The Pennsylvania State University

The Graduate School

College of Information Sciences and Technology

MODELING AND EVALUATING THE SURVIVABILITY OF AN INTRUSION TOLERANT DATABASE SYSTEM

A Thesis in

Information Sciences and Technology

by

Hai Wang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2007

The thesis of Hai Wang was reviewed and approved* by the following:

Peng Liu
Associate Professor of Information Sciences and Technology
Thesis Adviser
Chair of Committee

C. Lee Giles
Professor of Information Sciences and Technology

Dongwon Lee
Assistant Professor of Information Sciences and Technology

Akhil Kumar
Professor of Business

Joseph M. Lambert
Associate Professor of Information Sciences and Technology
Senior Associate Dean
Associate Dean for Graduate Programs

*Signatures are on file in the Graduate School.

Abstract

The immaturity of current intrusion detection techniques limits the traditional security systems in surviving malicious attacks. Intrusion tolerance approaches have emerged to overcome these vulnerabilities. Before intrusion tolerance is accepted as an approach to security, there must be quantitative techniques to measure its survivability. However, there are very few attempts to do quantitative, model-based evaluation of the survivability of intrusion tolerant systems, especially in database field. In this thesis, I focus on modeling the survivability of an intrusion tolerant database system in the presence of attacks.

Before studying the survivability of intrusion tolerant systems, we need to have a better understanding of the attack behavior and its degree of spreading. Based on the classical epidemic model, a stochastic database damage propagation model is proposed. This model leads to a better understanding and prediction of the scale and speed of database damage propagation.

To study the survivability, the intrusion tolerant database system is modeled as a series of state transition models. Based on the Continuous Time Markov Chain (CTMC) and semi-Markov models, quantitative measures are proposed to characterize the capability of a resilient database system surviving intrusions. These facilitate a systematic evaluation to capture the survivability of intrusion tolerant database systems and the impact of system deficiencies on it. An Intrusion Tolerant DataBase system (ITDB) is studied as an example.

My experiment results validate the proposed CTMC and semi-Markov models. Survivability evaluation is conducted to study the impact of attack intensity and different system deficiencies on system survivability. The impact of intrusion tolerance operations on performance is also evaluated using the TPC-C benchmark. The performance measurements show that ITDB system is cost-effective within reasonable False Alarm Rates and Detection Latencies range.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am indebted to my thesis advisors, Dr. Peng Liu for the guidance and the encouragement. He has greatly helped me to overcome various difficulties and frustrations, and developed my academic thinking and research skills. It is really fortunate to work with him during my time here at Pennsylvania State University.

I am grateful to my doctoral committee members, Dr. Lee Giles, Dr. Dongwon Lee, and Dr. Akhil Kumar, for spending time on my committee and providing insightful suggestions on the thesis.

Special thanks are due to Dr. Chao Chu, Dr. Prasenjit Mitra, Dr. Sencun Zhu, Dr. Xiaolong (Luke) Zhang. Discussions with or suggestions from them at different stages were very rewarding.

The thesis cannot be finished without my wife. She shared my happiness and sadness throughout the 4-year hard work. She gave me great supports, and made my life colorful and meaningful. I wish her the best in her doctoral study and future career. I thank my parents. They gave me unconditional supports, even though they are far away from me. The greatest debt I owe is to my brother. Thanks for his great support.

Finally, I thank my lab mates, Robert Cole, Kun Bai, Qijun Gu, Yoon-Chan Jhi, Kameswari Kotapati, Fengjun Li, Lunquan Li, Chi-Chun Pan, and Tao Yang, for the discussion on research problems. I thank my friends as well as anyone not in this list that gave me help on my work and life.

This work was supported by NSF CCR-0233324 and Department of Energy Early Career PI Award. Some of the materials in the thesis have been published or accepted for publication [1], [2], [3].

Chapter 1

Introduction

1.1 Research Problem

Database security concerns the *confidentiality*, *integrity*, and *availability* of information stored in a database. A broad span of research [4, 5, 6] addresses primarily how to protect the *secrecy* of a database, namely its confidentiality. However, very limited research has been done on how to survive successful database attacks, which can seriously impair the integrity and availability of a database. Experience with data-intensive applications such as credit card billing, banking, and online stock trading, have shown that a variety of attacks do succeed to fool traditional database protection mechanisms. These attacks include but are not limited to malicious transactions through insider attacks, identity theft attacks, SQL injection attacks, and cross site scripting flaws [7]. In fact, we must recognize that not all attacks, even obvious ones, can be averted at their outset. Attacks that succeed, to some degree at least, are unavoidable. To remedy the limitations of database protection mechanisms, survivable database systems that can tolerate malicious attacks are getting increasingly important to data-intensive applications. Some intrusion tolerant database systems [8, 9] have been proposed recently, where in-depth defenses have been added to detect and respond to database intrusions.

