action in this case and the damage will spread to a certain part of the database. There are some methods to recover the damaged part of the database. Nevertheless, any recovery algorithm should be fast in order to decrease denial of service time. In this paper, we present a new damage assessment and recovery approach that recovers the database from malicious transactions in the least possible time. The algorithm exploits the data dependency approach to store the needed log file data in a single matrix that will be later used during recovery.**

I. INTRODUCTION

Any computer that is connected to a network is vulnerable to malicious attack. The intrusion detection system can detect an attack and there are a lot of efforts in this domain like in [1] and [2]. Nevertheless, there is no guarantee that this detection will be immediate. Any delay in the detection process will give time to the attack to spread in the database. For this reason, there is always a need for an efficient recovery algorithm.

Traditional recovery techniques such as [3], [4] and [5] can recover a database in case of system failure. However, in case of malicious attack, all the transactions that directly and indirectly read from the malicious transaction should be undone and redone.

In this paper, we assume that the intrusion detection system provides our approach with the set of malicious transactions and our approach should take care of the recovery. Because some transactions may read data items written or updated by other transactions, transaction inter-dependency will take place. Thus, many transactions and data items will be indirectly affected by the malicious transactions. In order to return the database to consistency state, all these transactions will be rolled-back in the least possible time. The more is the accuracy of the algorithm, the less is the recovery time because only affected part of the database will be under recovery and the benign part will be available to other applications.

This research proposes a new approach for damage assessment and recovery in databases. The approach is based on matrices. We use a single matrix to store the dependency between data items in order not to read the entire log file during the damage assessment process. As a transaction commits, a new row will be added to the matrix. The algorithm is efficient and faster than other presented algorithms. Using only a single matrix decreases the execution time of the damage assessment process and it saves memory.

The next section we discuss the related work. Section 3 presents the proposed approach. Section 4 is a conclusion.

II. LITERATURE REVIEW

In some cases, the intrusion detection system cannot detect the malicious transaction directly. This detection delay will make the database exposed to malicious attack. For this reason, damage assessment and recovery algorithms always needed. Recovery approaches are categorized in two main categories: transaction dependency and data dependency. Algorithms based on transaction dependency take into consideration the data items read by a transaction and were updated by another one. Data dependency deals with the operations of the transaction. A data item doesn't necessary depend on the whole transaction; it may be affected by only one operation of the transaction.

In [6], the proposed approach was based on matrices. The dependencies between transactions were stored in dependency matrix. The cells of the matrix hold values that give an evidence of the transactions that one transaction are depending on. If one transaction was depending on more than one other transaction, a complementary array was needed to handle this case. The matrix is static matrix which causes memory problem.

In [7], the authors suggested a new approach that is based on matrices. However, it uses a single matrix without the need for complementary array which saves memory and decreases running time.

Some recovery algorithms were based on clustering. In [8], the authors applied clustering algorithms on the log file in order to segment it into clusters based on data dependency. The

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.73

IEEE computer society

drawback of this algorithm was the growing size of the clusters. In addition, dependent transactions may belong to different clusters. The authors of [9] solved this issue by proposing the sub clustering approach. Each cluster was then sub clustered based on either the space occupied or the number of data items in the cluster.

The authors of [10] suggested an algorithm that uses agents in order to reduce the number of accesses to the log file. The agents work in parallel to perform the damage assessment and recovery which minimize the denial of service time.

In [11] the authors presented the column dependency approach. The columns represent the attributes in the database. The authors proved that as the number of malicious transactions increase, the inconsistencies in the database will increase too.

The authors of [12] suggested an approach that uses two components that works in parallel to perform the damage assessment and recovery. They applied their approach on distributed system where they keep a copy of the log file at each site. Because the components work in parallel, the execution time was reduced.

The authors of [13] suggested using transaction fusion in recovery. They applied their algorithm on real time database systems. Transaction fusion decreases the number of transaction that will be recovered. This will decrease recovery time.

In [14], the authors suggested and approach that works at the operating system level. It is based on transaction dependency and is based on selective recovery that undoes only suspect transactions.

The authors of [15] show some cases where traditional transaction dependency approach cannot handle. They suggested new inter-transaction dependencies: phantom dependency, pseudo-identity dependency, domain integrity dependency and reference integrity dependency. Their approach was implemented and tested in [16] by using SQL rewriting. They showed that exploiting the aforementioned inter dependencies leads to more consistency in the database.

