LEBANESE AMERICAN UNIVERSITY

An Effective Hash Based Assessment and Recovery Algorithm for Healthcare Systems

By

Bahia Boukhari

A thesis

Submitted in partial fulfillment of the requirements

for the degree of Master of Science in Computer Science

School of Arts and Sciences

December 2019

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS APPROVAL FORM

Student Name: Bahia Boukhari I.D. #: 201604173

Thesis Title: An Effective Hash Based Assessment and Recovery Algorithm for Healthcare Systems

Program: Computer Science

Department: Computer Science and Mathematics

School: Arts and Sciences

The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

Master of Science in the major of Computer Science

Thesis Advisor's Name: Ramzi A. Haraty

Signature: [redacted] Date: 03 / 12 / 2019 (Day / Month / Year)

Committee Member's Name: Samer Habre

Signature: [redacted] Date: 03 / 12 / 2019 (Day / Month / Year)

Committee Member's Name: Azzam Mourad

Signature: [redacted] Date: 03 / 12 / 2019 (Day / Month / Year)

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS COPYRIGHT RELEASE FORM

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

By signing and submitting this license, you (the author(s) or copyright owner) grants the Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic formats and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: Bahia Boukhari

Signature: [redacted]

Date: 03 / 12 / 2019
Day Month Year

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

1. I have read and understood LAU's Plagiarism Policy.
2. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
3. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Bahia Boukhari

Signature [redacted] Date: 03 / 12 / 2019
Day Month Year

ACKNOWLEDGMENT

This thesis is completed with the help of multiple members.

First of all, my genuine gratitude I express for all the help I received from Dr. Ramzi Haraty while advising my thesis. With his professionalism and expertise, Dr. Haraty was always present to guide me through any confusions or troubles I faced. Also, I thank Dr. Samer Habre and Dr. Azzam Mourad for being very supportive and helpful committee members.

Finally, thank you for my family who encouraged me with love and patience to achieve this accomplishment.

An Effective Hash Based Assessment and Recovery Algorithm for Healthcare Systems

Bahia Boukhari

ABSTRACT

The immense improvements in the latest internet inventions encouraged the adaptation of technology within the healthcare sector. The healthcare systems storing highly sensitive information can be targeted by attackers aiming to insert, delete, or modify the data stored. These malicious activities may cause serious harm to the database accessibility and lead to catastrophic long-term harm to the patients' health. Since the adaptation of the most advanced security paradigm does not guarantee a full protection. Also, it is possible that the attack is not directly detected. Hence, this highlights the need for an algorithm that is capable of assessing the widespread damage scale before starting the repair of the inconsistent medical database. Within the scope of the damage assessment and recovery, several matrix based, cluster based, and graph based models were introduced. We propose a new hash based technique that is capable of correctly assessing the damage and recovering the database within a suitable time frame and efficient utilization of memory. Finally, the experimental results prove the improvements provided by our hash based algorithm over previously suggested models.

Keywords: Damage Assessment, Database Recovery, Information Warfare, Malicious Transactions, Transactions Dependency.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

Chapter One

Introduction

1.1 Overview

Currently, data plays a major role within the internet world we live in. Online databases save millions and even billions records of data from various resources. The massive improvements of technology in the previous years increased the online information storage. The utilization of stored information varies with respect to the context of the application. Given an example, an organization relies on a set of stored information as a basis for decision making. Also, data is modeled in order to derive the needed steps for process improvements. Another example that illustrates the importance of data is the medical field. Health care records store all the medical history of the patients including surgeries, medications, allergies, and much more. Hence, all the data utilization should be based on a clean version of records that are protected from different types of attacks. Due to the technological evolution, internet attacks are becoming more and more sophisticated (Jing et al., 2019). Integrity, availability, and consistency of online data are affected by an adversary who is trying to have unauthorized access to systems and online data. These attacks affect small to enormous medical centers with high-security systems. Also, the malicious alterations of patients records may

lead to catastrophic events such death and prolonged harmful effects on patients' health.

There exist three main security phases for protecting systems and information that are prevention, detection, and correction. The prevention phase includes access control, authentication, authorization, and many more techniques that help in protecting from unauthorized access to systems and disclosure of data. Detection methods contain intrusion detection systems, message digest, and checksums (Khochare et al., 2011). Backup and logging are the main techniques applied under corrective methods. Securing information is a journey that passes through multiple related phases (LaPiedra, 2002). Each phase has its own steps for moving to the next one. In order to gain a step ahead of the attacker, thinking outside the box is needed for designing the required capabilities of each phase. The main attributes of information that need to be protected are:

1. Confidentiality: The information is accessible only by authorized people.

2. Integrity: Protecting the information from unauthorized alteration or modifications.

3. Availability: Authorized users can access requested available information.

Different attacks may target different attributes. The disclosure of information by unauthorized parties is an attack on confidentiality. The disruption of data is an example of an integrity attack. Finally, the denial of service is considered an attack that targets availability. Thus, all theses phases enforce information security paradigm that includes confidentiality, integrity, and availability (Bidgoli, 2006).

Starting with the prevention phase in which the design and implementation of security policies, processes, and controls is done. Security programs and access controls form the main components of security policies (Fathy et al., 2013).

- Security Policy: It defines what is protected and clearly states the responsibility of each and every person taking part in applying it. All the steps needed for implementation, enforcement, auditing, and review are mentioned in a clear, coherent, and understandable style. Also, all the requirements needed for the policy are made known once the policy becomes ready.

- Security Awareness: It is very critical for employees to understand their responsibilities mentioned in the security policy to guarantee its correct implementation. Hence, security awareness is a procedure that raises the attention to these responsibilities for all parties in a continuous educational manner. In addition, the training phase for employees makes them ready to actively interact in applying security practices.

- Access Control: Access to information systems resources should not be granted to everyone. Each user is given access based on some identification, authentication, and authorization methods. The identifiers are important for uniquely differentiating between users. After providing the identity, the authentication step validates it before giving access to the resource. Finally, authorization allows identified and authorized users to gain access to the requested asset. The authorization is given on the basis of least privilege such that only the needed authority for performing an action is granted.

The detection phase comes into the scene once the preventive measures fail to

protect the asset. According to Bendovschi (2015) nothing proves the existence of a "silver bullet" security solution. Then, the occurrence of any attacks should be detected by utilizing an intrusion detection system. This tool is deployed within the network that is monitored for any attacks signals. In addition to notifying for any attack occurrence, an intrusion detection system can identify the attacks sources and scales.

In order for the detection phase to have a sound value, the correction phase is needed. After being notified about the attack, the damage is assessed and the system is recovered. The correction phase goal is to restore the stable states of the attacked assets. This goal can be achieved by applying efficient algorithms that can effectively assess the damage in order to correctly recover the database. A great attention is given for finding efficient solutions for both detection and correction phases due to the fact that preventing an attack is very hard due to the advancement of attacking tools within a highly connected internet world.

1.2 Motivation

Nowadays, the integration of technology within almost every field becomes a necessity for service enhancement and better customers satisfactions. Online medical systems serve a huge amount of users on a daily basis. These users utilize the web for checking healthcare records, reserving appointments, buying prescriptions, and many more. Hence, the online presence of these systems requires big storage databases to handle the pressing needs of all of these users. Thus, the data stored on online medical databases is growing in size and diversity. Unfortunately, these databases are subject to different types of attacks that involve malicious

transactions modifying, deleting, inserting, or exposing some private data. Since it is highly possible that these attacks stay unnoticed for a long period of time, their damages start spreading. When many active transactions occur in large scale medical systems, it becomes harder to detect any abnormal behavior. In addition, any latency in reporting the attack by the intrusion detection system causes an increased spread of the damage. Hence, it is highly important to figure out the attack scale as early as possible in order to quickly recover the biggest amount of health related data.

In traditional recovery, many malicious transactions are treated as clean ones as long as they do not violate the ACID properties of the database. However, this approach would not be able to detect the attack at is the earliest phases as a result, the damage starts affecting many clean transactions. Moreover, applying the traditional recovery to the database encompasses undoing the whole affected and unaffected transactions that occurred since the attack is detected. The database recovery is done by performing a scan to the log file from the attack occurrence spot until reaching the last entry in the log file. Thus, the time needed to recover the database is very long causing an extended offline time of the recovered system. Then, the malicious transactions are removed, whereas all transactions that are considered affected are re-executed knowing that many of them may be independent of the damage. Using this method, a redo operation is applied for all affected and unaffected transactions simply because their commitments follow the one of the malicious transaction. It may be argued that all these transactions may be affected; however, when the number re-executed clean transactions is large, the recovery efforts are wasted. When the recovery pro-

cess is slow, the denial of service period increases; therefore, it is unintentionally helping the attacker in achieving the aimed goal. Hence, old fashioned recovery approaches are inefficient especially with the current availability demand for big data platforms.

Recent recovery approaches tried to solve the inefficiency of traditional recovery by proposing new techniques for achieving faster recovery. Many researchers targeted reducing the recovery time by proposing alternative ways that decrease the log access time. This, motivated us to conduct this research to come up with an effective solution that applies the needed work to recover an attacked database. The focal point of our work is to achieve a reduced recovery time following a fast and efficient assessment of the attack by utilizing the suitable data structure. The main contributions of this thesis are

- Devising an efficient solution for recovering medical databases following an effective assessment of damage.
- Portability of the hash based technique to hardware levels.
- Reducing the number of used data structures to a single hash table.

1.3 Problem Statement

Once a database attack takes place, database recovery becomes inevitable. In order to avoid the damage spread, the recovery process should be applied directly after the attack is discovered by the intrusion detection system. It is highly possible that the attack is not discovered on spot; therefore, the database is affected by some transactions that hinder its consistency. These transactions are of two groups: malicious and affected. Starting with the malicious type

that contains a singular or multiple transactions operated by the attacker entity. These operations may insert some inaccurate records in the database. Then if the damaged data is read by clean transactions, they turn to be affected. Thus, it is very important to detect the damage spread by considering not only malicious transactions but also affected ones to have full coverage of the damage.

The manner in which the damage is assessed can be categorized into transaction dependency Panda and Zhou (2003) and database dependency Panda and Asharful Haque (2002) paradigms. The transaction dependency paradigm, as the name implies, focuses on transactions rather than data items to build the dependencies needed for assessing the damage. Following this paradigm, the transaction T_2 becomes affected after reading a data item written by T_1 that is identified to be malicious. On the contrary, the data dependency paradigm keeps track of the damage propagation in terms of data items affected by a series of reads and writes. Therefore, x becomes affected if its modification by transaction T_2 is based on a read of the maliciously modified value of y. The damage assessment algorithm plays a major role in picturing the amount of damage caused by an attack by grouping the transactions that happened after the attack took place as malicious or affected.

In the time being, following the huge advancements of the various fields in the world of computing, the amount of data stored in online databases had increased. The recovery algorithm would take an intensive amount of time with such huge data, the thing that caused such algorithms to become more and more time-consuming. Also, intruders may target database availability in addition to integrity by launching a denial of service attack to bring the database down.

Thus, the performance of the recovery algorithm is a critical consideration for our research, as the recovery algorithm should be fast, so that the offline period of the database during the recovery phase becomes shorter. In addition, accuracy is taken into consideration while designing recovery algorithms. In order to be accurate, a recovery algorithm should bring back the stable phase of the database in a process that clears out the malicious effects of transactions and re-executes the transactions identified as affected while keeping clean transactions intact (Panda and Zhou, 2003). Also, accurate recovery encompasses saving time by not considering unaffected transactions for re-executions. Hence, efficient recovery entails the accurate categorization of transactions as malicious and affected. Moreover, excessive access to the log file increases the time needed for recovery and I/O usage. Hence, the huge time spent in log file access becomes a bottleneck causing a denial of service for our own database. Therefore, we are interested in finding an efficient and fast solution that assesses the damage and recovers the database while preventing or at least reducing such drawbacks.

1.4 Information Warfare

Information warfare as a term evolved significantly since its early psychological usage until becoming a hot topic in the computing field. The term information warfare was used in the military in 1980 that changed to become a way of living during the Gulf war (Hutchinson, 2006). Within the military scope and according to US Airforce, information warfare is defined as a set of actions taken by an adversary to badly affect information. Also, it encompasses the defending measures taken to protect these information (Endsley and Jones, 1997). The

leading factor that affected information warfare evolution is the advancement in communications technology. It is hard to come up with an exact definition of information warfare. However, in the paper of Hutchinson and Warren (2001), information warfare is defined as a war for dominating the info sphere. Moreover, Libicki (1995) compared the definition of information warfare to a blind man who is trying to understand the nature of an elephant. He may think that the object touched is a rope when in fact it is a tail. Also according to him, the elephant's leg may be identified as tree. Hence, the scope of information warfare is very large and it is highly related to the exact area of interest that a person would like to tackle. In this research, we consider the following perspectives of information warfare:

1. Taking advantage of the adversary following a set of competitive operations that makes use of the adversary's information.

2. Protecting our system from different attacks.

The attacker may use different techniques to gain access to the target system using some vulnerabilities and back doors (Hutchinson and Warren, 2001). Some of these techniques are as follows:

1. Data Disruption: This happens by physically destroying data stored on a medium.

2. Denial of Service: Preventing the legitimate access to data by making it unavailable within a defined time period.

3. Data Stealing: A forbidden collection of competitors' data without being noticed.

4. Data Modification: Unauthorized alteration, modifications, and insertion of data such as fraud.

In order to perform the attacks, an intruder may use one or more of the following tools:

1. Viruses: They are sent mostly via emails attachments that result in the installation of harmful software affecting the host machine.

2. Logic Bomb: These tools may be embedded in a program or are stand-alone programs that are triggered under the occurrence of a set of colliding events.

3. SQL Injection: This technique is usually used to exploit a specific website and web servers vulnerabilities. It is based on the insertion of modified SQL statements that are not handled by the target system. It helps the attacker not only to access but also modify data in an unauthorized manner.

1.5 Scope of Work

In this thesis, we tackle the reaction phase of the defensive information warfare that also includes protection and detection. We propose a unique data structure, a hash table, to be used during both assessment and recovery stages. During the first phase which is the assessment, the hash table stores the needed information to be later used during recovery. In order to reach a better memory utilization, the hash table stores only dependent transactions following transaction dependency paradigm. Moreover, during recovery, log file access is reduced by storing the log file in a dedicated hash table. Following this approach, our algorithm is capable of efficiently preforming the assessment while maintaining a reduced period of

log file scanning. In order to visualize and prove the efficiency of our algorithm, it will be implemented and analyzed with respect to similar work found in the literature. This research aims to reach faster recovery by proposing a suitable data structure that facilitates the assessment process and reduces the log file access; and thus, decreases database offline time.

1.6 Thesis Organization

The remaining part of this report has the following components. Chapter II illustrates previous work done within the scope of damage assessment and recovery from the literature. Then, chapter III introduces our proposed algorithm along with detailed explanation and examples illustrating how the algorithm works. Moreover, chapter IV includes the detailed analyzed results of the performance of our algorithm when compared to previously done work. Finally, chapter V concludes this report and represents the work that can be done in the future.

Chapter Two

Literature Review

2.1 Overview

The significance of damage assessment and recovery within the information warfare, encouraged many studies to tackle this area of research. Some papers targeted improving existing approaches, while others focused on proposing new data structures for assessment and recovery. Moreover, both the transaction and data dependency paradigms were used during the assessment phase. Different researches proposed different data structures as the basis for algorithms used to recover the database after applying a damage assessment. The proposed data structures followed matrix based, clustered based, and graph based techniques. Within this part of the report, various studies found in the literature are covered and categorized according to the used method.