However, despite that intrusion tolerance techniques, which gain impressive attention recently, are claimed to be able to enhance the system survivability, evaluation

models are largely overlooked in the previous research. Quantifying survivability metrics of database systems are needed and important to meet the user requirements and compare different intrusion tolerant architectures. Efforts aimed at survivability evaluation have been based on classic reliability or availability models.

The present work is motivated by the limitations of existing criteria for reliability and availability to evaluate survivability. The evaluation criteria for system reliability and availability have a fairly matured literature as summarized in [10, 11]. As defined in [12], *reliability* is the probability that a system can perform a specified service throughout a specified interval of time. *Availability* is a quantification of the alternation between proper and improper service, and is often expressed as the fraction of time that a system can be used for its intended purpose during a specified interval of time or in steady state. Survivability refers to the capability of a system to complete its mission in a timely manner, even if significant portions are compromised by attacks or accidents [13]. However, the reliability and availability models cannot be used to quantify survivability of a security system. Aside from the differences between security and fault tolerance, a fundamental reason for this is the fact that the classic the availability model assumes "fail-stop" semantics, whereas "attack-stop" semantics probably can never be assumed in trustworthy data processing systems, not only because of the substantial detection latency, but also because of the need for degraded services.

The goal of this work is to take the first step toward development of a survivability evaluation model that can systematically address the inherent limitations of classic reliability and availability evaluation models in measuring survivability. The approach I proposed utilizes a state transition graph to model an intrusion tolerant database system.

I attempt to model the system in a modular way, so that it can be easily adapted to a wide variety of intrusion tolerant database systems. Quantitative measures are proposed to characterize the capability of a security database system surviving intrusions. Furthermore, I am interested in understanding the impact of existing system deficiencies, such as false positive, and attack behaviors on the survivability. In this thesis, I take the first steps toward performing a detailed, quantitative evaluation of the survivability of intrusion tolerant database system and assessing the impact of system deficiencies and attack behaviors on it.

1.2 Contribution of This Study

In particular, the main goals of this thesis are five-fold:

1. Based the traditional epidemic model, a stochastic (Markovian) damage propagation model in continuous time is proposed. This model leads to a better understanding and prediction of the scale and speed of damage propagation in database system.

2. Extending the classic availability model to a new survivability model. Comprehensive state-space approaches are applied to study the complex relationships and their transition structure encoding sequencing response of intrusion tolerant database systems facing attacks.

3. Novel quantitative survivability evaluation metrics are proposed. Mean Time to Attack (MTTA), Mean Time to Detection (MTTD), Mean Time to Marking (MTTM), and Mean Time to Repair (MTTR) are proposed as basic measures

of survivability. I found that there is a natural mapping between the MTTA-MTTD-MTTM-MTTR model and the steady state probabilities of the system in state transition modeling. This mapping not only provides valuable insights into why the MTTA-MTTD-MTTM-MTTR model can measure survivability, but also provides a convenient way to use mathematical analysis to quantify survivability. Based on the MTTA-MTTD-MTTM-MTTR model, this survivability measuring methodology is no longer ad hoc.

4. In some real cases the attack and repair time are not exponentially distributed. Semi-Markov process has the advantage of allowing nonexponential distributions for transitions between states and to generalize several kinds of stochastic processes. I extended the CTMC models to semi-Markov process which can fit the real world better.

5. To validate the survivability model I proposed, a representative intrusion tolerant database system, ITDB [8], is studied as an empirical example. A real testbed is established to conduct comprehensive validation experiments running TPC-C benchmark transactions. Experiment results show the validity of the survivability model.

6. To further validate the security of ITDB, I have done an empirical survivability evaluation, where maximum-likelihood methods are applied to estimate the values of the parameters used in my state transition models. The impacts of existing system deficiencies and attack behaviors on the survivability are studied using quantitative measures I defined.

1.3 Outline

The rest of the thesis is organized as follows. In Chapter 2, I overview related work. It discusses the existing intrusion tolerance systems, the security criteria, and evaluation techniques.

In Chapter 3, I give an overview of the ITDB framework. Four important components of ITDB, namely transaction proxy, attack recovery, damage quarantine, and intrusion detection subsystems, are introduced.