In [17], the authors suggested adding a new table called the before image table. This table has the same structure as the other tables in the database without any constraints and cannot be accessed by users. The purpose of adding this table stores the values of the deleted items in the database so that if they were deleted by a malicious transaction, they can be restored. Also, when an item is updated by malicious transaction, the old value will be stored in the before image table to be used upon recovery. However, the queries that are performed on the before image table affects the efficiency of the algorithm in a negative way.

The authors of [18] suggest that the recovery algorithms should be application specific. The banking system was taken as an example. There are two types of transactions in such system: deposit and withdraw. Taking into consideration the semantics of the each type of transactions, time can be saved.

The authors of [19] suggested an algorithm that is based on the fuzzy logic. Fuzziness decreased the time of damage assessment because there is no need for exact transaction dependency. However, this approach is not reliable because it is not accurate.

In [20], the approach was based on studying the actions done by the users that access the database. All the users' actions are assumed to be malicious until the system proves that they are clean. There is a certain amount of time that should pass to study the users' transactions. If the system could proves that it is clean, the transaction will commit, and otherwise, it will be aborted.

III. THE PROPOSED MODEL

The algorithm is based on data dependency approach because it is considered more accurate than transaction dependency. Data dependency deals with the operations of the transaction instead of dealing with the whole transaction. The dependencies are stored in one matrix which saves memory and reduces recovery time. This is one of the key advantages of the proposed approach.

A. 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 is rigorous serializable because 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.

B. 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 wi[x] of a transaction is dependent on a set of data items I, if $x = f(I)$; i.e, the values of data items in I are used in calculating the new value of x. If $x \neq I$, 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 guarantees strictness, and no data item may be written until the transaction which previously read either commits or abort [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].

C. The Damage Assessment Algorithm

The damage assessment algorithm is based on data dependency approach. 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 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.

The algorithm needs only one data structure to store the dependencies. A two-dimensional matrix will keep track of the dependencies between the data items that can directly point out the affected data items without any need to scan the whole matrix or the log file. Using only one matrix makes no need for any logical operations such as the work done in [26].

The matrix is dynamic structure that grows as the number of committed transactions increase. Initially, the matrix is empty. When a transaction commits, 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. Also, when a data item is written by a transaction, a new column will be added to the matrix to reserve an index for this data item for later use. 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 two columns to the matrix which are the number of data items in a row and the index of the first data item. Because the matrix is sparse matrix, the algorithm will move over a lot of empty cells in order to search for the data items that were written by certain transaction. To save time, the first column tells the algorithm how many data items it should expect. In this case, when it reaches this number, the algorithm will stop searching. The second column is the index of the first data item in a row. For example, if the first data item was at index 5, there is no need to scan the row from 1 to 4. The algorithm starts at 5. This also saves the time of the damage assessment process.

The blind write was also considered in the matrix so that execution time will be decreased. Consider a data item that becomes affected by malicious transaction T_x. Then, another transaction T_y, where y > x and T_y is a clean transaction, blindly written that data item, then, this data item will become clean. All the transactions T_z where z>y, should not be scanned because they will be updated by reading a clean data item. So, we add a value of -1 in the cell that corresponds to the transaction T_y and the data item that was blindly written. When the algorithm encounters the -1 value, it stops working on this data item and removes it from the affected data items list.

Fig. 2 shows the matrix that will be generated after the commitment of the transactions of Fig. 1. 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 saves damage assessment time.

Assume that T1 is the malicious transaction. T1 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 T4 and T5. Since T4 commits after T1, then, the activity at T4 will be checked. There is no need to look at any transaction that precedes T4 since damage spreading started at T4. Data item A was written after reading B in T3 but it will not be considered affected because T3 committed before T4. When T3 committed, data item B was clean.

Considering the data item B, we will move on to the Transactions Ti 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 the Transactions Ti where i > 5, in column E to detect the data items that were affected by E. The result should be X and D. However, E was blindly written before being used in writing X and D because there is a -1 value at the row of T6. Therefore, E became clean before being used in transaction T7. 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 Fig. 3. Assume that the dependency matrix is called M and there is n-transactions. The column that corresponds to the number of data items in the row has an index 0 and the column corresponds to the index of first data items in the row has and index 1.