2.2 Classical Recovery

Classical recovery or traditional recovery approaches, preform a log file scan from the attack spot till the whole file is covered so that a complete rollback of the database can be performed by deleting the bad transactions effects and re-executing the transactions categorized as affected. These recovery approaches

can be categorized into different types. The first category is known as transaction rollback (Kumar and Son, 1998). The transactions are rollbacked in reverse order according to Page Oriented Undo which means that updates are restored to their previous logical state. Two different types of transactions rollback exist that are normal rollback and redo rollback. During a normal rollback, the rollback is requested by the transaction or happens after a sudden interruption. Redo rollback comes after the occurrence of a system or server crash by which active transactions are rollbacked.

Redo recovery is the second category of traditional recovery approaches that is mostly used to ensure the saving of transaction operations following a system or server crash. The log file is scanned in order to redo all operations in their same original order. The rollback will bring the database to a state that is physically similar to the one that preceded the attack with a different ordering of data (Kumar and Son, 1998).

The third category of traditional approach is rollforward recovery that is used to restore database consistency following physical or logical error. The database is restored from a saved backup before applying the needed updates. The application of the rollforward approach is highly recommended when a failure occurs for a currently in-use database. Also, in the event of disk space loss the application of rollforward is required for restoring the database.

Another traditional recovery category that was highlighted in Liu and Jajodia (2002) is known as branching. It relies on a the tree structure of database versions. In case any branch is found to be inconsistent, another branch will be used as an alternative.

The final method of traditional recovery, known as redundancy approach, was mentioned in Liu and Jajodia (2002) by which copies of data are used to reconstruct it. Also, data can be reconstructed using other system data that is kept and used when needed.

Panda and Yalamanchili (2001) proposed a fusion technique that provides an alternative approach to traditional ones. The malicious and affected transactions are fused together, then an undo operation will be applied for all newly fused transactions while fused affected transactions are re-executed. The main benefits of this approach are the reduction of data items that are combined during fusion, also the log file access is reduced. In addition, the proposed model follows transaction dependency as an assessment paradigm while taking the following assumptions into account:

1. The scheduler is strictly serializable.

2. Blind writes are not allowed.

3. The protected log file is logging the entire transactions of the scheduler.

The fusion process starts from the point of attack that affected the schedule until reaching the end. Then, the affected and malicious transactions are fused into the same group if no other malicious or affected transactions occur between them.

The proposed fusion scheme was compared to other traditional schemes that may or may not apply some fusion techniques. Results showed that the work of Panda and Yalamanchili (2001) had the best effect by lowering the average required transaction time. When traditional schemes didn't use any fusion tech-

niques, the efficiency of the recovery was harmfully affected in course of the high usage of input and output. Also, traditional schemes that applied the fusion of worthless transactions caused a bad impact on the recovery efficiency.

Another replacement of traditional approaches was proposed in the work of Bai and Liu (2009) that represented a lightweight assessment of damage that can be used for PostgreSQL kernel. The proposed solution, named TRACE, guarantees small run-time overhead by:

1. Instantly removing compromised data.

2. Using multiple versions in order not to block read-only transactions.

3. Concurrently performing damage assessment and damage cleansing for read and write transactions.

In addition, a novel tagging scheme was used to build causality between transactions by avoiding the usage of the log for read operations. There exist two operations modes for the proposed model that are standby and cleansing mode. The standby mode is used when no threats are reported by the intrusion detection system. On the other hand, the cleansing includes quarantine and assessment steps that are executed after the discovery of any malicious transaction.

In their work, Bai and Liu (2009) assumed the existence of an intrusion detection system to report any malicious transactions, also the database does not contain any blind writes. Finally, the performed experiments highlighted the performance of TRACE that achieved zero run time overhead, zero systems downtime, avoided the blockage done to some operations performed in read mode only, and decreased the lags of operations executed in write mode only.

Transaction fusion was also adapted in the work of Chen et al. (2012) that followed a similar approach of recovery algorithm used in the work of Panda and Yalamanchili (2001). The proposed model targeted real-time databases in which time limits of transactions are important. The role of a real-time fusion recovery algorithm is manifested after the occurrence of the attack on the real-time database.

2.3 Matrices in Recovery

Matrices as data structures were widely used in the scope of damage assessment and recovery. Lala and Panda (2001) proposed a model to decrease the time spent in damage appraisal. Multiple data structures were used to keep track of transactions dependencies relationships that are essential for determining the affected transactions. The model is a transaction dependency approach that used the sequence of committed transactions instead of transactions IDs in order to search faster. In the work of Lala and Panda (2001) the following assumptions are taken into account :

1. The attack on the database occurred and the malicious transactions are known.

2. The schedule's history is strictly serializable.

3. The log file saves all the transactions operations is the same order of their occurrence.

4. Access to the log file is restricted and it can not be purged.

Four main data structures carry the main bulk of what is needed by the

algorithms. The dependency list data structure stores for each transaction a sub-list of dependent transactions commit sequence. Also, the precursor list saves for each transaction identifier its corresponding commit sequence list. The precursor bit vector and precursor byte list store the corresponding bit/byte representations of all the transactions precursor lists. Finally, four other data structures contain additional information saved in separate lists and tables.

In another paper, Panda and Zhou (2003) proposed two algorithms for assessing the damage and recovering a database. The first one is a transaction dependency algorithm. On the contrary, the second approach is a data dependency algorithm. The proposed model overcomes the main deficiency of Lala and Panda (2001) manifested in keeping hold of the data item until the totality of the recovery process is done.

The model of Panda and Zhou (2003) releases unaffected data items and avoids the access of the log file by utilizing two-bit matrices to maintain both transactions and data item dependencies. Then, AND and OR logic operations on the matrices are carried to assess the damage. During the damage assessment process, matrices that capture the operations of read and write are formulated on the basis of log file entries. Also, two additional data structures are utilized including Damaged_Data_Vector and Damaged_Transaction_List to keep track of the damaged data items and transactions. A logical OR is performed between the matrix capturing write operations and the vector of damaged data items in order to determine the damage. Then, the results are examined so that the cells containing the value 1 are damaged and are moved to a vector containing damaged data. On the other hand, a logical AND is applied between the matrix

of read operations and the vector of damaged data items. Then, cells that contain the value 1 are added to damage transaction list. However, the main drawback of this approach is the increased time spent in the assessment due to the steps done in building the matrices and performing the logical operations.

The work done in (Haraty et al., 2017, 2018; Haraty and Sai, 2017; Haraty and Zbib, 2014; Haraty et al., 2016; Kaddoura et al., 2015) reduced the number of data structures utilized by the damage assessment algorithm. Transaction dependency approaches was utilized in both (Haraty et al., 2017; Haraty and Sai, 2017). On the other hand, other papers (Haraty et al., 2018; Haraty and Zbib, 2014; Haraty et al., 2016; Kaddoura et al., 2015) utilized the data dependency paradigm for assessing the damage.

The work of Haraty and Zbib (2014) utilizes data dependency approach to extract the dependencies that will be stored in a matrix. These dependencies are used to classify transactions as clean or affected.

A two-dimensional array represents the used matrix in which data items and their corresponding transactions are stored respectively in the columns and rows of the matrix. Each cell in the matrix may have one of the following values:

1. 00: Indicating that there is not any transaction that modified this data item.

2. 01: Indicating that a transaction blindly writes this data item.

3. Transaction with a Negative ID: Showing that the data item modification is related to more than one transactions reads.

In case the negative transaction ID is added to the matrix, a complementary

array is used to carry all other transactions on which this specific cell depends. During matrix traversal, all entries having '00' or '01' are skipped. Finally, all cells with values higher or lower than zero are moved to their corresponding affected or malicious lists.

The recovery algorithm receives both affected and malicious list before proceeding to the recovery process. After completing the recovery, the database is cleaned from all malicious effects and all affected transactions are re-executed. The main drawback of this approach is the extra array needed for building the dependencies the thing that requires larger memory utilization.

Rather than using an extra two-dimensional array, Kaddoura et al. (2015) utilized a single two-dimensional array. At the beginning, the used data structure contains no data; then, for each committed transaction a designated row is populated such that the matrix columns contain all the modified value of the data items. Each matrix cell stores the data item modified by the corresponding transaction. Also, blind writes are added in a reserved row to indicate that the corresponding data item is no longer affected in order to reduce the covered affected transactions.

When the performances of both assessment and recovery algorithms were compared to traditional approaches, they showed noticeable improvements. However, the proposed work utilizes string representations of the transactions that takes longer time in comparison. In addition, the matrix is sparse causing a wasted time in the coverage of multiple empty cells.

In Haraty et al. (2016), the authors proposed damaged assessment and recovery model that is very similar to the work of Haraty and Zbib (2014). The

model was used to remove malicious transactions effects from a data source in a healthcare system. The authors kept the same matrix structure of Haraty and Zbib (2014) to store the dependencies during the assessment phase.

In addition the matrix, a complementary array is used to maintain a list of transactions that a specific cell depends on. In a manner similar to Haraty and Zbib (2014), the values stored by each cell are used to identify if the corresponding transaction is affected or not. The completion of the assessment process triggers the initiation of the recovery phase that undoes all malicious operations and redoes all affected ones.

Both (Haraty et al., 2017; Haraty and Sai, 2017) utilised matrices of a linked list as the main data structure while following transaction dependency paradigm.

Haraty and Sai (2017) proposed a lightweight solution for recovering the attacked database while comparing its performance with other existing approaches. The proposed work focuses on reducing the time spent in the log file access such that high-efficiency standards are maintained. This is a transaction dependency solution that stores all the dependent transactions in a matrix. In addition to reducing the needed data structure to one, the transactions are added to the corresponding linked list only when they are dependent on some other transactions in order to save memory.

The proposed assessment algorithm receives malicious transactions from the intrusion detection system. Then, the dependencies stored in the matrix are populated by adding entries of committed transactions that read one previously modified data item. Each transaction is at the array index that equals its ID. For each entry, a linked list array stores all transactions from which the considered

transaction reads.

The recovery phase follows the detection phase by removing the bad transactions operations and re-executing the transactions considered as affected based on the data stored in the dependency matrix. The performance of the proposed model is assessed by conducting an experiment for comparing the performance of the detection and recovery algorithms with other traditional models. The results showed that the lightweight solution had a better running time than all other selected traditional models.

Moreover, Haraty et al. (2017) proposed an algorithm that utilized the transaction dependency paradigm for building transaction dependencies in a matrix of linked lists. This paper differs from the work done in Haraty and Sai (2017) in terms of the condition needed for adding the transactions to the matrix. In this approach, all committed transactions are added to the matrix rather than adding only dependent transactions. The matrix reserves a row for every committed transaction by maintaining a linked list of transaction IDs that read from this transaction. The main drawback of this model is manifested in the large memory utilization when adding all the committed transactions to the matrix.

The experimental results showed that the algorithm had the best execution time among other matrix-based algorithms. However, in terms of memory consumption, the work of Haraty et al. (2017) was outperformed by Haraty and Sai (2017) that used a more compact matrix size.

The drawbacks of Kaddoura et al. (2015) model were fixed in an improved version of the algorithm done in Haraty et al. (2018). The proposed model of Haraty et al. (2018) stores integers instead of strings providing a reduced time

spent in comparisons. Also, the matrix has a similar structure to the work of Kaddoura et al. (2015) but it preserves two additional columns. The first one indicates for each row the number of data items it contains. The other added column shows the index of the first cell that is not empty. By using this additional information, the detection algorithm was capable to deal with the sparsity issue of the matrix by skipping empty cells and saving a lot of assessment time. Also, the counter column enables the enhancement of the search process by stopping it whenever the number of the covered data items is equal to the number saved in the counter column.

The advantages proposed by Haraty et al. (2018) were tested by running an experiment to check the execution time with respect to the work of Kaddoura et al. (2015). The analysis of the experimental results proved that the suggested model of Haraty et al. (2018) had better execution time than the one of Kaddoura et al. (2015). Hence, Haraty et al. (2018) enhanced algorithm was capable to remedy the drawbacks of Kaddoura et al. (2015) approach.

2.4 Clusters and Subclusters

Log files segmentation is the main concept on which clustering approaches are built. The sub clustering approaches apply further segmentation of the log file to reduce the sizes.

A new protocol for logging all information required for completely repairing an affected database was proposed in (Sobhan and Panda, 2002). The introduced algorithm analyzes the set of transactions predicate and statements that are found in the new log. This log file offers easier and convenient storage of all

the information needed for database recovery.

For each transaction, this information is maintained: single or multiple conditions, single or multiple statements of each condition, and modified data items before and after images. The visualization of transactions as programs encompasses the existence of a predicate that once evaluates to true, its associated statement is executed. Usually, the predicate is followed by a single or multiple related statements that target(s) a specified goal and form(s) a predicate cluster.

According to this method, a branch represents the block of the predicate statement that defines the cluster's predicate along with its corresponding statements. Also, a Predicate Block defines a collection of Predicate Statement Blocks that are connected by an exclusive conditional structure.

There exist two types of Predicate Block: conditional, and unconditional. In the conditional type, each branch has a predicate that is evaluated for the associated statement to be executed once the evaluation result is true. If the result is false, the predicate of the next branch is evaluated. On the contrary, the unconditional type lacks the presence of a predicate for each branch; however, a virtual predicate is assumed to exist having a lasting true evaluation. Hence, all statements are executed for unaborted transactions.

The transaction manager is required to send to the recovery manager all the transactions predicate, statements, and their identifiers. Also, for proper operation the following assumptions are taken into account:

1. Log sequence number (LSN) is used by the model for each log record.

2. Predefined logging protocols are used by the logging scheme.

3. Both undo and redo log files are used.

4. Checkpoints are used on a periodical basis.

5. Cache consistent checkpoints are used.

6. The database schedule's history is rigorous serializable.

7. No purging or modification of the log file are allowed.

8. No nested transactions are allowed.

9. Write operations are executed only on the stable database.

Ragothaman and Panda (2003) proposed three ways for log file segmentation with an aim to decrease log file access; thus, improving recovery. The first approach segments the log file after checking the count of transactions that are committed. In this approach, each segment has specific committed transactions count. In the other method, each segment is formed after the passing of a defined time period. In the final method, the occupation of a specified memory size triggers the segmentation process. The following assumptions form the basis of the model:

1. A rigorous two-phase locking scheduler is used.

2. An intrusion detection system detects the attacker's identity.

3. No purging for the log file is allowed.

4. No blind writes are allowed.

Any of the segmentation processes can be carried for producing the needed resources to be used by the damage assessment algorithm. Then in order to determine affected transactions, an intersection operation is applied between the

list of affected items and the read items collected in a dedicated set. In addition, once a transaction is found affected, its set that collects the written items is moved to a list containing affected items.

The effect of the segmentation on performance is proven in the experimental phase in which the segmented damage assessment approach had a better performance than the unsegmented one.

Moreover, the number of committed transactions and occupied space factors were also referred to in the work of Haraty and Zeitunlian (2007) for clusters and subclusters segmentation. The discussed algorithm is based on exact data dependency and it suggests to partition the cluster into subclusters based on the previously mentioned factors. Moreover, for efficiency and speed purposes, only the subclusters are scanned. Also, controlling the subclusters growth is not affected by the growing size of the original cluster. For proper operation, the model assumes the followings:

1. The intrusion detection system can identity the origin of the attack.
2. The scheduler produces the operations of read and write according to some predefined customizations.
3. A rigorous serializable history is produced.
4. The log that saves the transactions being executed is temporary.
5. Transactions IDs are incremented sequentially.

Enhancing the process of assessing the damage is done by adding two data structures for the sub-clustering model (Haraty and Zeitunlian, 2007). The first

one, known as Transaction Sub Cluster List (TSC), keeps track of the transactions IDs and the corresponding subcluster that stores for each transaction its related modified data. This structuring is important for the detection of affected subclusters during the assessment phase. The second data structure is known as Sub Cluster Data List (SCD) that stores subclusters IDs, the transaction ID with the corresponding read or write data item, actual/overlooked, and predicate or statement. SCD is used to obtain malicious transactions data items after selecting the corresponding subcluster.