In Chapter 4, damage spreading phenomena in database is discussed. Based on the classical epidemic SIS model, a stochastic (Markovian) damage propagation model in continuous time is proposed.

In Chapter 5, a series of state transition models are proposed. The basic state transition model focuses on a general intrusion tolerant database system. In the intrusion detection system model, a comprehensive model of intrusion detection subsystem is integrated into the whole system. A comprehensive state transition model of ITDB is proposed in the end of this chapter.

In Chapter 6, quantitative measures are proposed to facilitate evaluating the survivability of intrusion tolerant database systems from several aspects. Mean Time to Attack (MTTA), Mean Time to Detection (MTTD), Mean Time to Marking (MTTM), and Mean Time to Repair (MTTR) are proposed as basic measures of survivability. Instead of using some traditional metrics, two novel evaluation metrics, Integrity and Rewarding Availability, are proposed to quantify survivability. I also analyze the state models proposed in this chapter.

In Chapter 7, I extend the CTMC models to semi-Markov processes. Semi-Markov processes are able to allow nonexponential distributions for transitions between states and to generalize several kinds of stochastic processes. It is important to model the most real cases which do not follow exponential distribution.

In Chapter 8, a real testing environment is built to validate the proposed state transition models. TPC-c benchmark is chose as background transaction. A representative intrusion tolerant database system, ITDB, is implemented as an empirical example.

Chapter 9 discusses the experiments conducted to validate the established models. I compare the steady state probabilities of my models with a set of measured ITDB behaviors facing attacks.

Survivability and performance evaluation results are reported in Chapter 10. Instead of evaluating the performance of a specified system, I focus on the impact of different system deficiencies on the survivability in the face of attacks. Different detection deficiencies, such as false alarm rate, detection latency, and different workloads, such as attack rate are studied in this part.

I conclude my thesis in Chapter 11 where future work is also discussed.

Chapter 2

Overview of Previous Research

The development of techniques for quantitative evaluation of computer system security has a long and rich history. In this chapter, I review the previous research on modeling and evaluating the survivability in the literature.

2.1 Intrusion Detection

Limited research has been conducted in the field of database intrusion detection. The Hidden Markov Model has been proposed in [14] to detect malicious data corruption. A misuse detection system called DEMIDS presented by Chung et al. in [15]. DEMIDS uses audit logs to derive profiles that describe typical behavior of users working with the DBMS. The profiles computed can be used to detect misuse behavior, in particular insider abuse. Furthermore, the profiles can serve as a valuable tool for security re-engineering of an organization by helping the security officers to define/refine security policies and to verify existing security policies, if there are any.

Lee et al. [16] have used time signatures in discovering database intrusions. Their approach is to tag the time signature to data items. A security alarm is raised when a transaction attempts to write a temporal data object that has already been updated within a certain period.

Hu et al. [17] proposed a data mining approach for detecting malicious transactions in a database system. Their approach is to mine data dependencies among data items in the database. A data dependency miner is designed for mining data correlations from the database log. The transactions not compliant to the data dependencies mined are identified as malicious transactions. The experiment illustrates that the proposed method works effectively for detecting malicious transactions provided certain data dependencies exist in the database.

2.2 Intrusion Tolerance

The need for intrusion tolerance, or survivability, has been recognized by many researchers in such contexts as information warfare [18]. Recently, extensive research has been done in general principles of survivability [19], survivable software architectures [20], survivability of networks [21], survivable storage systems [22], etc., and many survivable systems have been designed and implemented. Forensix [23] uses monitoring and reconstruction mechanisms to improve the survivability of distributed systems and reduce the human overhead of performing forensic analysis. BackTracker [24] logs system calls and analyzes dependencies between operating system objects. After an IDS detected a suspicious process (update), BackTracker generated the backward graph that led to this detection point.

This research is helpful for database survivability, but the techniques cannot be directly applied to build intrusion tolerant database systems.

Some research has also been done on database survivability. In [25], focusing on recovery from malicious but committed transactions, both static and on-the-fly trusted

repair algorithms are proposed to restore only the damaged part of the database based on the read-write dependencies between transactions. The dynamic repair algorithms, compared with static repair algorithms, may back out more good transactions, but they give users more availability and less service delay. Performance issues are discussed at the end of the paper, but there is no quantitative result presented.

An on-the-fly damage assessment and repair tool for intrusion tolerant commercial database applications, ODAM, is proposed and implemented in [26]. ODAM locates and repairs damage on-the-fly without the need to periodically halt normal transaction processing. Performance evaluation is performed and the results show that the impact of ODAM on transaction processing throughput is small. The impact of intrusion detection deficiencies on survivability is, however, not considered in that thesis.