T1: C := D	T2: D := D + 2
T3: A := B + 1	T4: B := C
T5: E := C + 3	T6: E := 3
T7: X := E + 5	T8: D := E + B
T9: Y := B	

Fig. 1 Sample Transactions Set

	Number of data items in the row	Index of first data item	C	D	A	B	E	X	Y
T1	1	3		1					
T2	1	3		2					
T3	1	5				3			
T4	1	2	4						
T5	1	2	5						
T6	1	6					-1		
T7	1	6					6		
T8	2	5				2	2		
T9	1	5				7			

Data items used the transactions

Fig. 2 The Dependency Matrix

```
Receive a set of malicious transactions S
While there is unprocessed transaction in S
    Select the minimum unprocessed transaction id Ti in S
    //Identify the data items set D written by Ti
    n=M[i][1]
    m= M[i][0]

        for l=n to end of columns
                if m=0
                    break
                if M[i][l] is not null
                        add M[i][l]to D
                        m--
                n++
    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][k] = -1 /*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
```

Fig. 3 The Damage Assessment Algorithm

D. 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 re-executed. The other part of the database will be available. The operations of malicious transactions will be undone.

```
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
```

Fig. 4 The Recovery Algorithm

E. Check Points

The matrix may grow and the data inside it may become obsolete. Thus, to save memory, the above algorithm requires 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 re-read the log file. Also, 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. It will be time consuming if we re-read 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 will be used is Condensed Row Storage (CRS) [27]. The CRS format makes no assumption about the sparsity structure of the matrix and doesn't store any unnecessary element of the matrix. Assuming we have an M ×N sparse matrix A = [aij], containing NZ non-zero elements, the CRS format is constructed as follows:

- One dimensional vector AN holds all the non-zero values of the matrix A.
- One dimensional vector AJ which has an equal length to AN and holds the column number of each element (starting from 1).
- One dimensional vector AI stores the locations in the AN vector that start a row.

Thus, the CRS of the matrix in Fig. 2 will be:

AN = [C D A B E E X D D 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 serializable, we will only build the matrix starting from the malicious transaction. The transactions that precede the malicious one don't 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 checkpoint 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 re-built.

F. 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:

- Doctor (DrID, DrName, DrSpecialization)
- Patient (PID, PName, PGender)
- Disease (DID, DName)
- PatientRecord (PID, DrID, DID)
- PatientBillItems (PBID, PID, Nitems, cost)
- PatientBill (BID, PID, Nitems*cost)

Consider the following insert transactions in the database:

- T1 = Doctor (‘1’, ‘Mike’, ‘Allergist’);
- T2 = Patient (‘5’, ‘Hana’, ‘F’);
- T3 = Disease (‘11’, ‘eye allergy’);
- T4 = PatientRecord (‘5’, ‘1’,11);
- T5 = PatientBillItems (3, 5, 6, 150)
- T6 = PatientBill (2, 5, 900)

The dependency matrix M will hold the dependencies of the above committed transactions. The transactions T1 to T3 don’t 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. T4 and T5 are insert transactions but since they read values from other transactions and put them in another table, the data items will be considered dependent. T4 reads from T1, T2 and T3. T5 reads from T2. T6 is an insert transaction that reads from T5 and writes a new value.

Consider the case where damage assessment algorithm receives from the intrusion detection system that T5 is a malicious transaction. T5 writes 3, 5, 6 and 150. After that T6 reads 5, 6 and 150 to write 5 and 900. Then, from the columns of the matrix, the data item 5 and 900 are affected and will be added to the affected data set. The operation of *Nitems*cost* in T6 should be rolled back and re-executed. T5 operations must be deleted.

	BW	5	1	11	6	150
T1	1					
T1	Mike					
T1	Allergist					
T2	5					
T2	Hana					
T2	F					
T3	11					
T3	Eye allergy					
T4		5	1	11		
T5	3	5				
T5	6					
T5	150					
T6	2					
T6		5			900	900

Fig. 5 Sample Matrix

IV. CONCLUSION

This paper presents a new efficient algorithm for damage assessment and recovery for databases. The algorithm is based on data dependency. The dependencies are saved in a two dimensional matrix that is updated as new transaction commits. Using only a single matrix, the damage assessment time will be decreased and saves memory. The affected data items will be directly retrieved because they appear in the column of the data written by the malicious transaction. All these properties decreases processing time and in turns reduce denial of service of the database management system. After implementing the algorithm, it will be compared to the approaches that use matrices.

ACKNOWLEDGEMENTS

This work was sponsored by the Lebanese American University – Beirut, Lebanon.