The discussed model periodically checks for subclusters that become ready after the temporary log is filled with all transactions that are committed. The subclusters are formed by taking into consideration the count of the transactions that are committed or the subcluster size. In order to determine affected data items, both TSC and SCD lists are checked after the malicious transaction is identified. The damage assessment algorithm uses additional data structures that are Damaged_DI and Damaged_PI. The first list stores directly or indirectly affected data items and the second one saves a directly or indirectly affected predicate block. Initially, both lists are null. Then, determining data items that are affected is done by checking all items in TSC. Finally, the recovery process passes by the following steps:

1. Damaged_PI is scanned once the assessment is done.

2. The subcluster saving each TSC item is obtained.

3. A reevaluation of each block is carried.

4. Items that are restored will be returned to the database.

5. The additional used data structures are freed.

The clustering technique performance discussed in Haraty and Zeitunlian (2007) was compared to other traditional methods. The analysis showed a performance improvement for both assessment and recovery when performed on a clustered log. Moreover, the fixed-size clustering technique proved to be a little bit better than the fixed number of transactions technique.

In the work of Fu et al. (2008), new concepts known as Extended Read Operations and Fine Grained Transaction Log (FGTL) were introduced. In order to produce the FGTL, a dedicated FGTL Generator was developed. The authors state the difficulty of recording what every transaction reads or writes since the logging should be done after the SQL statement is executed. Also, the damage can easily spread following a series of reads and writes. SQL statement subqueries may include nested read operations that are not recorded by most damage assessment approaches. Hence, Extended Read concept can represent the read of SQL subqueries. Then, FGTL can record all the read and write operations of SQL queries and subqueries. The FGTL Generator is responsible for the efficient allocation of the log. For security purposes, FGTL Generator is placed within the database system in which the table that captures the log file transactions is protected from any users modifications.

FGTL table base components include some dedicated read generator, a monitor for the database, temporary database, triggers, and a table for the transactions log. The table dedicated for the transaction log contains all the performed read and write operations. In order to generate the read log entries, all queries sent to the database are analyzed and rewritten. The DB monitor receives the

SQL statement sent by the client user, and extracts it before sending it to the read generator. Then, the statement is analyzed by the generator. The process of generating the read log entries passes by a process known as the Divide and Combination algorithm. This algorithm saves in intermediary tables the read data items of transaction. In order to cover both queries and subqueries, each statement is divided into subparts before being temporary saved. Finally, the log entries are saved in the Transactions Log Table after combining all the generated segments into an SQL sequence with the exact effect of the original statement. On the other hand, write log entries are saved by the triggers in the Transactions Log Table if the modified data item belongs to a user table. Divide and Combination Process is followed by an algorithm for the generation of the transaction log. FGTL records all the users' operations that are later used by the damage assessment algorithm. Thus, in order to protect the generated log, DB monitor analyzes each statement received from the user. If it attempts to modify the Transaction Log Table, it will be discarded. The experimental results showed a weak aspect of this approach caused by the degradation of throughput once the association degree of transaction increased.

Based on the work of Haraty and Zeitunlian (2007), Haraty and Mohsen (2014) tried to enhance the clustering techniques by the utilization of additional auxiliary data structures with the application of fast mapping of information retrieval. In addition to TSC and SCD lists, the model uses the Responsible Transaction List (RTL) and Affected Transactions List (ATL). RTL stores the latest transaction responsible for updating each data item along with their old values. ATL identifies the affected items by the write operation done by each

transaction. In addition, affected predicates/blocks are recorded in order to determine the transactions requiring recovery. After launching the algorithm, ALT sends all the affected transactions for storage. Also, the corresponding data items are stored within dedicated lists. Then when both lists are filled, the recovery process commences. The recovery time of the proposed algorithm is improved by scanning only subclusters and moving back to the database RTL previous values.

The damage assessment algorithm suffers from subcluster size limitation that is controlled by the maximum number of transactions allowed before allocating a new subcluster. Also, the additional proposed data structures increased the memory allocations.

2.5 Before Image Tables in Recovery

Xie et al. (2008) proposed an innovative recovery model for tracking the scale of the attack by storing before image tables (BI tables) and checking read operations performed on them. This process can repair the database with no need to rely on a log file scan. The model presented is capable of inducing the inter-transactions dependencies of deleted transactions carried by an extended recovery to be implemented in a DBMS. Moreover, the recovery is ensured to have exact results by introducing fewer modifications than other recovery models. Since most write operations are based on previous reads of deleted data items, it is problematic to keep track of these deleted items. Most DBMS store deleted data, known as before image, in a complex format for damage assessment. Thus, the BI table was introduced to facilitate this process. Each BI table is tied to a base one such that each deleted record is automatically inserted into the BI table. The

growth of BI table is adjusted by tracking a predefined time period in order to reduce the table size by removing useless data.

Since the model of Xie et al. (2008) captures inter-transaction dependencies, a dependency graph is generated during query processing. Also for each data item x, the transactions that added or deleted this data item are stored respectively in $x.ins\_tran$ and $x.del\_tran$. Using both fields, the model keeps track of the latest modifying transaction for each data item. For all active transactions an ID set, known as DS, is maintained for storing the latest modifying transaction for a read data item x. Also, all dependencies of inter-transactions are stored within a dedicated table. This table has the following columns Xie et al. (2008):

1. *commit_cord*: Used to store the commit order of *dependent_tran*.

2. *dependent_tran*: Identifies transactions that depend on other.

3. *precursor_tran*: Saves transactions dependent by *dependent_tran*.

The various types of DBMS queries are decomposed into subqueries acting on single tables such that the rows satisfying the search conditions are locked and recorded by $x.ins\_tran$ in DS. In addition, each base table executed subquery concurrently runs on the BI table in order to save the potential read data items by $x.del\_tran$ stored in DS.

The recovery process is triggered after receiving an attack alarm. The first step in recovery entails the identification of the transactions to be undone. Since all transaction dependencies are stored in a dedicated table, it is quite easy to identify the affected transactions. The second step of the repair consists of identifying the effects of these transactions so that their removal is quickly carried.

$x.del\_tran$ and $x.ins\_tran$ facilitates the removal of the added data items and the reconstruction of the dumped ones. There exist two operations mode for the recovery process that are static and on the fly. The former operates by bringing the system down. On the other hand, on the fly mode enables the system to stay online by confining damaged data items.

The performance of the model was tested on a prototype DBMS known as ITDM which is an intrusion-tolerant DBMS. The results showed that on the fly mode had tolerable performance degradation and proved the flexibility of implementing it in other DBMS.

2.6 Column Dependency

In the work of Chakraborty et al. (2010), a column dependency-based approach was proposed for identifying malicious and affected transactions. The model has both static and online versions. During static recovery, the database becomes unavailable for users. Initially, compensation and re-execution phases carry the execution of the recovery algorithm. Using static recovery, the algorithm receives the set of malicious transactions accompanied by the execution log and all committed transactions. The expected output is a consistent database with the effects of malicious transactions removed.

On the contrary, online recovery keeps the database available with possible performance degradation. Also, the confinement of damaged data items is very critical to prevent the damage from spreading to active transactions during online-based recovery. The major difference between static and online recovery is the vulnerability window. The online recovery uses a flexible window in which

transactions keep executing. This results in a higher chance of having malicious transactions. Also, online recovery contains more phases than the static approach. The three phases of online recovery are applied through assessment, recovery, and confinement procedures.

2.7 Graphs and Agents in Recovery

Ammann et al. (2002) utilizes graphs to repair attacked databases. The proposed models are categorized into coldstart and warmstart semantics. The first category causes the database to be offline during recovery. On the contrary using warmstart model, the database is kept available with potential performance degradation. The transactions are collected using two main lists. The first collects bad or undesirable transactions. On the opposite side, the second list contains good or desirable transactions. The recovery process was enhanced by building dependent transactions via graph dependency containing both good and bad transactions. This data structure is used to detect which transactions are dependent on the bad collected ones.

In Zuo and Panda (2004) other graph-based models were introduced. In this work, the log file can't be damaged and no blind writes are allowed. The paper discussed the assessment phase that outputs the malicious and affected transactions without implementing the recovery that is postponed for future work. Also, the utilized database is distributed along multiple sites. A manager, known to be local, can be used to establish a correlation with the multiple existing sites. Two models have been discussed within the work of Zuo and Panda (2004).

The first model, known as peer to peer, can be used without the need for a

damage assessment coordinator. The site manager of each site scans the log file and notifies all other sites in which the detected affected transactions have some executed sub-transactions. Then, the notified sites run a further scan to detect any potential new affected transactions to be also sent to other related sites. The followings illustrate the main advantages of this model:

1. Avoiding a single point of failure.

2. Distributed assessment approach.

3. Faster assessment due to the distributed scanning process.

However, this model suffers from high network bandwidth requirements to cope with the huge demand for sharing information between sites. Also, it is complicated to keep all sites synchronized.

On the contrary, the presence of an assessment coordinator is essential for the second model which is the centralized. Electing the coordinator site that holds the coordinator is done through an election process that is carried to pick the site that meets the highest number of following requirements:

1. It is the most suitable place in terms of network distance.

2. The site has the best performance capabilities.

3. It is connected to all other sites via fast network links.

4. Backups are carried by at least one machine on the site.

The centralized model contains the following sub models:

1. Models to Forward and Receive: Global affected transactions from each site manager are received by the coordinator that notifies all sites having executed sub-transactions dependent on the received global ones.

2. Graph Model for Local Dependencies: G_i graph is locally maintained at each site. The coordinator receives these graphs from each site before proceeding to build the list of affected transactions.

3. Central Graph Repository Model: Each local graph is sent periodically to the coordinator that stores them and performs updates only when the received graph differs from the stored one.

An agent based model was introduced in (Kurra et al., 2015). In this work, many agents carry assessment of damage and the database recovery in a parallel fashion. Timestamps and expressions of the versioned data items are maintained. The assessment model only considers data items with some corrupted versions after checking the stored timestamps and expressions. This expedites the recovery process by only considering damaged data items for recovery.

Each agent in the model manages a unique group of data items while maintaining inter-agent communications channels. Also for each managed data, its latest version signifies the most recent value written by a transaction. Whenever a data item is written, its version number is taken from the modification time. In addition, all the modified data items versions are recorded.

DL list is used to maintain a list of damaged data items versions. If an agent A_i updates DL, all other agents get notified about the update. Then, all the timestamps of the added data items in DL are analyzed by every agent. Any version of A_i data items is considered free of damage, if its not part of DL and

its timestamp is less than the current checked one. Special types of messages are exchanged between agents to indicate the status of the assessment phase.

Another agent based model that utilises graphs was presented in Saba (2018). All agents are controlled by a single agent controller that receives and forwards all messages to the controlled agents. The main utilized data structure is a graph to represent dependent transactions. A node for transaction T_2 is added under T_1 to signify that T_2 reads a data item modified by T_1.

Each agent manages a set of graphs enabling the processing of complex problems. Moreover, all agents concurrently scan their graphs to check the existence of any malicious activities. If any malicious activity was detected, the scanning agent reports back to the recovery manager a pointer that can be used to locate transactions considered as malicious or affected. Once the recovery process starts, the returned sub-graphs is scanned. The scan picks the returned ID with the smallest value, then the next ID is selected. The scan stops when the last node reached belongs to a malicious or an affected transaction. Then, the rollback process is carried until reaching a transaction that is not malicious signifying the end of the recovery process.

This approach is capable of easily expanding without affecting the performance of the recovered database since only the affected portion is isolated.

2.8 Fuzzy Dependency

In Yanjun Zuo and Panda (2004) a fuzzy dependency approach specifies a type of logical dependency that can not be expressed by functional dependencies that are used for schemes and organization of data. The proposed model utilizes

the previously mentioned database scheme to perform damage assessment. Also, fuzzy dependency along with its inference rules form the basis of the fuzzy recovery algorithm. Utilizing fuzzy dependency contributes to the faster repair of the database when compared to traditional recovery schemes.

There exist three main usage scenarios for fuzzy dependency. First, it can be used for fuzzy integrity checking by enforcing constraints on a tuple forming an instance r on a relation R. The main benefit of integrity check is defining rules that if violated, illegal users modifications are aborted. The second example of fuzzy dependency usage is playing the role of an intrusion detection system. If $X => F(Y)$ and Y's instance tuple can't have an agreement on the range of the fuzzy value that is determined by X, it means that Y's data value is damaged. The final usage of fuzzy dependency is producing the damaged data items fuzzy values that contributes to a reduced denial of service but with added overhead.

The recovery model utilizes the fuzzy value generator that operates based on some stored fuzzy dependency relationships. The recovery process follows the detection of the attack. Also, even if the supplied fuzzy values are not fully accurate, they are acceptable under the condition that the conventional recovery system handles this inaccuracy and restores the database. Also, fuzzy value generation should perform faster than other recovery processes the thing that is guaranteed by using fuzzy reasoning on stored fuzzy relationships.

Thus, the proposed model of Yanjun Zuo and Panda (2004) contributes to a faster recovery model than traditional approaches without requiring a detailed scanning procedure of the log file. However, the usage of fuzzy dependency requires the supplement of semantic reasoning unit for achieving the needed accu-

racy.

2.9 Distributed Recovery

The existence of indirect dependencies within executed sub-transactions makes distributed damage assessment and recovery more complicated Zuo and Panda (2006). In order to collaboratively proceed with damage assessment, two main approaches were presented in Zuo and Panda (2006) that are peer to peer and centralized.

The first model does not require a coordinator to exist such that each site is a peer to all others. The site manager of each site forwards any global damaged transactions to all sites in which some related sub-transactions were executed. The following points highlight the benefits of this model:

1. Avoiding a single point of failure.

2. Full distribution of the whole process by carrying the same algorithm with balanced data and communications load at each site.

3. Load balancing by distributing the load and offering a collaborative high-speed assessment.

However, this model utilizes a big amount of network traffic due to the information-sharing scheme. Also, synchronization is another issue that the model has to deal with.

On the contrary, the centralized model utilizes an elected coordinator through a voting process. Each site manager can directly communicate with the coordinator by sending all globally affected transactions. Then, the coordinator forwards

the information to all affected sites. The main advantages of this model are avoiding high network traffic and the assessment procedure is not recursively done at each site. On the other hand, the main drawbacks are manifested in the high load posed on the coordinator, also the recovery process is delayed until the coordinator receives all affected transactions.

Another distributed damage assessment and repair was presented in Liu and Yu (2011). The authors highlight three main requirements for effective distributed damage assessment and recovery that are:

1. Performing both models at each site since data is partitioned across multiple sites.

2. All sub-transactions relationships are checked to highlight affecting relationships between distributed transactions.

3. Tolerance for communication and site failures.

The work done in Liu and Yu (2011) contributes to the followings:

1. Full distributed process that avoids the single point of failure.

2. Continuous handling of incoming transactions.

3. Concurrent execution of damage assessment and recovery.

4. It can be easily integrated within DBMS.

5. Transparency of the proposed models.

Two main processes that exist in all sites. The first is known as local damage assessment and repair (DAR) while the second one is a DAR executor. The log

is scanned by the DAR executor at each site in order to identify any affected sub-transactions and prepare their corresponding cleaning sub-transactions. The local DAR manager coordinates between processes that assess and repair the transactions. In sum, the DAR executor is monitored by the manager during the assessment of damage caused by malicious transactions. Also, it coordinates the preparation of cleaning transactions for all identified malicious transactions.

In order to simply carry the repair process, all malicious transactions are repaired by their corresponding cleaning transactions. Each DAR manager produces the cleaning transactions. Then, the DAR executor generates the cleaning sub-transaction for all the sub-transactions belonging to all identified malicious transactions.