In [8], an Intrusion Tolerant Database system (ITDB) is developed. ITDB can detect intrusions, assess, and repair the damage caused by intrusions in a timely manner such that sustained, self-stabilized levels of data integrity and availability can be provided to applications in the face of attacks. Preliminary evaluation results are reported, which show that ITDB can achieve attack recovery with reasonable performance degradation. In [27], another portable intrusion-resilient database management system is proposed, which can restore the consistency of database after being compromised by a malicious attack or a human error. However, survivability evaluation is missed in both papers.

2.3 Security Criteria

First published in 1983, the Trusted Computer System Evaluation Criteria, or TCSEC, known as the Orange Book [28] is a set of basic requirements and evaluation

classes for assessing the effectiveness of security controls built into a computer system. These criteria are intended for use in the evaluation and selection of computer systems being considered for the processing and/or storage and retrieval of sensitive or classified information by the United States Department of Defense (DoD). The Orange Book, and others in the Rainbow Series, are still the benchmark for systems produced almost two decades later. It was eventually replaced with development of the Common Criteria international standard.

The Common Criteria (CC) [29] is an international standard (ISO/IEC 15408) for computer security. It describes a framework in which computer system users can specify their security requirements, developers can make claims about the security attributes of their products, and evaluators can determine if a Target of Evaluation (TOE) meets its claimed security functionality and assurance requirements. In other words, Common Criteria provides assurance that the process of specifying, developing, and evaluating a computer security product has been conducted in a rigorous manner.

However, Common Criteria allows a vendor to make certain assumptions about the operating environment and the strength of threats faced by the product in that environment. Based on these assumptions, the claimed security functions of the product are evaluated. So, if a product is ISO 15408 certified, it should only be considered secure in the assumed, specified circumstances.

Some engineers also use reliability and availability models to evaluate system survivability. This is due to the similarity between system failures due to intentional attacks and those due to accidental component failures.

Recommendations E.800 of the International Telecommunications Union (ITU-T) defines reliability as follows: "The ability of an item to perform a required function under given conditions for a given time interval." In computer science, reliability is defined as the probability that the system continues to function throughout the interval $(0, t)$. It is not necessary to (but it is often) assume that the system is functioning at time 0.

Availability is closely related to reliability, and is defined in ITU-T Recommendation E.800 as follows: "The ability of an item to be in a state to perform a required function at a given instant of time or at any instant of time within a given time interval, assuming that the external resources, if required, are provided." An important difference between reliability and availability is that reliability refers to failure-free operation during an interval, while availability refers to failure-free operation at a given instant of time, usually the time when a device or system is first accessed to provide a required function or service.

However, the availability model, which has a fairly matured literature as summarized in [10], has limitations to quantify survivability of a security system. Besides the differences between security and fault tolerance, a fundamental reason is because the availability model assumes the "fail-stop" semantics, but the "attack-stop" semantics probably can never be assumed in trustworthy data processing systems, not only because of the substantial detection latency, but also because of the needs for degraded services.

2.4 Survivability evaluation

Despite that intrusion tolerance techniques, which gain impressive attention recently, are claimed to be able to enhance the system survivability, suitable and precise measures to evaluate the survivability of an intrusion tolerant system are largely missed in the previous research. Most of the research in the literature report and discuss the survivable capability of their work from a qualitative point of view. Little research has proposed quantitative evaluation metrics of survivability.

In [13] and [30], formal definitions of survivability are presented and compared with related concepts of reliability, availability, and dependability. [30] defined the survivability from several aspects and claimed that the big difference between reliability and survivability is that degraded services of survivable systems are acceptable to users, reliability assumes that the system is either available or not. However, the quantitative measurements of survivability and the level of degraded services are missing in that study.

The attacks and the response of an intrusion tolerant system are modeled as a random process in [31]. Stochastic modeling techniques are used to capture the attacker behavior as well as the system's response to a security intrusion. Quantitative security attributes of the system are proposed in the paper. Steady-state behaviors are used to analyze the security characterization. A security measure called the *mean time (or effort) to security failure* is proposed. However, "good guestimate" values of model parameters were used in their experiments. And the validation of their models is missing in their work.

Efforts for quantitative validation of security have usually been based on formal methods [32]. [33] shows that probabilistic validation through stochastic modeling is an attractive mechanism for evaluating intrusion tolerance. The authors use stochastic activity networks to quantitatively validate an intrusion-tolerant replication management system. Several measures defined on the model were proposed to study the survivability provided by the intrusion tolerant system. The impacts of system parameters variations are studied in that work.