REFERENCES

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

[2] L. LaPadula. "State of the Art in Anomaly Detection and Reaction", Technical Report, Center for Integrated Intelligence Systems, The Mitre Corporation, Bedford, MA, July 1999.

[3] P. Bernstein, V. Hadzilacos, and N. Goodman, “Concurrency Control and Recovery in Database Systems”, Addison-Wesley, Reading, MA, 1987.

[4] R. Elmasri and S. B. Navathe, “Fundamentals of Database Systems”, Second Edition, Addison-Wesley, Menlo Park, CA, 1994.

[5] J. Gray and A. Reuter, “Transaction Processing: Concepts and Techniques”, Morgan Kaufmann, San Mateo, CA, 1993.

[6] R. Haraty and M. Zbib. "A matrix-based damage assessment and recovery algorithm", in Proc. Innovations for Community Services (I4CS), 14th International Conference, 2014.

[7] S. Kaddoura, R. Haraty, A. Zekri, and M. Masud, "Tracking and Repairing Damaged Healthcare Databases Using the Matrix", International Journal of Distributed Sensor Networks, Article ID 914305, 2015 - in press.

[8] R. Haraty and A. Zeitunlian. "Damage assessment and recovery from malicious transactions using data dependency for defensive information warfare". ISESCO Science and Technology Vision, vol. 3, no. 4, 43-50, 2007.

[9] R. Haraty and H. Mohsen. "Efficient Damage Assessment and Recovery Using Fast Mapping", in Proc. Second International conference of advanced computer science and information technology (ACSIT), June 2014.

[10] 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 Proc. 48th Hawaii International Conference on System Sciences, Jan 2015.

[11] A. Chakraborty, A. Majumdar, S. Sural. "A column dependency based approach for static and dynamic recovery of databases from malicious transactions", *International Journal of Information Security (ACM)*, vol. 9(1), pp. 51 – 67, Jan. 2010.

[12] P. Liu and M. Yu, "Damage assessment and repair in attack resilient distributed database systems", *Computer Standards & Interfaces*, vol. 33 (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, pp. 473–479, 2012.

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

[15] G. Fu, H. Zhu, and Y. Li. "A Robust Damaged Assessment Model for Corrupted Database Systems", in *Proc. 5th International Conference on Information Systems Security*, pp. 237-251, 2009.

[16] G. Fu, H. Zhang, X. Liu. "Efficient Damage Propagation Detection for Compromised Database Systems". *Journal of Network & Information Security*, vol. 4(1), pp. 1-13, 2013.

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

[18] 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(1), pp. 35 - 45, 2013.

[19] B. Panda and Y. Zuo. "Fuzzy dependency and its applications in damage assessment and recovery", in *Proc. the 2004 IEEE Workshop on Information Assurance*, 2004, pp. 350 – 357.

[20] D. Hua, Q. Xiaolin and Z. Guineng. "SQRM: An effective solution to suspicious users in database", in Proc. The Third International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA), 2011.

[21] B. Panda and S. Tripathy. "Data dependency based logging for defensive information warfare", in *Proc. the 2000 ACM Symposium on Applied Computing*, 2000, pp. 361-365.

[22] X. Qin, J. Sun and J. Zheng. "Data dependency based recovery approaches in survival database systems", In *Proc. Computational Science -- ICCS 2007: 7th International Conference 2*, 2007, pp. 1131-1138.

[23] S. Tripathy and B. Panda. "Post-Intrusion Recovery Using Data Dependency Approach", in *Proc. the 2001 IEEE Workshop on Information Assurance and Security*, 2001, pp. 156-160.

[24] B. Panda and R. Yalamanchili. "Transaction Fusion in the Wake of Information Warfare", in *Proc. the 2001 ACM Symposium on Applied Computing, Special Track on Database Systems*, 2001.

[25] B. Panda and J. Giordano, "Reconstructing the Database after Electronic Attacks", in *Proc. IFIP TC11 WG 11.3 Twelfth International Working Conference on Database Security XII: Status and Prospect*, pp. 143-156, 1999.

[26] B. Panda and J. Zhou. "Database damage assessment using a matrix based approach: An intrusion response system", in *Proc. the 7th International Database Engineering and Applications Symposium (IDEAS 2003),*2003, pp. 336 – 341.

[27] J. Dvorský J and M. Krátký. "Mutli-dimensional Sparse Matrix Storage", DATESO 2004.