2.10 Self-Healing Recovery

The work of Xia et al. (2012) proposed a new approach known as self-healing database that can be used for repairing a database. After the occurrence of the attack, the database heals itself and offers a continuous service. The authors suggest a new self-healing database structure that separates between the intrusion detection system and the method that converges the main-auxiliary database. There exist two main databases in this architecture: main and auxiliary. After recovering the auxiliary database, the main database follows suit. The end of the recovery process should bring back both databases to their consistent states.

The self-healing architecture of Xia et al. (2012) has the following components:

1. Intrusion Detector: Offers a real time detection of malicious transactions by analyzing database trails and transactions.

2. Damage Assessor: Extracts the affected transactions when an attack alarm is fired.

3. Database Repair: Rolls back the auxiliary database after undoing the affected transactions effects.

4. Controller for the Transactions: Transactions are sent to two separate databases. In addition, a buffer is used by the auxiliary database after being brought offline during the repair phase.

5. Repair Log: Stores all modifications of the repair process.

When the attack is identified, the buffer zone maintains the stored transactions until the auxiliary database is recovered. Once the main database becomes ready, its repair process gets executed.

2.11 Multilevel Recovery

A hybrid approach to recover multilevel databases was introduced in (Zarif and Haraty, 2019). The authors suggest that this is the first work to tackle multilevel databases. The main difference between single and multilevel databases is the way data items are stored. In single level databases, each data item is stored once. On the other hand, multilevel databases keep track of multi values for each data item.

The proposed work in Zarif and Haraty (2019) is composed on two main approaches: Graph and clusters. The graph approach applies transactions dependency paradigm between read and write transactions. The clustering approach groups transactions together based on transactions levels.

The authors in Zarif and Haraty (2019) assume the followings:

1. The schedule is rigorous serializable.
2. Polyinstantiation is respected by allowing multilevel tuples having the same primary key, but with an assigned explicit level.
3. Writes are recorded on the same level.
4. Four different security levels are utilized.

Each write is recorded in real time in four buckets representing the level of each transactions. These buckets form the basis of determining transactions dependencies following a single scan of the log. In order to correctly represent these buckets, a bipartite graph is used. In each graph, data items are present on the first side, and the corresponding transactions are present on the second one. Every transaction node contains its ID, the altered entry primary key, its corresponding table name, and the before image.

The log file is clustered into buckets of write, also adjacency lists are used to record all transactions dependencies. In order to capture all affected transactions, the adjacency list is searched using breadth first algorithm. The algorithm starts with the first malicious transaction, then each visited neighbor is collected by a dedicated queue that stores affected transactions. The recovery process is issued after the end of the assessment algorithm. During recovery, all buckets with transaction considered as affected are scanned in order to collect their writes in addition to their before images. The collected data forms the basis of restoring the stable phase of the database.

Chapter Three

The Model

3.1 Overview

A new hash-based technique for damage assessment and recovery is presented in this section. The proposed model has been optimized to increase efficiency as well as decrease memory utilization during the assessment and recovery phases.

To quantify the damage the assessment phase is used. In this phase, the transactions are categorized as affected or clean in order to identify the ones requiring recovery. The end of the assessment phase initiates the recovery phase. The recovery algorithm goes over the affected and malicious transactions while trying to redo the affected ones and undo the malicious ones. The result of the recovery process is a database in a consistent state.

3.2 Definitions

Definition 1: The Transaction T_i's write, denoted as $write_i[x]$, is considered dependent on T_i's read operation $read_i[y]$ if $write_i[x]$ is performed based on value obtained by $read_i[y]$ (Panda and Tripathy, 2000).

Definition 2: When a transaction T_i modifies a data item z without the need to read its latest value, then this write is considered a blind write (Zheng et al.,

2007).

Definition 3: Giving that $z = f(i)$ means: T_i's write operation $write_i[z]$ is dependent on L's data items when the latest value computed of z is based on them. Also, if $z \neq L$ the operation is considered a blind write. Thus when the write operation is finished, z's value will be re-obtained (Panda and Tripathy, 2000).

Definition 4: Given a total order Y under $\leq$, then for all d, e, f in Y the followings hold:

If $d \leq e$ and $e \leq d$ then $d = e$ (antisymmetry).

If $d \leq e$ and $e \leq f$ then $d \leq f$ (transitivity).

if $d \leq e$ or $e \leq d$ (totality) (Boros and Szaz, 2008).

Definition 5: A serializable schedule is one that produces an identical and similar effect of a serial schedule on a consistent database (Sumathi and Esakkirajan, 2010).

Definition 6: A schedule is strict if it follows strict property (Breitbart et al., 1991). The strict property means that a data item can not be accessed in read or write mode if it is being currently modified by another uncommitted or unaborted transaction (Bernstein et al., 1986). Using strict property, the database scheduler avoids Write/Read and Write/Write conflicts (Breitbart et al., 1991).

Definition 7: Two conditions are required to be met for the schedule to be rigorous. First, the schedule has to guarantee strictness. Second, if a transaction T_1 is reading a data item x, x can not be written by T_2 until T_1 commits or aborts (Kim et al., 2010). This property prevents Read/Write conflicts between transactions (Weikum and Vossen, 2001).

3.3 Assumptions

First, a intrusion detection system monitoring its environment for any signals of attack is assumed to exist. After receiving an attack alert from the intrusion detection system, the damage assessment algorithm is fired. In order to start the assessment process, the damage assessment algorithm receives from the intrusion detection system its identified malicious transactions list.

Also, the produced history is rigorously serializable. As mentioned before, the outcome of a serializable history is the same as the serial one (Sumathi and Esakkirajan, 2010). Thus, serializability guarantees correctness (Chrysanthis and Ramamritham, 1998).

Our assessment algorithm follows the transaction dependency paradigm for building the dependencies. If the transaction T_2 reads T_1's modified data item, the algorithm does not focus on which data item is being read. In fact, it captures that T_2 is dependent on T_1 since the read of T_2 is based on the value written by T_1.

The transactions IDs are assigned uniquely and sequentially. The first transaction is given ID = 1. Then, the second transaction is assigned ID = 2. Hence, there is no case that two transactions are assigned the same ID.

We also assume that the log file of the database can not be accessed by users. Moreover, the log file would be the main basis of the recovery procedure.

The new proposed hash based technique was not suggested in any previously done work. This technique relies on a hash table that provides very fast access and search time. The selection of the right data structure is very critical to enhance the damage assessment and recovery algorithm. Also, reducing recovery time is

of a great interest in our study, because the offline time of the database should be reduced especially within the high availability demands of heavily accessed medical databases. Finally, our proposed hash table data structure was optimized towards decreasing memory utilization.

3.4 Overall Algorithm Structure

The damage assessment and recovery are the two main components of the proposed solution. In Figure 1 the solution steps are illustrated. The phase that assesses the damage receives the identified malicious transactions from the intrusion detection system. In case there exist multiple malicious transactions, the transaction with minimum transaction ID is sent to the assessment process.

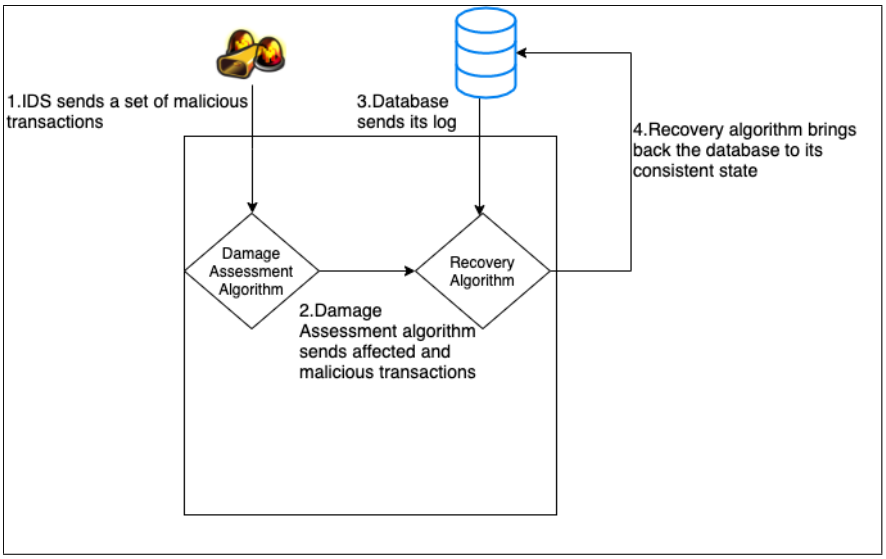


Figure 1: Overall Model Structure

The damage assessment algorithm analyzes the received malicious transactions and outputs their corresponding affected transactions list. Then, the recovery algorithm receives the malicious transactions list along with the affected list outputted at the end of the previous phase. During recovery, the database log is used for the database repair.

The recovery process causes the database to be taken offline. Reaching a reduced database offline time is one of the main goals we aimed to achieve in this thesis. Hence, by proposing a faster algorithm, we increased the database availability. In order to produce faster assessment and recovery, we propose the utilization of a hash table for both phases. Also, providing a memory efficient solution was attainable by storing the only needed information in the hash table to ensure the correct operation of our algorithm.

3.5 Hash Table

The hash table is the main and only data structure used for the damage assessment and recovery algorithm. The first utilization of the hash dependency table is highlighted within the assessment phase in which the hash table is generated. During transactions executions, the hash dependency table is built to store the transactions dependencies. In order to enhance memory utilization, only dependent transactions are added to the hash dependency table.

Transactions may blindly write a data item, keep a data item unmodified, or read single or multiple data items from committed transactions in order to modify a specific data item. Our hash table only stores dependent transactions that read one or more data items modified by previous transactions. Since our assessment algorithm follows the transaction dependency model, it only considers which transaction is dependent on another. In sum, it only captures the fact that the data item written by T_1 is read by T_2. Hence, T_2 is dependent on T_1. The details of dependencies are not needed because if T_1 is the malicious transaction, the recovery algorithm will eventually undo T_1 and redo T_2.

Each transaction stored in the hash table is accessed based on the hash value of its ID. Also for each stored transaction, the hash table keeps track of a list of dependent transactions that contains a set of transactions reading single or multiple data item that were previously altered by the stored transaction. Hence, if a transaction T_2's modification of data item x is carried after reading a value written by transaction T_1, the hash table adds T_2 to the dependent transactions list of T_1. In Figure 2, a sample of the hash dependency table is manifested.

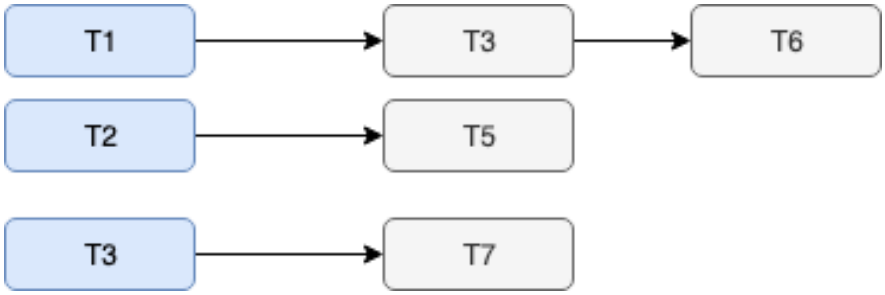


Figure 2: Hash Dependency Table

The process of continuously accessing the log file during recovery utilizes a lot of resources and execution time. In order to deal with such an issue, we stored the log file within a hash table. Hence, resources will be saved and the execution time will be much more improved by reducing the log file access time; thus, improving recovery.

Using the above-mentioned approach, the only data needed for both assessment and recovery is stored. Also, checkpoints are used to control the size of the log file that may become tremendously big. In addition, rather than using secondary or tertiary data structures, we reduced the number of utilized ones to a single hash table.

3.6 Algorithm for Damage Assessment

The main role of the intrusion detection system is to detect and identify the source of the attack. Also, transaction having the smallest ID is selected in case multiple malicious transactions were identified. The assessment algorithm starts the scan from the identified malicious ID. Also if $j < i$, T_j can not depend on T_i because the schedule's history is rigorously serializable. The damage assessment algorithm only considers the transactions whose commitments follow the one of the malicious transaction. In other words, if the transaction with ID equals 100 is the malicious transaction, our assessment algorithm will only consider transactions with IDs greater than 100. Using this approach, the assessment time will be greatly improved by only considering the transactions that may be affected.

The hash table will be scanned in order to collect the transactions that can be directly and indirectly affected. The two categories of affected transactions are directly and indirectly. The former occurs when the maliciously inserted data item is read by a clean transaction while in the later case, the data item read by a transaction was written by an affected one.

When the malicious transaction ID is received, the algorithm identifies its index in the hash table based on the hash value of the transaction ID. Then, it copies the list attached to the malicious transaction to the affected list. Afterward to cover all forms of affected transactions, the algorithm loops over all affected transactions and copies their corresponding dependent list to the affected transaction list until all affected transactions are covered. The algorithm steps are found in Algorithm 1.

Algorithm 1 Damage Assessment Algorithm

Input: Hash dependency table and malicious transactions list
Output: Malicious and affected transactions lists

```
1: locate the malicious transaction index based on its ID hash
   value
2: Copy the index's list to the affected list
3: for t in affected transactions do
4:    locate the transaction index based on t ID hash value
5:    Copy the index's list to the affected list
6: end for
```

Taking into consideration the hash table sample of Figure 2 when the malicious transaction is T_1, the algorithm locates its index in the hash table. Then, both T_3 and T_6 are part of the list that contains affected transactions. This process is repeated by copying the list of T_3 that contains T_7 to the affected list. Since T_6 and T_7 have no dependent transactions, the algorithm stops and outputs T_3, T_6, and T_7 as affected transactions while T_1 is outputted as the malicious transaction.

3.7 Illustration of the Damage Assessment Process

Considering a health sharing database management system storing the following tables:

- Doctors: DoctorID Unique ID of the doctor.

  DoctorName: Name of the doctor.

  DoctorMajor: Major of the doctor.

  DoctorHiringDate: The hiring date of the doctor.

- Patients:

  PatientID: Unique ID of the patient.

  PatientName: Name of the patient.

  PatientBDate: Birthdate of the patient.

PatientAddress: The Address of the patient.

- Category:

  CategoryID: Unique ID for each treatment category.

  Description: Details about each treatment category.

- Treatment:

  TreatmentID: Unique ID of the treatment.

  TreatmentName: Name of the treatment.

  TreatmentCategoryID: The category ID of each treatment.

- Prescriptions:

  PrescriptionID: Unique ID of the prescription.

  PatientID: ID of the patient taking the prescription.

  DoctorID: ID of the doctor giving the prescription.

  PrescriptionDate: Date of the prescription.

- PresciptionDetails:

  PresciptionDetailsID: Unique ID of each prescription detail.

  PrescriptionID: Prescription ID of the current detail.

  TreatmentID: Treatment ID of the current detail.

  PrescriptionDetailFrequency: Frequency of taking the prescription on a daily basis.

  PrescriptionDetailDuration: The number of days required for taking the prescription.