Although several survivability models and corresponding measurements were proposed in the literature, they are limited in evaluating the security attributions of an intrusion tolerant database system. Zhang and Liu [34] take the first step towards delivering database services with information assurance guarantees. In particular, (a) the authors introduce the concept of Quality of Integrity Assurance(QoIA) services; (b) a data integrity model, which allows customers or applications to quantitatively specify their integrity requirements on the services that they want the database system to deliver, is proposed; and (c) the authors present an algorithm that can enable a database system to deliver a set of QoIA services without violating the integrity requirements specified by the customers on the set of services.

An online attack recovery system for work flow is proposed in [35]. The behaviors of the recovery system are analyzed based on a Continuous Time Markov Chain model. Both steady-state and transient behaviors are studied in that paper. Only three state categories, 'NORMAL', 'SCAN', and 'RECOVERY', are considered in the model. The deficiency of intrusion detection and damage propagation are not considered in that model.

In [3], the authors have done detailed, quantitative evaluation on the impact of intrusion detection deficiencies on the performance and survivability by running TPC-C benchmark. However, only some ad hoc survivability metrics were used. Systematic survivability model and measurements were not proposed in [3].

Chapter 3

ITDB: An Motivating Example

ITDB is motivated by the following practical goal: "After the database is damaged, automatically locate the damaged part, quarantine and repair it as soon as possible, so that the database can continue being useful in the face of attacks or intrusion." In other words, ITDB wants to provide sustained levels of data integrity and availability to applications in the face of attacks.

ITDB focuses on the damage caused by malicious, committed transactions. Note that an active transaction will not cause any (durable) damage before it commits. The ultimate goal of ITDB is to locate and repair the damage in a timely manner such that self-stabilized levels of data integrity and availability can be provided to applications. The major components of ITDB are shown in Figure 3.1. ITDB is built on top of an off-the-shelf DBMS, such as Oracle. Note that in [36], a comprehensive ITDB system has been proposed. In this thesis, I only focus on some important components of ITDB, namely the attack recovery subsystem and damage quarantine subsystem.

In the rest of this chapter, I give an overview of the functions of some important ITDB components. Based on the ITDB model proposed in [36], I improved some subsystems of ITDB.

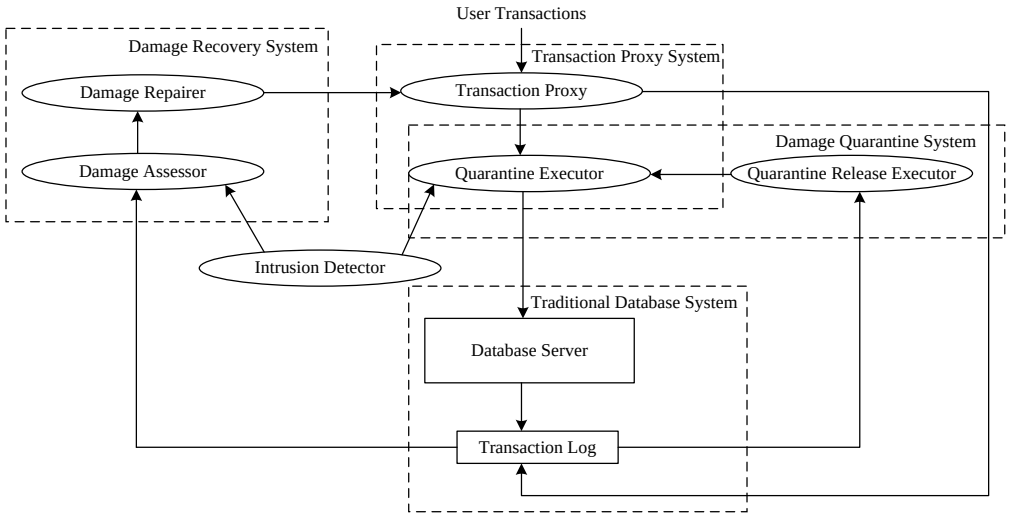


Fig. 3.1. Basic ITDB System Architecture

3.1 Transaction Proxy Subsystem

The transaction proxy subsystem functions as a "proxy" for each user transaction and transaction processing call to the database system. ITDB is a concurrent system triggered by a set of specific events. The proxy subsystem is able to keep useful information about these events which are important to generate the corresponding logs for attack recovery and quarantine. The major events are as follows:

- When a user transaction T is submitted by an application, the proxy will send each SQL statement and transaction processing call of T , and keep useful information about T and these SQL statements;