The below transactions modified the above-mentioned database in the following order:

- T_1: Insert into Doctors values (12, Jane, Heart, 1-01-2000)
- T_2: Insert into Category values (3, Blood Vessels)
- T_3: Insert into Treatment values (21, Atenolol, 3)
- T_4: Insert into Category values (18, Vitamin)
- T_5: Insert into Treatment values (29, Niacin, 18)
- T_6: Insert into Doctors values (15, Justin, Dermatology, 02-04-2005)
- T_7: Insert into Patients values (14, Selena, 02-03-1992, New York)
- T_8: Insert into Patients values (23, Sarah, 02-03-1992, Texas)
- T_9: Insert into Prescriptions values (1, 23, 15, 22-02-2019)
- T_{10}: Insert into Prescriptions values (2, 14, 12, 05-07-2019)
- T_{11}: Insert into Prescriptions values (3, 23, 12, 24-08-2019)
- T_{12}: Insert into PrescriptionDetails values (1, 1, 29, 1, 30)
- T_{13}: Insert into PrescriptionDetails values (2, 2, 29, 1, 60)
- T_{14}: Insert into PrescriptionDetails values (3, 3, 21, 2, 90)

The hash dependency table, denoted as H, is built during the execution of the above transactions using the same mentioned order. When T_1 and T_2 are executed, they will not be added to H since no other transactions depend on them. Before inserting its record, T_3 reads the category with ID = 3 that was inserted by T_2. Thus because T_3 is dependent on T_2, both are stored in H such that T_3 is added to the transaction dependent list in the hash table row of transaction

T_2. At its execution time, T_4 is not added to H because no other transactions depend on it. Then before inserting its record, T_5 reads the category with ID = 18 that was added by T_4. Hence, a row in H is allocated for T_4 while adding T_5 to the dependent transactions list of T_4.

T_6, T_7, and T_8 insert new records into Doctors and Patients tables without being dependent on other transactions so they are not added to H. Prescriptions table is dependent on Doctors and Patients tables. Thus, T_9 is depends on T_6 and T_8. T_{10} is dependent on T_1 and T_7. T_{11} is dependent on T_1 and T_8. Hence, T_6 and T_8 dependent lists contain T_9. Also, T_{10} gets added to the dependent lists of T_1 and T_7. Finally, both T_1 and T_8 have T_{11} in their dependent lists.

The final three transactions T_{12}, T_{13}, and T_{14} modify the table Prescription-Details that is dependent on Prescriptions and Treatments tables. T_{12} reads from T_5 and T_9, while T_{13} reads from T_5 and T_{10}, and T_{14} reads from T_3 and T_{11}. Hence, T_{12} is added to T_5 and T_9 dependent lists. Also, T_{13} is part of both T_5 and T_{10} dependent lists. The final transaction T_{14} is added to T_3 and T_{11} dependent lists. The hash table H is shown in Figure 3.

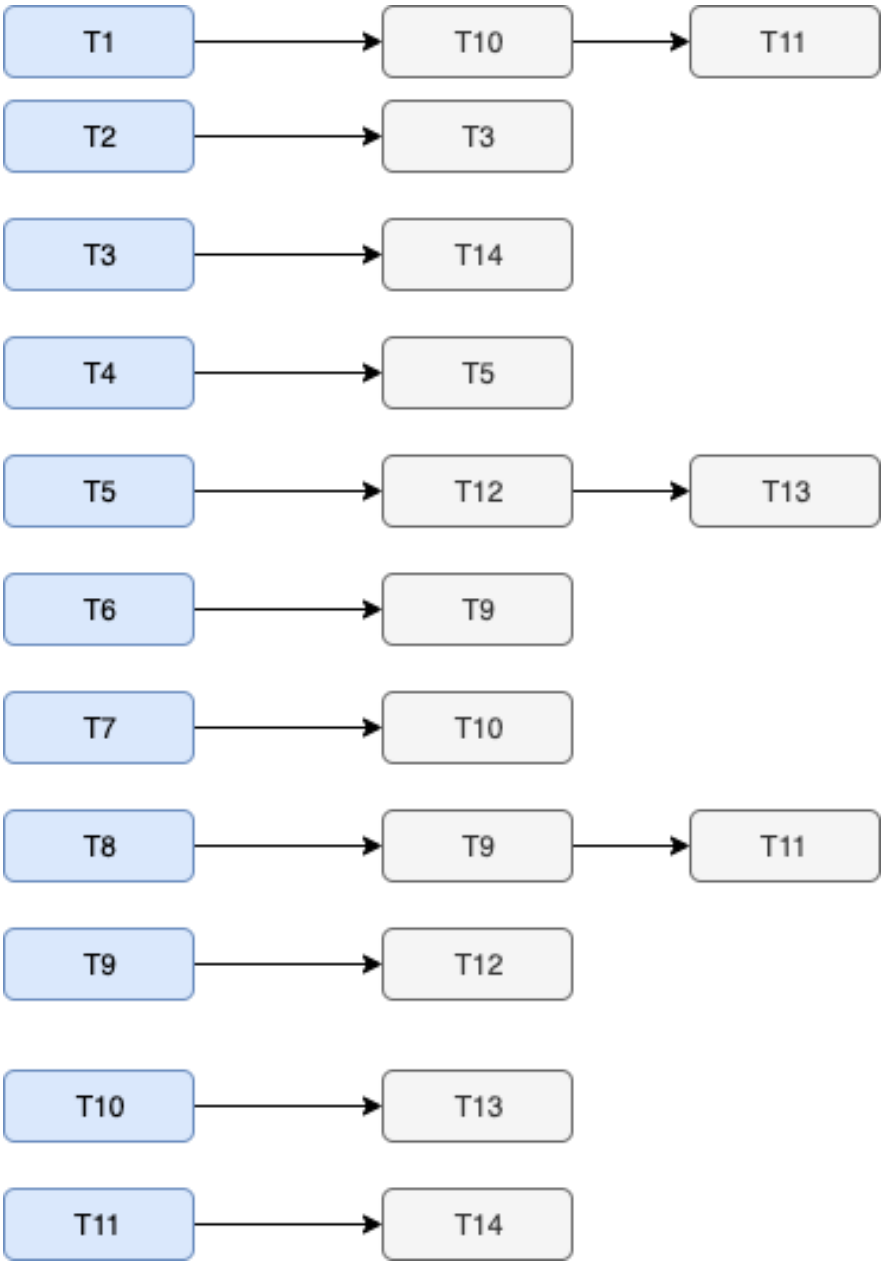


Figure 3: Hash Dependency Table H

As previously mentioned, transactions are added to the hash table only when they depend on other transactions. Moreover, the hash table stores the transactions in an increasing order of their IDs the thing that enhances the search process. In this manner if the malicious transaction was T_3, the algorithm only scans the needed part of the hash table my omitting all transactions with IDs less than 3.

The damage assessment algorithm receives the populated hash table to detect

a list of affected transactions that are related to the previously identified malicious transaction. The damage assessment algorithm results are forwarded to the recovery algorithm that is responsible of running malicious transactions and affected transaction undo and redo processes.

Given that T_6 is the malicious transaction, the damage assessment algorithm skips all transactions with IDs less than 6 due to the presence of the rigorous serializable history. The algorithm accesses the row in the hash table H that belongs to T_6 based on the hash value of its ID. It copies the dependent list of T_6 that contains T_9 to the affected transactions list. At this point, the affected list becomes $\{T_9\}$. Also, in order to cover indirectly affected transactions the algorithm loops over the affected list and repeats the same process until all the affected list items are covered. Following this manner, the algorithm picks the first item in the affected list which is T_9. Then, all the transactions that depend on T_9 are directly moved to the list that collects the affected items. Since T_{12} is the only item in T_9's dependent list, the affected items list is updated to $\{T_9, T_{12}\}$. After picking T_{12}, the algorithm will stop since T_{12} does not have any dependent transactions. Hence, the assessment phase respectively outputs $\{T_6\}$ and $\{T_9, T_{12}\}$ as malicious and affected.

3.8 Algorithm for Database Recovery

Once the assessment phase is done, the recovery algorithm is executed. This algorithm is responsible for restoring the database to its stable phase after receiving the assessment phase results. In order for this to be achieved, all malicious transactions are undone, also all affected transactions are re-executed.

Enhancing the recovery time entails storing the log file into a hash table. Only the needed part of the log file is stored by starting with the malicious transaction ID until reaching its end. The recovery algorithm considers only transactions that occurred after the malicious transaction since earlier ones are neither affected nor malicious. Moreover, access to each log file entry is very fast by getting the transaction ID hash value.

The recovery algorithm starts by removing the effects of the malicious transaction before moving to redo affected ones. In fact for each identified malicious transaction, the recovery algorithm selects its operation stored in the recovery hash table. The details of the operation indicate what should be done to reserve its effect. In case the malicious transaction inserted a record in the database, undoing this operation is done by deleting the inserted record.

On the opposite side for all identified affected transactions, the algorithm retrieves the transaction operation from the log in order to redo it. In case the affected transaction updated a record, the algorithm re-executes this operation after making sure that no malicious transaction's interference exists. The recovery algorithm steps are shown in Algorithm 2.

Algorithm 2 Recovery Algorithm

Input: Affected and malicious transactions and the log file
Output: Consistent database

```
1: Copy the log file into the recovery hash table
2: for t in malicious transactions do
3:    Get the operation of t from the recovery hash table
4:    Undo the operation
5: end for
6: for t' in affected transactions do
7:    Get the operation of t' from the recovery hash table
8:    Redo the operation
9: end for
```

Once the last affected transaction is executed, the algorithm stops. At this stage, the database is back to its consistent state as all malicious effects were removed and the affected transactions were re-executed accordingly.

3.9 Illustration of the Recovery Process

The example showed in this section is a completion of the previous one presented in section 3.7 that ended with $\{T_6\}$ as the malicious transactions list and $\{T_9, T_{12}\}$ as the affected transactions list.

The recovery algorithm selects T_6 which is the malicious items list first and only item. The retrieved operation of T_6 from the recovery table is as follows:

T_6: Insert into Doctors values (15, Justin, Dermatology, 02-04-2005)

Undoing this operation entails deleting the doctor with ID = 15 from the Doctors table. At this stage all the malicious transaction effects of T_6 are removed.

Then, the algorithm goes over all affected transactions in order to redo them. The first affected transaction is T_9 with the following stored operation:

T_9: Insert into Prescriptions values (1, 23, 15, 22-02-2019)

Redoing the retrieved operation is done by inserting a new record within the Prescriptions table for a patient with ID = 23 prescribed by the doctor with ID = 15. However, the doctor with ID = 15 was previously deleted during the removal of T_6 effects. Then, the prescription having the ID = 1 is removed because there is no record for the doctor with ID = 15 within the Doctors table.

The next selected affected transaction is T_{12} having the following operation:

T_{12}: Insert into PrescriptionDetails values (1, 1, 29, 1, 30)

Re-executing this operation is not feasible because it is inserting into Prescrip-

tionDetails a record having a prescription with ID = 1 that was deleted during the previous step. Thus, this record is removed from the PrescriptionDetails table.

When all affected transactions are covered, the database is no longer inconsistent since all malicious transactions were removed while affected ones were if possible re-executed without malicious transaction interference. At this stage, the database can be back online to offer normal service within the health care system.

3.10 Summary

The model presented in this thesis utilizes a single hash table to store all data that is required to achieve the correct assessment of the attack. Also, the hash table access is done in a fast manner by basing the retrieval of the transactions on the ID hash value. Moreover, the hash table scan omits transactions that are obviously not affected and not malicious by starting with the malicious transaction ID. In addition, the recovery algorithm copies into a hash table the only needed part for recovery from the log file. This ensures the enhancement of the recovery process in terms of speed and memory consumption.

Finally, our model requires the utilization of checkpoints in order to control the size of log file. With time passing, the data stored within the log file becomes obsolete. Also if no malicious transactions exist, this data will not be beneficial for the database recovery. Hence instead of having a huge log file with enormous clean transactions, the log file will be flushed at specific checkpoints. This helps in reducing size and decreasing the reading time of the log file.

Chapter Four

Performance Analysis

4.1 Overview

In this section, an analytical and experimental performance analyses of the proposed algorithm are conducted. The analytical study analyses the time and space complexity using big-O notation. Then, the experimental part covers the performance of our algorithm in a simulated environment that is typical to a real system in which the algorithm can be implemented.

As mentioned before, an intrusion detection system is assumed to exist such that any malicious transaction is identified and then received by the process that assesses the damage. Thus, the damage assessment process is initiated as soon as it receives the identified malicious transactions. The transactions dependencies are the basis for the assessment phase that retrieves the transactions dependencies from a dedicated hash table. Moreover, this hash table is used during recovery to store the needed part of the log file by providing fast access and retrieval.

Since we use a rigorously serializable history, all transactions that precede a specific transaction can not be dependent on it. This assumption is very important for performance improvement because all transactions that occurred before the malicious transaction will be omitted by the algorithm.

The intrusion detection system's alarm initiates the damage assessment algorithm by sending the identified malicious transaction. In case several malicious transactions exist, the one with the smallest ID is selected. The malicious transaction ID determines the beginning point of the scan that collects all directly and indirectly affected transactions. When the damage assessment algorithm is finished, its output is forwarded to the recovery algorithm so that the malicious and affected transactions are undone and redone respectively.

During the conducted experiments, multiple transactions IDs will be selected in order to reflect on the performance of the algorithm with respect to scanning distinct log file regions. Also, the experimental study includes a detailed analysis of the memory utilization, which is a feature that our model is optimized to achieve. Moreover, the scalability of our algorithm will be tested on different databases sizes.

The Northwind database is used during the experimental phase with some applied modifications to serve the goal of the study. This database as a template is provided by Microsoft Access or Microsoft SQL Server. Also, SQL Server (Microsoft, 2017) is used to manage the database. Java is the programming language selected (Oracle, 2019). The simulated environment ran on a system with 2.5 GHz CPU, dual-core, and a RAM size of 4 GB.

In section 4.2, the analytical study of the time and space complexity are covered. Then in section 4.3, we present and analyse the performance of damage assessment algorithm that is also compared to other existing solutions. The detailed analysis and comparisons of the recovery algorithm are later discussed in section 4.4. Also, section 4.5 includes the analysis of the used memory. Finally,

the summary of the results is covered in section 4.6.

4.2 Complexity Analysis

The algorithm running time is affected by the size of the input, the type of data structure selected, and their counts. We considered best and worst-case scenarios. The former occurs when only one malicious transaction exists while no other transactions depend on. In other words, there is not any transaction that read the maliciously altered data item. The latter case occurs when the maliciously modified data item is read by all other transactions. For both time and space complexity, n is used to represent the number of transactions that are committed.

In the best-case scenario, the malicious transaction has nothing dependent on it. Hence, nothing is added to the hash table since it stores only dependent transactions. Also, each transaction can be accessed in constant time $O(1)$ based on the hash value of its ID. Thus, the malicious transaction is found to be non-existing within the hash table in constant time. Hence, $O(1)$ is the algorithm running time in this case.

In the worst-case scenario, the malicious transaction has all other transactions dependent on it. Hence, the number of dependent transactions is n. In order to collect all affected transactions, all the n items will be covered; thus, $O(n)$ is the algorithm running time.

In terms of space complexity analysis, each hash table row may store up to n transactions. Hence, each row has a space complexity of $O(n)$. Moreover, since only committed transactions are stored, the number of rows in the table is denoted

as r such that the number of transactions that other transactions depend on is accumulated in the value of r. Therefore, the space complexity of the hash table is $O(rn)$. Finally, if for every transaction there exists only one other transaction that depends on it, the space complexity becomes $O(n^2)$.

4.3 Analysis of the Damage Assessment Algorithm Performance

It is the responsibility of the damage assessment process to output the list of transactions identified as affected based on the generated hash dependency table. In this section, the analysis of the damage assessment algorithm performance collected during the simulation is presented.

For some simulation purposes, the Northwind database was slightly edited. A subset of the original database was selected as follows:

- Categories Table: It stores the categories of the products using 5 columns.
- Customers Table: Using this table 12 columns, the customers related data are stored.
- Employees Table: It stores employees' information who sold products to customers using 18 columns.
- Order_details Table: It uses 7 columns to store placed orders detailed information.
- Order Table: It stores the customers purchased products within 15 columns.
- Products Table: It contains products related data and it has 13 columns.
- Suppliers Table: It contains data of suppliers within 13 columns.

The following simulations assumptions are added to the preceding ones:

1. The maximum number of items that a transaction can access equals the one that is used in the table on which this transaction operated. In other words, a transaction that inserts a new category, may access up to five data items stored in that table.

2. Maintaining the unique identity of transactions is done by adding a Global ID for distinguishing between transactions. Since multiple transactions may be operating on several tables, a primary key may be redundant for multiple transactions. Hence, using the Global ID is very crucial. In case a transaction has an ID = 1, it would not be the case that another transaction would have the same ID which is 1.

The damage assessment algorithm performance is affected by the scanned transactions number. We analyzed how the algorithm running time changes when varying the ID belonging to the transaction identified as malicious. In fact, when the malicious transaction ID selected is the smallest, the amount of the covered transactions is the largest.

Figure 4 shows how the damage assessment algorithm running time changes while varying the attacking entity ID. The attacker ID is the identified malicious transaction ID sent to the assessment phase. Figure 4 shows that the increase in the ID of the attacker results in a decrease in the time needed to complete the assessment phase. Thus, our model is accurately operating because with increasing the ID of the attacking entity, the scanned transactions are less; therefore, the performance is improved.

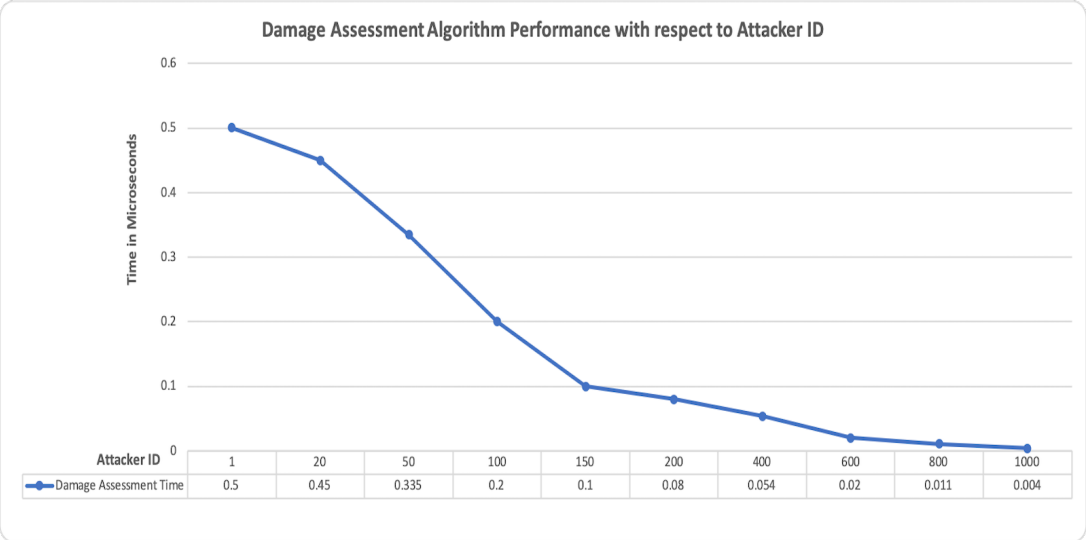


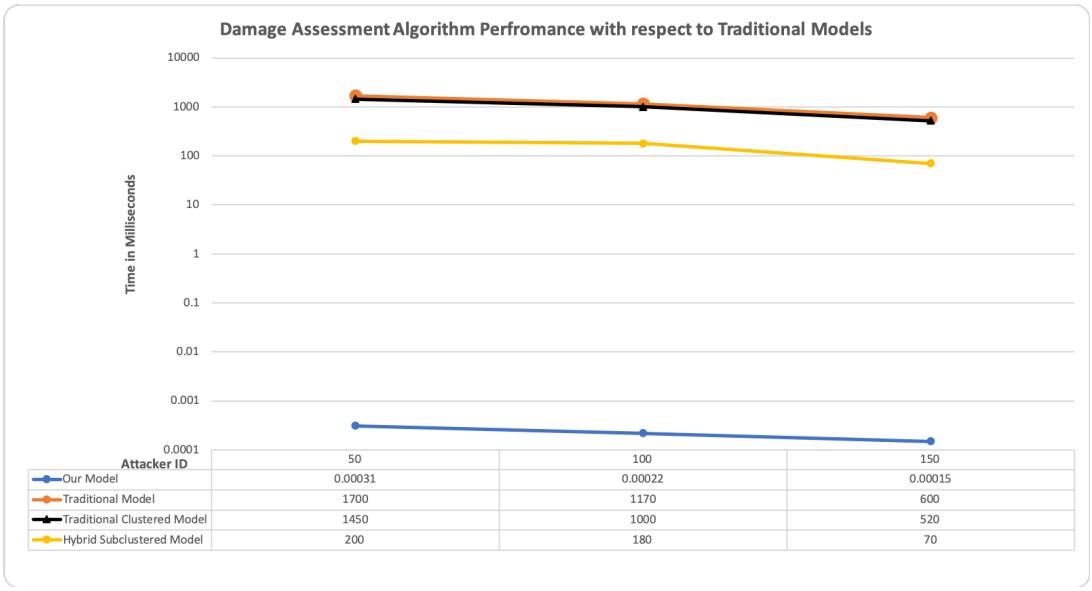
Figure 4: Damage Assessment Algorithm Execution Period w.r.t the ID of the Attacking Entity

The Northwind database is filled with 1080 entries during this experiment. Also, each transaction may access up to 18 data items as the Employee table has 18 columns.

A transaction T_j is not dependent on T_i if $j < i$. Thus, all transactions that happened before the malicious transaction are skipped since they can not be affected. The advantages of this approach can be seen in Figure 4. When the smallest transaction ID is selected, the running time needed to execute the algorithm performing the assessment is at its highest with 0.5 µs. This happens because the number of scanned transactions is 1079. When the highest transaction ID is selected, the running time reaches 0.004 µs as it scans only 80 transactions. Hence, our assessment methodology is proved to be efficient by not wasting time in unneeded scans of the unaffected transactions.

We compared the running time of our damage assessment algorithm to other models referred to in (Haraty and Zeitunlian, 2007). These models are based on applying some clustering techniques to the log file. The model applying a

traditional technique for recovery is presented in the work of (Bai and Liu, 2009). Since the experiments in (Haraty and Zeitunlian, 2007) were conducted using a smaller number of records, a customized version of the database containing a reduced number of 200 records was prepared. Figure 5 samples the collected comparisons.



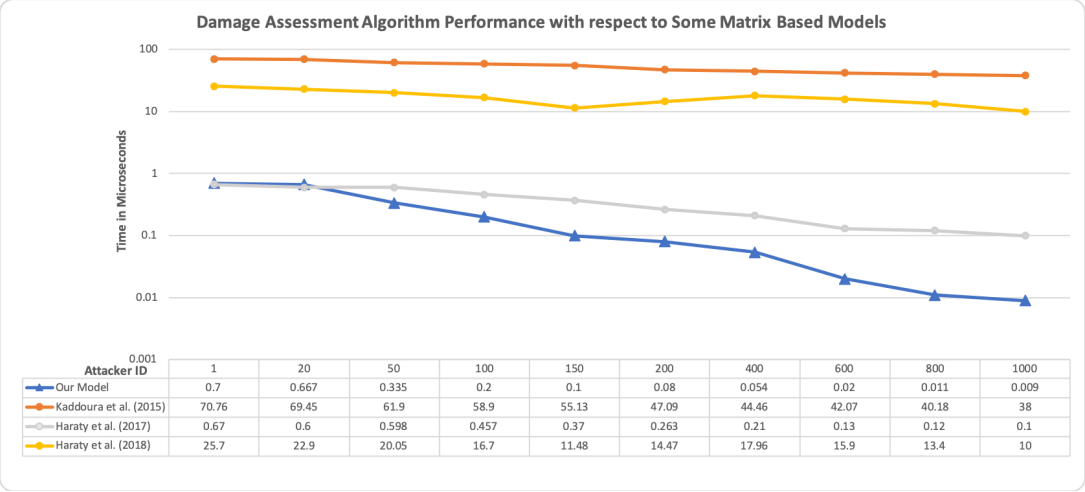
Attacker ID	50	100	150
Our Model	0.00031	0.00022	0.00015
Traditional Model	1700	1170	600
Traditional Clustered Model	1450	1000	520
Hybrid Subclustered Model	200	180	70

Figure 5: Damage Assessment Algorithm Execution Period w.r.t Traditional Models

After looking at the running time produced by our damage assessment algorithm, we can confirm that it has the best performance. For instance, with attacker ID equals 50, our algorithm takes 0.000 31 ms which is at least 645,161 times faster than the traditional hybrid sub-cluster model that is the best performing among the models discussed in (Haraty and Zeitunlian, 2007). When the malicious transaction ID reaches 150, our algorithm is 466,666 times faster than the hybrid sub-cluster model. Also, we compared our assessment running time with the slowest of the three models, which is the traditional model. We found that our algorithm is at least 5,483,870 times faster with attacker ID equals 50 and 4,000,000 times faster with attacker ID equals 150.

The performance enhancement of our algorithm is mainly due to the correct selection of the data structure storing transactions dependencies. Since our hash dependency table stores the only needed information to correctly output the affected transactions. Also, the hash dependency table provides fast access based on the hash value of the transaction ID. Finally, our algorithm differs from traditional approaches by omitting portions of the log file that contain unaffected transactions.

In another experiment, we compared our damage assessment algorithm with more recent matrix based approaches including (Haraty et al., 2017, 2018; Kaddoura et al., 2015). In order to match their experimental requirements, we adjusted the Northwind database to contain 2000 data items. Also, 1200 transactions were executed. Figure 6 shows the findings of this experiment.



Attacker ID	1	20	50	100	150	200	400	600	800	1000
Our Model	0.7	0.667	0.335	0.2	0.1	0.08	0.054	0.02	0.011	0.009
Kaddoura et al. (2015)	70.76	69.45	61.9	58.9	55.13	47.09	44.46	42.07	40.18	38
Haraty et al. (2017)	0.67	0.6	0.598	0.457	0.37	0.263	0.21	0.13	0.12	0.1
Haraty et al. (2018)	25.7	22.9	20.05	16.7	11.48	14.47	17.96	15.9	13.4	10

Figure 6: Damage Assessment Algorithm Execution Period w.r.t Some Matrix Based Models

As shown in Figure 6, the slowest algorithm is the one of (Kaddoura et al., 2015). The algorithm starts with a running time of 70.76 µs when the malicious transaction ID is equal to 1. Also, when the attacker ID increases, the algorithm has the slowest running time that ends with 38 µs. The slow performance of this

algorithm is due to the utilization of string representation of the transactions within the dependency matrix that requires a long time in comparison.

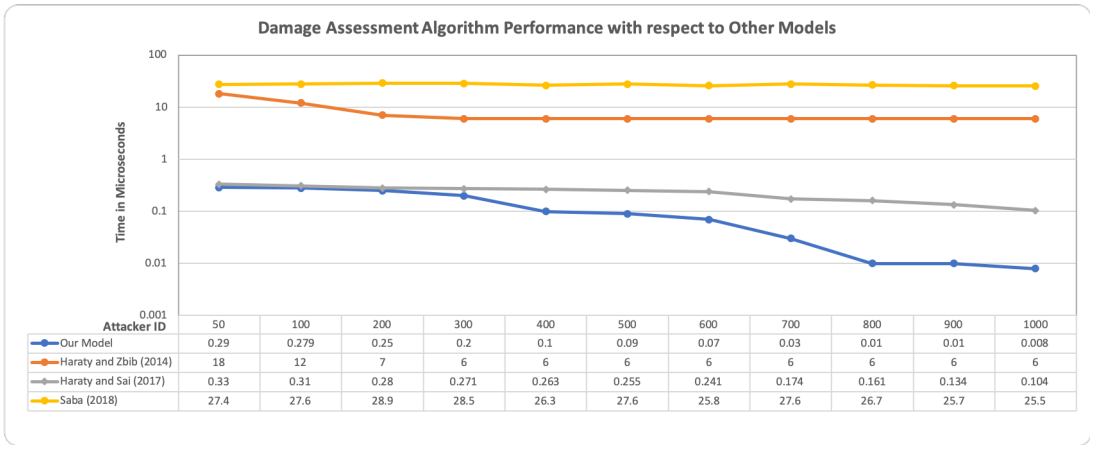
On the contrary, Haraty et al. (2018) improved the assessment process of Kaddoura et al. (2015) by changing string representation to integer. Also the authors of Haraty et al. (2018), augmented the matrix with additional columns that improved the search time. These improvements are clearly visible in Figure 6, since Haraty et al. (2018) algorithm has a better running time than the one of Kaddoura et al. (2015) such that the factor had a value of 2.7 when the smallest ID is selected. Then, the factor increased to 3.8 when the highest ID is selected.

Our damage assessment algorithm has the best running time. Although the work Haraty et al. (2017) has a slightly better performance with a ratio of 1.04, our algorithm outperforms the work of Haraty et al. (2017) directly when the attacker ID exceeds 20. Also, the better performance of our algorithm is consistent until the highest attacker ID is selected at which the ratio reaches 11. The work done in Haraty et al. (2017) utilizes an indexed structure of the linked list to allow the algorithm to be fast in accessing the intended transactions. Also, our algorithm provides a fast accessible allocation of transactions by basing the retrieval on the hash value of their IDs. However, the main difference between these algorithms is manifested within the condition needed to add the transaction to the structure storing the transaction dependencies. Our algorithm stores only dependent transactions rather than reserving a row for all committed transactions as done in Haraty et al. (2017).

When compared to the slowest algorithm of Kaddoura et al. (2015), our algorithm has a better performance by a ratio factor of around 100 when the smallest

and biggest attacker IDs are selected.

Highlighting how our data structure is useful can done by analyzing its performance to other models that utilized linked lists, graphs, and two dimensional arrays (Haraty and Sai, 2017; Haraty and Zbib, 2014; Saba, 2018). The final experiment was conducted using 1080 entries in the database. The results of the comparison are illustrated in Figure 7.



Attacker ID	50	100	200	300	400	500	600	700	800	900	1000
Our Model	0.29	0.279	0.25	0.2	0.1	0.09	0.07	0.03	0.01	0.01	0.008
Haraty and Zbib (2014)	18	12	7	6	6	6	6	6	6	6	6
Haraty and Sai (2017)	0.33	0.31	0.28	0.271	0.263	0.255	0.241	0.174	0.161	0.134	0.104
Saba (2018)	27.4	27.6	28.9	28.5	26.3	27.6	25.8	27.6	26.7	25.7	25.5

Figure 7: Damage Assessment Algorithm Execution Period w.r.t Recent Algorithms

After analyzing the results shown in Figure 7, we can conclude that our hash based assessment algorithm outperforms all other algorithms. The running time reaches 0.008 µs in the best-case and 0.29 µs in the worst-case. Our damage assessment algorithm takes advantage of the fast retrieval of the transactions provided by the hash dependency table. Also, the affected list attached to each transaction can be accessed by a one-hit to the hash table. For instance, if the malicious transaction is T_2 it can be accessed based on the ID hash value without the need to loop over the table to locate its index.

On the contrary in the work of Haraty and Sai (2017), the transaction dependencies are stored in a matrix of linked lists. In case of an attack, the algorithm

checks the whole list of transactions stored in the matrix. Moreover for each identified malicious transaction, its related row is scanned in order to identify all affected transactions. These factors affected the running time of the assessment phase the thing that is demonstrated in Figure 7 showing a slower performance than our algorithm with a ratio factor that reaches 13 when the attacker ID is 1000.

The algorithms of Haraty and Zbib (2014) and Saba (2018) have the slowest running time. In the work of Haraty and Zbib (2014), the authors utilize a primary and secondary arrays to maintain the dependencies of transactions. The traversal of both arrays to locate all affected transactions is the obvious drawback of this work. The slow performance of this algorithm is affected by the time consumed in array traversal that is 62 times slower than our algorithm when the attacker ID is 50. Also when the ID selected is the highest, our algorithm running time becomes 750 times faster than the algorithm of Haraty and Zbib (2014). The work of Saba (2018) utilizes graphs for storing transactions dependencies. Although the advantages of using graphs can be manifested in better scalability and damage isolation, the time taken for graphs construction and coverage is outperformed by our selected hash table offering fast allocation, access, and search. These advantages help in reaching a better running with a ratio of around 94 when the smallest attacker ID is selected.