- When a SQL statement is executed, the Trail Collectors will log the corresponding writes in the Write Log, the corresponding reads will instead be extracted from the statement text into the Read Log, and the Intrusion Detector will assess the

suspicion level of the corresponding transaction and session using the trails kept in the Write Log (and possibly some other audit tables);

- When the Intrusion Detector identifies a malicious transaction, the Damage Assessor will start to locate the damage caused by the transaction, and the proxy and the damage quarantine subsystem will start the multi-phase damage quarantine process (if needed);

- When the damage caused by a malicious or affected transaction is located, the Damage Repairer will compose and submit a specific cleaning transaction to repair the damage;

- When the Damage Assessor finds an unaffected transaction, when the Damage Quarantine Executor identifies an undamaged object, or when a cleaning transaction commits, some objects will be reported to the proxy to release from quarantine.

The transaction proxy subsystem is the foundation of the whole ITDB system. All the other subsystems of ITDB rely on it.

3.2 Attack Recovery subsystem

The attack recovery subsystem has the responsibility to perform on-the-fly damage assessment and repair. To do this job, first, Intrusion Detector monitors and analyzes the trails of database sessions and transactions in a real-time manner to identify malicious transactions as soon as possible. Alarms of malicious transactions, when raised, will be instantly sent to the Repair Manager, which will locate the damage caused by the

attack and repair the damage. During the whole intrusion detection and attack recovery process, the database continues executing new transactions.

The key challenge of attack recovery is a vulnerability denoted damage spreading. In a database, the results of one transaction can affect the execution of some other transactions. When a transaction T_i reads a data object x updated by another transaction T_j , T_i is directly affected by T_j. If a third transaction T_k is affected by T_i, but not directly affected by T_j , T_k is indirectly affected by T_j. It is easy to see that when a (relatively old) transaction B_i that updates x is identified malicious, the damage on x can spread to every object updated by a good transaction that is affected by B_i, directly or indirectly. In a word, the read-from dependency among transactions forms the traces along which damage spreads. In Chapter 4, I will discuss the damage spreading phenomena in detail.

The job of attack recovery is two-fold: damage assessment and repair. In particular, the job of the Damage Assessor is to locate each affected good transaction, i.e., the damage spreading traces; and the job of the Damage Repairer is to recover the database from the damage caused on the objects updated along the traces. The damage repair algorithm proposed in [25] can be seen in Appendix A. In particular, when an affected transaction T is located, the Damage Repairer builds a specific cleaning transaction to clean each object updated T (and not cleaned yet). Cleaning an object is simply done by restoring the value of the object to its latest undamaged version. In contrast with traditional database undo operations which are performed backwardly, ITDB executes cleaning transactions forwardly for two reasons: (1) to reduce the repair time; (2) to not block newly incoming user transactions.

In this way, it is possible that a data item cleaned by a previous cleaning transaction is mistakenly damaged by a later cleaning transaction if the cleaning transaction of every transaction just restores each item updated by the transaction to the value before the update. To avoid this problem, ITDB, when composing a cleaning transaction, only handles the items that are damaged and are not under cleaning.

Temporarily stopping the database will certainly make the attack recovery job simpler since the damage will no longer spread and the repair can be done backwardly after the assessment is done. However, since many critical database servers need to be 24*7 available and temporarily making the database shut down can be the real goal of the attacker, on-the-fly attack recovery which never stops the database is necessary in many cases.

On-the-fly attack recovery faces several unique challenges. First, ITDB needs to do repair forwardly since the assessment process may never stop. Second, cleaned data objects could be re-damaged during attack recovery. Finally, the attack recovery process may never terminate. Since as the damaged objects are identified and cleaned new transactions can spread damage if they read a damaged but still unidentified object, so ITDB faces two critical questions: (1) Will the attack recovery process terminate? (2) If the attack recovery process terminates, can ITDB detect the termination?

To tackle challenge 1, ITDB ensures that a later cleaning transaction will not accidentally damage an object cleaned by a previous cleaning transaction. To tackle challenge 2, ITDB never mistakes a cleaned object as damaged, and ITDB never mistakes a re-damaged object as already cleaned. To tackle challenge 3, a previous study in [25] shows that when the damage spreading speed is quicker than the repair speed, the

repair may never terminate. Otherwise, the repair process will terminate, and under the following three conditions ITDB can ensure that the repair terminates: (1) every malicious transaction is cleaned; (2) every identified damaged object is cleaned; (3) further (assessment) scans will not identify any new damage (if no new attack comes).