To sum up, the above discussed experiments compared our damage assessment algorithm to traditional and more recent models by using different snapshots of Northwind database. The detailed analysis of the obtained results proves that our algorithm outperforms the other models by using a fast accessible hash table

that stores the only needed information for maintaining a correct operation.

4.4 Analysis of the Recovery Algorithm Performance

A customized database version was used during the recovery algorithm experimental work. It is expected that with a higher number of transactions to be recovered, the time spent in recovery is longer. The time spent in recovery varies with the changes applied to the number of transactions requiring recovery. 200 entries stored in the database were used during this experiment where the maximum number of the accessed data items is 18. Figure 8 highlights the obtained simulation results of the recovery algorithm.

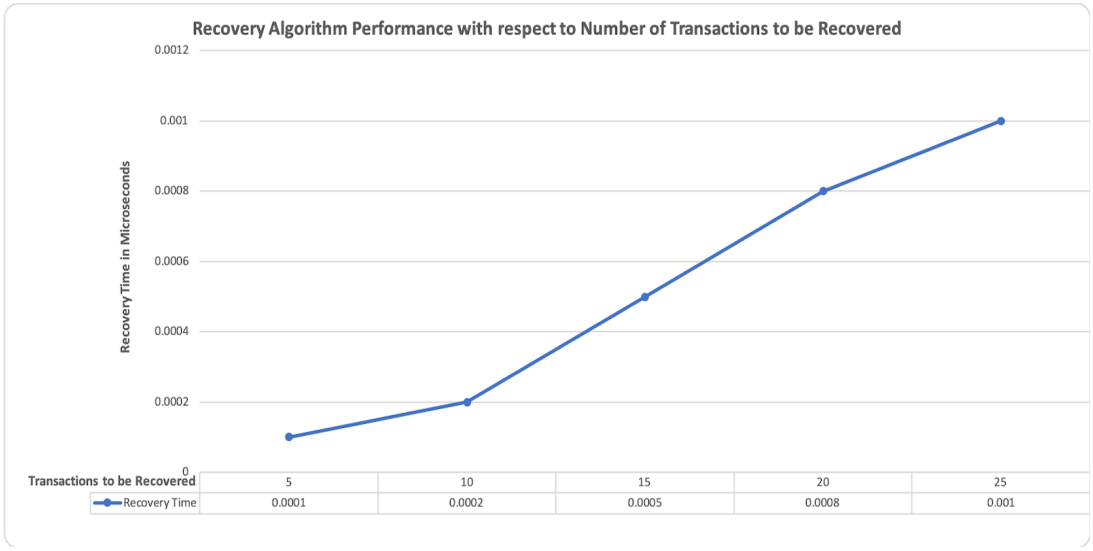
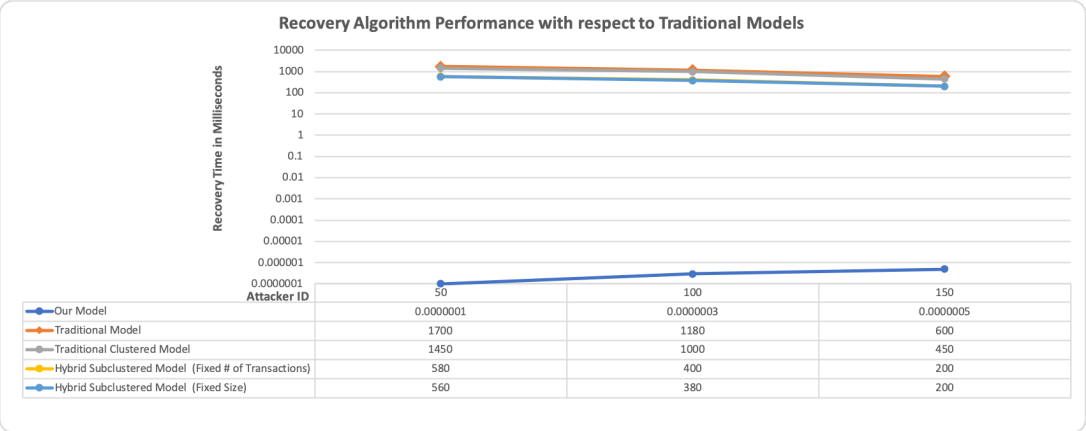


Figure 8: Recovery Algorithm Execution Period w.r.t Recovered Transactions Numbers

Malicious and affected transactions are included within the selected transactions to be recovered. Figure 8 proves that while increasing the number of recovered transactions, the execution time needed for recovery also increases. However, the ratio of the recovery running time is acceptable as it increases from 0.0001 µs up to 0.001 µs while augmenting the number of recovered transactions by 5 folds.

In another experiment that is shown in Figure 9, our recovery algorithm execution time was compared to some clustered and non-clustered models discussed in (Haraty and Zeitunlian, 2007).



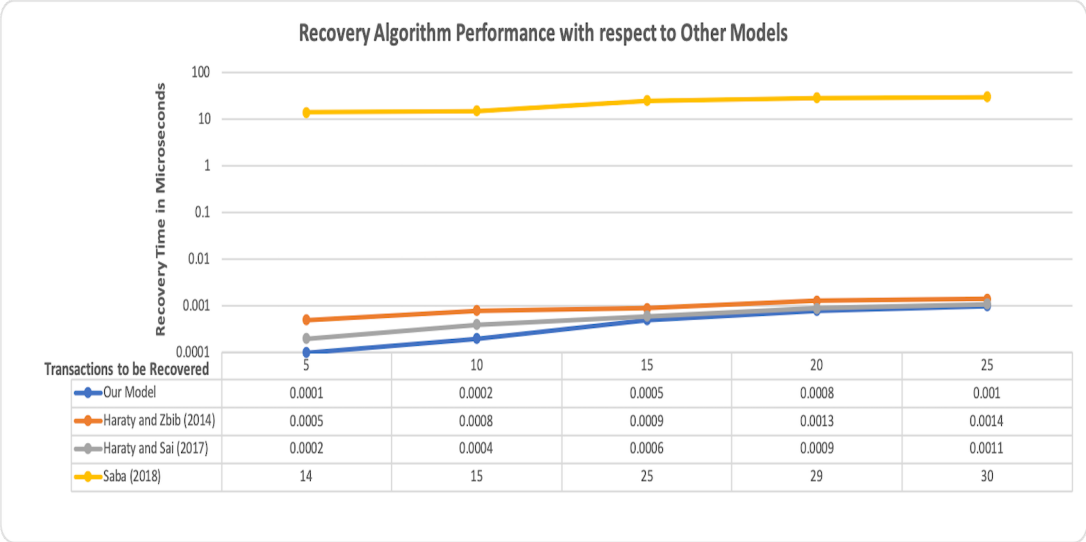
Attacker ID	50	100	150
Our Model	0.0000001	0.0000003	0.0000005
Traditional Model	1700	1180	600
Traditional Clustered Model	1450	1000	450
Hybrid Subclustered Model (Fixed # of Transactions)	580	400	200
Hybrid Subclustered Model (Fixed Size)	560	380	200

Figure 9: Recovery Algorithm Execution Period w.r.t Traditional Models

The results of Figure 9 proves that the hash based algorithm of this thesis has the best running time among all other models. For instance when 50 is the selected ID, the hash based algorithm improves the running time of the model applying a traditional technique that has the worst performance by a factor of 17,000,000,000. Also, this factor reaches 5,600,000,000 when compared to the best performing clustered model that is based on a predefined cluster size. These performance improvements offered by our model are guaranteed by decreasing the assessment phase time that precedes the recovery. By storing the transactions dependencies in a hash table, the assessment time is greatly enhanced. Moreover, the recovery time is reduced by storing the only needed part of the log file in a dedicated hash table that provides faster access and retrieval operations. Instead of utilizing huge I/O for accessing the log file during a long-time period, we store from the log file the operations that occurred after the detected attack. Due to the existence of the rigorously serializable history, transactions that are older

than the malicious transactions can not be affected.

In the final experiment, we compared our recovery algorithm with recent matrix-based and graph-based models (Haraty and Sai, 2017; Haraty and Zbib, 2014; Saba, 2018). Figure 10 shows the final experiment collected data.



Transactions to be Recovered	5	10	15	20	25
Our Model	0.0001	0.0002	0.0005	0.0008	0.001
Haraty and Zbib (2014)	0.0005	0.0008	0.0009	0.0013	0.0014
Haraty and Sai (2017)	0.0002	0.0004	0.0006	0.0009	0.0011
Saba (2018)	14	15	25	29	30

Figure 10: Recovery Algorithm Execution Period w.r.t Recent Algorithms

After analyzing the results of Figure 10, we can conclude that our algorithm has the fastest recovery time among all other algorithms. It is almost twice time better than matrix-based algorithm of Haraty and Sai (2017) that offered an enhanced recovery among the remaining two models of Haraty and Zbib (2014) and Saba (2018). Although the work of Haraty and Sai (2017) improves the recovery time by saving the needed part of the log in an array, our solution maintains a better performance by providing faster retrieval and search time offered by the selected hash table.

Also, our algorithm is five times faster than the other matrix-based algorithm of Haraty and Zbib (2014). This is due to the fact that the work Haraty and Zbib (2014) reads the log file data before storing it into a dedicated array.

Finally, the graph-based algorithm of Saba (2018) has the slowest recovery time than all other models. The recovery time takes the longest time because it traverses the received subgraph twice. In the first time, it starts with the pointer having the least ID until the last node is covered, then it starts a rollback back process from the last node reached until arriving at the first node that is not malicious. This recovery procedure takes an extensive amount of time when compared to our model that clearly enhances both damage assessment and recovery procedures, and thus it improves database availability.

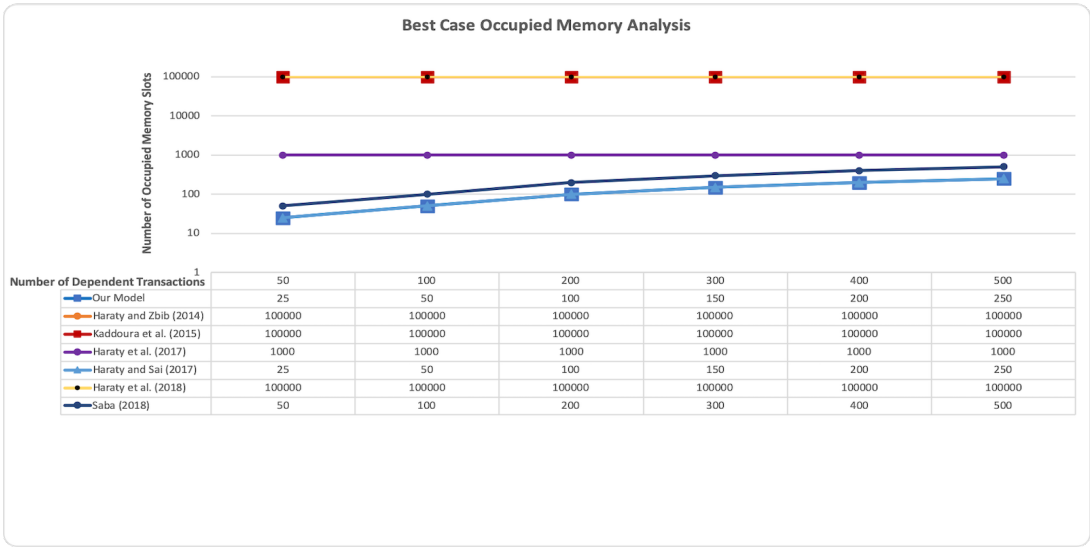
4.5 Analysis of the Memory Utilization Performance

In addition to the optimal usage of assessment and recovery time, our solution offers efficient utilization of memory. The low consumption of memory provides high scalability feature that suits the growing size of medical databases. The algorithm's memory consumption was compared to other existing models (Haraty et al., 2017, 2018; Haraty and Sai, 2017; Haraty and Zbib, 2014; Kaddoura et al., 2015; Saba, 2018). The best and worst case memory utilization analyses are conducted in a way that reassembles the preceding parts of the report.

Matrices of 2-arrays are utilized in the work of (Haraty et al., 2018; Haraty and Zbib, 2014; Kaddoura et al., 2015). These data dependencies algorithms store transactions dependencies in a matrix in which data items represent the columns and transactions represent the rows. During the memory utilization experiments, 1000 transactions are assumed to operate on a database of 100 data items. In this case, 100,000 entries will be part of the dependency matrices of (Haraty et al., 2018; Haraty and Zbib, 2014; Kaddoura et al., 2015) that are

related to the data items operated on along with the committed transactions number. Also, the algorithm of Haraty et al. (2017) allocates 1000 records in the dependency matrix as it reserves a row for each committed transaction. In sum, this algorithm uses an unchanged amount of used memory that relates to the committed transaction number rather than the dependent transactions number. On the contrary, our algorithm along with the work of Haraty and Sai (2017) and Saba (2018) are not related to the committed transactions number or the data items number.

In the first experiment, we compared the best-case scenario memory consumption of our solution with the work of (Haraty et al., 2017, 2018; Haraty and Sai, 2017; Haraty and Zbib, 2014; Kaddoura et al., 2015; Saba, 2018). The results are shown in Figure 11.



Number of Dependent Transactions	50	100	200	300	400	500
Our Model	25	50	100	150	200	250
Haraty and Zbib (2014)	100000	100000	100000	100000	100000	100000
Kaddoura et al. (2015)	100000	100000	100000	100000	100000	100000
Haraty et al. (2017)	1000	1000	1000	1000	1000	1000
Haraty and Sai (2017)	25	50	100	150	200	250
Haraty et al. (2018)	100000	100000	100000	100000	100000	100000
Saba (2018)	50	100	200	300	400	500

Figure 11: Occupied Memory Analysis in the Best-Case

The best-case scenario happens when each transaction has on single transaction on which it depends. This ensures the least possible memory usage. Figure 11 shows that our proposed solution utilizes the least amount of memory. Similarly is the case for the algorithm of Haraty and Sai (2017). Since both models

add only dependent transactions to the data structure that stores the transactions dependencies, the amount of memory that is occupied equals the dependent transactions number divided by two. When the dependent transactions number reaches 50, our utilized hash table will contain 25 rows in which each transaction has only one item in the transactions dependent list. Thus, the number of occupied memory slots is 25 because each transaction has only one dependent transaction. The same case applies for Haraty and Sai (2017) that utilizes a matrix of the linked list of length 25 such that each transaction has only one transaction on which it depends.

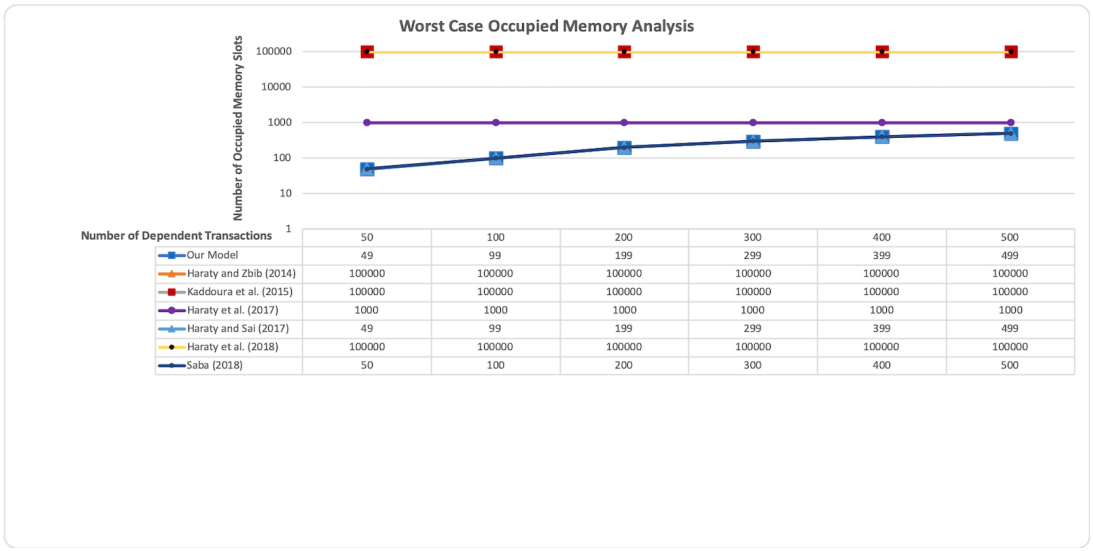
Moreover, our model consumes a number of memory slots that is two times better than the algorithm of Saba (2018). The occupied memory slots of Saba (2018) reaches 50 with 50 dependent transactions. Since the model of Saba (2018) utilizes a graph to store the dependencies, each transaction has one memory slot per each connection in the graph. Thus, the amount of memory occupied is the same the dependent transactions count.

Haraty et al. (2017) proposed algorithm consumes an amount of memory slots that is 40 times bigger than our model. The dependency matrix of Haraty et al. (2017) adds all the transactions that are committed. Hence, the amount of used memory is 1000 since this model is affected by the committed transactions count. By following this approach, Haraty et al. (2017) algorithm is efficient in terms of memory utilization.

All data dependency models of (Haraty et al., 2018; Haraty and Zbib, 2014; Kaddoura et al., 2015) are related to data items count along with the count of transactions operated. The consumed amount of memory of these algorithms is

4,000 times bigger than our model as shown in Figure 11. Since The dependency matrices of (Haraty et al., 2018; Haraty and Zbib, 2014; Kaddoura et al., 2015) represent all data items and the transactions in the database, the number of occupied memory slots equals 100,000 (1000 transactions * 100 data items). Hence, these algorithms (Haraty et al., 2018; Haraty and Zbib, 2014; Kaddoura et al., 2015) have the worst memory consumption.

In the second experiment, we compared the worst-case scenario memory utilization of our model along with the ones of (Haraty et al., 2017, 2018; Haraty and Sai, 2017; Haraty and Zbib, 2014; Kaddoura et al., 2015; Saba, 2018). The experimental results are shown in Figure 12.



Number of Dependent Transactions	50	100	200	300	400	500
Our Model	49	99	199	299	399	499
Haraty and Zbib (2014)	100000	100000	100000	100000	100000	100000
Kaddoura et al. (2015)	100000	100000	100000	100000	100000	100000
Haraty et al. (2017)	1000	1000	1000	1000	1000	1000
Haraty and Sai (2017)	49	99	199	299	399	499
Haraty et al. (2018)	100000	100000	100000	100000	100000	100000
Saba (2018)	50	100	200	300	400	500

Figure 12: Occupied Memory Analysis in the Worst-Case

The worst-case memory consumption happens when the most amount of memory is consumed. This occurs when one transaction has all other transactions dependent on it. For instance, if the database contains 60 transactions, a transaction T_1 has 59 other dependent transactions.

As the best-case scenario, our model uses the least amount of memory. Also, the algorithm of Haraty and Sai (2017) has a similar performance. Both algo-

rithms store only dependent transactions rather than storing all committed ones. Thus, when the database has 50 dependent transactions both algorithms consume only 49 memory slots since one transaction is dependent on the remaining 49 ones.

The work of Saba (2018) has the same memory utilization of the best-case scenario. Since each node can have only one connection, the amount of occupied memory slots is always the same as the dependent transactions number.

In the work of Haraty et al. (2017), the matrix reserves a row for each committed transaction. In this case, the consumed memory slots number is the same as the one of committed transactions which is 1000. In sum, the memory consumed by the algorithm of Haraty et al. (2017) is still high similar to the best-scenario since the algorithm is connected to the number of transactions that are committed.

As seen in Figure 12, the models of (Haraty et al., 2018; Haraty and Zbib, 2014; Kaddoura et al., 2015) utilize a number of memory slots that is around 2041 times bigger than our model when there exist 50 dependent transactions. Since the algorithms of (Haraty et al., 2018; Haraty and Zbib, 2014; Kaddoura et al., 2015) depend on the of data items and transactions counts, the memory utilization stays high by storing 100,000 entries in the dependency matrix.

The above-mentioned experiments prove the high efficient utilization of memory by our suggested model. This suits well high scalable databases. Moreover since our model improves the time spent in assessment and recovery, the whole algorithm performance is enhanced the thing that helps in reducing the database offline time; thus, database availability is increased.

4.6 Summary

The hash based model of this thesis was analyzed in terms of execution time, correctness, and memory utilization. The damage assessment algorithm is proven to provide correct and efficient execution. Since the scan omits unaffected transactions, the running time of the algorithm improved greatly when the scanned number of transactions is reduced. Also, the hash dependency table outperforms other previously utilized data structures including graphs, linked lists, and arrays.

The recovery algorithm maintains the best performance among recent and traditional models. The performance improvement is guaranteed by storing in the hash table only the needed log file portions. Hence, the period spent in recovering the database was greatly improved.

The final part of the experimental analysis focuses on the optimized feature of our model which is the reduction of memory utilization. The detailed analysis proves how our model utilises less resources by adjusting the size of the hash table to store the only needed information for correct operation.

Our model provides a scalable damage assessment and recovery algorithm that suits medical databases with varying sizes. Also by improving the damage assessment and recovery time, the database can become available for normal operation in a bearable amount of time.

Chapter Five

Conclusion

The major security pillars adopted in information warfare contain prevention, detection and correction. The prevention methods pose some barriers carried by firewalls and antivirus to protect the system or reduce the possibility of breaches, however the highly secure medical databases are still attacked. Then, the methods used for detection are carried by intrusion detection systems that identify the presence of an attack and its source. However, the attacks may stay unnoticed for a while causing the damage to sweep to large area of the database. Here comes the role the reaction phase to complement the work of the detection phase. The reaction methods act as the last defense line by providing a recovery plan that assesses the damage and restores the stable phase of the database. The reaction phase techniques should be quick, efficient, and optimally reduce the database offline time.

As hackers continue to develop new methods to penetrate into systems, the need to continually improve the solutions used for database recovery is very essential. The reaction phase of information warfare is highly focused within this thesis that proposes a suitable solution for recovering a database following an attack. Our suggested model uses a new hash based technique that is efficient in terms

of execution time and resources utilized. By improving the damage assessment and recovery phases, we achieve our intend goal of reducing the medical database offline time; thus, improving its availability. We implemented our algorithm and compared its performance with other existing models. The experimental results proved the performance improvements offered by our model in terms of running time, efficiency, and memory utilization when compared to other models.

As for future work, more effort can be done to apply this model in distributed databases. When distributed databases are utilized, the algorithm should take into consideration that data is distributed and replicated in many cites. This poses new challenges that should be solved in future versions of the algorithm. For instance our model can play a major role within a medical blockchain in which the distributed decentralized system connects a series of shared healthcare records stored within publicly accessible databases. Also, blockchain technology appliance within healthcare system is gaining a huge research attention due to its high benefits suiting the high demands of the current medical sectors. Moreover, the application of our hash based technique in an edge computing environment provides numerous opportunities for further research efforts. Within the medical field, edge computing can be applied to push some of the computations to areas closer to the client while enhancing the services and reducing the consumed bandwidth.

References

Ammann, P., Jajodia, S., and Peng Liu (2002). Recovery from malicious transactions. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1167–1185.

Bai, K. and Liu, P. (2009). A data damage tracking quarantine and recovery (dtqr) scheme for mission-critical database systems. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 720–731, New York, NY, USA. ACM.

Bendovschi, A. (2015). Cyber-attacks–trends, patterns and security countermeasures. *Procedia Economics and Finance*, 28:24–31.

Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1986). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bidgoli, H. (2006). *Handbook of information security*. John Wiley Sons.

Boros, Z. and Szaz, A. (2008). Reflexivity, transitivity, symmetry and anti-symmetry of the intersection convolution of relations. *Rostock. Math. Kolloq.*, pages 55–62.

Breitbart, Y., Georgakopoulos, D., Rusinkiewicz, M., and Silberschatz, A. (1991). On rigorous transaction scheduling. *IEEE Transactions on Software Engineering*, 17(9):954–960.

Chakraborty, A., Majumdar, A. K., and Sural, S. (2010). A column dependency-based approach for static and dynamic recovery of databases from malicious transactions. *International Journal of Information Security*, 9(1):51–67.

Chen, C., Liu, Q., Liu, Y., and Shen, G. (2012). *A Recovery Approach for Real-Time Database Based on Transaction Fusion*, pages 473–479. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chrysanthis, P. and Ramamritham, K. (1998). Correctness criteria and concurrency control.

Endsley, M. and Jones, W. M. (1997). Situation awareness information dominance & information warfare. Technical report, LOGICON TECHNICAL SERVICES INC DAYTON OH.

Fathy, M., Azer, M., Bahgat, M., and Yehia, A. (2013). Security access control research trends. In *6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 1–6.

Fu, G., Zhu, H., Feng, Y., Zhu, Y., Shi, J., Chen, M., and Wang, X. (2008). Fine grained transaction log for data recovery in database systems. pages 123 – 131.

Haraty, R., Kaddoura, S., and Zekri, A. (2017). Transaction dependency based approach for database damage assessment using a matrix. *International Journal on Semantic Web and Information Systems*, 13:74–86.

Haraty, R. and Zeitunlian, A. (2007). Damage assessment and recovery from malicious transactions using data dependency for defensive information warfare. *ISESCO Science and Technology Vision*, 3(4):43–50.

Haraty, R. A., Kaddoura, S., and Zekri, A. S. (2018). Recovery of business intelligence systems: Towards guaranteed continuity of patient centric healthcare systems through a matrix-based

recovery approach. *Telematics and Informatics*, 35(4):801 – 814.

Haraty, R. A. and Mohsen, H. (2014). Efficient damage assessment and recovery using fast mapping. In *Proceedings of the International Conference of Advanced Computer Science and Information Technology (ACSIT14)*.

Haraty, R. A. and Sai, M. E. (2017). Information warfare: a lightweight matrix-based approach for database recovery. *Knowledge and Information Systems*, 50(1):287–313.

Haraty, R. A. and Zbib, M. (2014). A matrix-based damage assessment and recovery algorithm. In *2014 14th International Conference on Innovations for Community Services (I4CS)*, pages 22–27.

Haraty, R. A., Zbib, M., and Masud, M. (2016). Data damage assessment and recovery algorithm from malicious attacks in healthcare data sharing systems. *Peer-to-Peer Networking and Applications*, 9(5):812–823.

Hutchinson, W. (2006). Information warfare and deception. *Informing Science*, 9.

Hutchinson, W. and Warren, M. (2001). Information warfare: Corporate attack and defence in a digital world. *ECU Publications*.

Jing, X., Yan, Z., and Pedrycz, W. (2019). Security data collection and data analytics in the internet: A survey. *IEEE Communications Surveys Tutorials*, 21(1):586–618.

Kaddoura, S., Haraty, R., Zekri, A., and Masud, M. (2015). Tracking and repairing damaged healthcare databases using the matrix. *International Journal of Distributed Sensor Networks*, 2015.

Khochare, N., Chalurkar, S., Kakade, S., and Meshramm, B. (2011). Survey on sql injection attacks and their countermeasures. *International Journal of Computational Engineering & Management (IJCEM)*, 14.

Kim, T., Wang, X., Zeldovich, N., Kaashoek, M. F., et al. (2010). Intrusion recovery using selective re-execution. In *OSDI*, pages 89–104.

Kumar, V. and Son, S. H. (1998). *Database recovery*. Springer.

Kurra, K., Panda, B., Li, W., and Hu, Y. (2015). An agent based approach to perform damage assessment and recovery efficiently after a cyber attack to ensure e-government database security. In *2015 48th Hawaii International Conference on System Sciences*, pages 2272–2279.

Lala, C. and Panda, B. (2001). Evaluating damage from cyber attacks: a model and analysis. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 31(4):300–310.

LaPiedra, J. (2002). The information security process: Prevention, detection and response. *Style (DeKalb, IL)*.

Libicki, M. C. (1995). What is information warfare? Technical report, NATIONAL DEFENSE UNIV WASHINGTON DC INST FOR NATIONAL STRATEGIC STUDIES.

Liu, P. and Jajodia, S. (2002). *Trusted Recovery Models*, pages 27–38. Springer US, Boston, MA.

Liu, P. and Yu, M. (2011). Damage assessment and repair in attack resilient distributed database systems. *Computer Standards Interfaces*, 33(1):96 – 107. Special Issue: Secure Semantic Web.

Panda, B. and Asharful Haque, K. (2002). Extended data dependency approach: a robust way of rebuilding database. pages 446–452.

Panda, B. and Tripathy, S. (2000). Data dependency based logging for defensive information warfare. In *Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 1*, SAC '00, pages 361–365, New York, NY, USA. ACM.

Panda, B. and Yalamanchili, R. (2001). Transaction fusion in the wake of information warfare. In *Proceedings of the 2001 ACM Symposium on Applied Computing (SAC), March 11-14, 2001, Las Vegas, NV, USA*, pages 242–247.

Panda, B. and Zhou, J. (2003). Database damage assessment using a matrix based approach: An intrusion response system. In *Seventh International Database Engineering and Applications Symposium, 2003. Proceedings.*, pages 336–341. IEEE.

Ragothaman, P. and Panda, B. (2003). *Analyzing Transaction Logs for Effective Damage Assessment*, pages 89–101. Springer US, Boston, MA.

Saba, R. (2018). *Information reconciliation through agent controlled graph model.(c2018)*. PhD thesis, Lebanese American University.

Sobhan, R. and Panda, B. (2002). *Reorganization of the Database Log for Information Warfare Data Recovery*, pages 121–134. Springer US, Boston, MA.

Sumathi, S. and Esakkirajan, S. (2010). *Fundamentals of Relational Database Management Systems*. Springer Publishing Company, Incorporated, 1st edition.

Weikum, G. and Vossen, G. (2001). *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Xia, X., Ji, Q., and Yang, A. (2012). Research on a new architecture of self-healing database system. In *2012 Fifth International Symposium on Computational Intelligence and Design*, volume 2, pages 474–477.

Xie, M., Zhu, H., Feng, Y., and Hu, G. (2008). Tracking and repairing damaged databases using before image table. In *2008 Japan-China Joint Workshop on Frontier of Computer Science and Technology*, pages 36–41.

Yanjun Zuo and Panda, B. (2004). Fuzzy dependency and its applications in damage assessment and recovery. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 350–357.

Zarif, O. E. and Haraty, R. A. (2019). Towards information preservation in healthcare systems. In *Innovation in Health Informatics: a Smart Healthcare Primer*. Elsevier.

Zheng, J., Qin, X., and Sun, J. (2007). Data dependency based recovery approaches in survival database systems. In Shi, Y., van Albada, G. D., Dongarra, J., and Sloot, P. M. A., editors, *Computational Science – ICCS 2007*, pages 1131–1138, Berlin, Heidelberg. Springer Berlin Heidelberg.

Zuo, Y. and Panda, B. (2004). Damage discovery in distributed database systems. In Farkas, C. and Samarati, P., editors, *Research Directions in Data and Applications Security XVIII*, pages 111–123, Boston, MA. Springer US.

Zuo, Y. and Panda, B. (2006). Distributed database damage assessment paradigm. *Inf. Manag. Comput. Security*, 14:116–139.