3.3 Damage Quarantine subsystem

Traditional damage quarantine approaches are one-phase. An item o will not be quarantined until it is identified as damaged. However, significant damage assessment latency can cause the damage on o to spread to many other data items before o is quarantined. To overcome this limitation, ITDB uses a novel technique called multi-phase damage quarantine as an alternative. This approach has one quarantining phase, which instantly quarantines the damage that might have been caused by an intrusion as soon as the intrusion is identified, and one or more later quarantine releasing phases, denoted *quarantine relaxation*, to releae the items that are mistakenly quarantined during the quarantine phase.

ITDB models the quarantine phase as a simple estimation problem. When a malicious transaction B_i is detected, the quarantine phase gets the set of items to quarantine by 'estimating' which items in the database may have been damaged. This set is called a *quarantine set*. Accurate damage assessment is not possible in this phase, since our approach requires the quarantine phase be finished in a very short period of time so that the damage can be quarantined before any new transaction is executed (to possibly spread the damage), and assessing damage usually needs a much longer period of time to do accurate assessment. Although some approximate estimation approaches are also

investigated, ITDB currently performs over-estimation using time stamps. ITDB lets the transaction proxy tag each record with a time stamp, which is kept in the time stamp columns of each table, to time the latest write on the record. When B_i is identified, the quarantine phase is simply done by denying any new access to the items that are updated between the time B_i starts and the time this quarantine control is enforced.

ITDB models each later quarantine releasing phase as a process to transform one quarantine set to another. Hence the whole multi-phase quarantine process can be modeled by a sequence of confinement sets, denoted S_E, S_2, S_3, $\cdots$, S_n, S_D. S_E indicates the result of the initial quarantine phase. S_D indicates the set of data items that are really damaged. S_i, $2 \leq i \leq n$, indicates the result of an quarantine releasing phase. The goal of the quarantine releasing phases is to converge the sequence to S_D. The sequence can be converged to S_D in many different ways. The way ITDB takes is that $S_E \supseteq S_1, S_i \supseteq S_j$ for $2 \leq i < j \leq n$, and $S_n \supseteq S_D$. In particular, when the Unconfinement Executor scans the transactions logs and finds that a transaction G_i is not a damage spreading transactions, all the items that are updated by G_i and are associated with a time stamp earlier than the commit time of G_i will be released

In this way, the ITDB can ensure that after the quarantine phase no damage leaks out. As a result, the attack recovery process needs only to handle the quarantine time window, and the termination problem disappears. To implement the Damage Quarantine Subsystem, we use the Mediator to rewrite normal SQL statements to transparently maintain time stamps. We also use the Quarantine Executor use reads extracted from SQL statements to enforce damage quarantine control.

3.4 Intrusion Detection Subsystem

Intrusion Detection (ID) is the art of detecting inappropriate, incorrect, or anomalous activity. The intrusion detection subsystem of ITDB has the responsibility to detect and report malicious transactions to the attack recovery subsystem.

3.4.1 Host-based versus Network-based Intrusion Detection

ID systems that operate on a host to detect malicious activity on that host are called host-based ID systems, and ID systems that operate on network data flows are called network-based ID systems.

Host-based ID involves loading a piece or pieces of software on the system to be monitored. The loaded software uses log files and/or the system's auditing agents as sources of data. In contrast, a network-based ID system monitors the traffic on its network segment as a data source.

ITDB uses Transaction Proxy subsystem to keep log files of user transactions. The ID subsystem uses these log files as sources of data to detect intrusions. To get complete coverage, a host-based ID system is more suitable for ITDB.

3.4.2 Knowledge-based Intrusion Detection

The most common approaches to ID are statistical anomaly detection and knowledge-based detection. Knowledge-based intrusion detection techniques apply the knowledge accumulated about specific attacks and system vulnerabilities. The intrusion detection system contains information about these vulnerabilities and looks for attempts to exploit these vulnerabilities. When such an attempt is detected, an alarm is triggered. In other

words, any action that is not explicitly recognized as an attack is considered acceptable. Therefore, the accuracy of knowledge-based intrusion detection systems is considered good. However, their completeness (i.e. the fact that they detect all possible attacks) depends on the regular update of knowledge about attacks.

Advantages of the knowledge-based approaches are that they have the potential for very low false alarm rates, and the contextual analysis proposed by the intrusion detection system is detailed, making it easier for the security officer using this intrusion detection system to take preventive or corrective action.

Drawbacks include the difficulty of gathering the required information on the known attacks and keeping it up to date with new vulnerabilities and environments. Maintenance of the knowledge base of the intrusion detection system requires careful analysis of each vulnerability and is therefore a time-consuming task. Knowledge-based approaches also have to face the generalization issue. Knowledge about attacks is very focused, dependent on the operating system, version, platform, and application. The resulting intrusion detection tool is therefore closely tied to a given environment. Also, detection of insider attacks involving an abuse of privileges is deemed more difficult because no vulnerability is actually exploited by the attacker.

3.4.3 Behavior-based Intrusion Detection

Behavior-based intrusion detection techniques assume that an intrusion can be detected by observing a deviation from normal or expected behavior of the system or the users. The model of normal or valid behavior is extracted from reference information collected by various means. The intrusion detection system later compares this model

with the current activity. When a deviation is observed, an alarm is generated. In other words, anything that does not correspond to a previously learned behavior is considered intrusive. Therefore, the intrusion detection system might be complete (i.e. all attacks should be caught), but its accuracy is a difficult issue (i.e. you get a lot of false alarms).

Advantages of behavior-based approaches are that they can detect attempts to exploit new and unforeseen vulnerabilities. They can even contribute to the (partially) automatic discovery of these new attacks. They are less dependent on operating system-specific mechanisms. They also help detect 'abuse of privileges' types of attacks that do not actually involve exploiting any security vulnerability. In short, this is the paranoid approach: everything which has not been seen previously is dangerous.

The high false alarm rate is generally cited as the main drawback of behavior-based techniques because the entire scope of the behavior of an information system may not be covered during the learning phase. Also, behavior can change over time, introducing the need for periodic online retraining of the behavior profile, resulting either in unavailability of the intrusion detection system or in additional false alarms. The information system can undergo attacks at the same time the intrusion detection system is learning the behavior. As a result, the behavior profile contains intrusive behavior, which is not detected as anomalous.

Database systems usually contain large amount of sensitive data or information. A high detection rate is more important for a secure database ID system, other than a low false alarm rate. ITDB focuses on detecting novel intrusions. A behavior-based ID system is suitable for ITDB.

The ID subsystem of ITDB uses the trails kept in the logs and some relevant rules to identify malicious transactions. In addition to some common characters, the ID subsystem distinguishes itself from other ID systems in four aspects. (a) It is a database intrusion detection system. (b) It is application aware. (c) It is at transaction level. (d) Instead of developing new intrusion detection techniques, ITDB applies existing behavior-based intrusion detection techniques, and instead focuses on the system support for flexible application-aware transaction-level database intrusion detection, and seamless integration of the intrusion detection subsystem into the ITDB prototype.

Chapter 4

Damage Propagation Model

The analysis of damage propagation has been the subject of a continuous interest in the computer security community. A computer virus can copy itself and infect another computer without permission or knowledge of the user. Worms become more effective at spreading more machines. The intrusion tolerant approaches allow but tolerate intrusions. When the strategies for resistance to attacks fail, the damage caused by the intrusion may be propagated in a large range before the intrusion are detected. If the attack is detected successfully, intrusion tolerance system picks up where attack resistance leaves off. A question is whether the intrusion tolerance system can recover the all damage before the system is crashed by the attack. To answer this question, we need to study the degree of damage spreading. However, there is little work to study the damage spreading and model its behaviors in a database system. In this chapter, I will take the first step to study the threat model and its propagation in a database system.

4.1 Background

An accurate propagation model can aid in identifying the weakness in damage spreading chain and provides accurate prediction for the damage assessment. In epidemiology research, there exist several deterministic and stochastic models for virus and worm spreading [37, 38, 39].

The standard Susceptible-Infected-Susceptible (SIS) model used in the study of computer virus infections is epidemiological model borrowed from biological epidemiology [40, 41]. In [37], each node of the network represents an individual and each link is a connection along which the infection can spread to other systems. Individuals exist in one of a small number of discrete states, such as "infected" or "susceptible". At each time step, each susceptible (healthy) node is infected with rate β if it is connected to one or more infected nodes. At the same time, infected nodes are cured and become again susceptible with death rate δ. A more realistic model of computer virus spreading is introduced in [38]. The model modified the SIS epidemiological model and includes a reintroduction parameter, which models the rerelease of a computer virus, or the introduction of a new virus.

Internet worms are similar to computer viruses in that both have been studied via epidemiological models that were originally developed in the field of biological epidemiology. In [39, 42], a two factor worm model is introduced. Based the classic epidemic model, two major factors that affect an Internet worm propagation are proposed. One is the effect of human countermeasures against worm spreading, countermeasures such as cleaning, patching, filtering or even disconnecting computers and networks; the other is the slowing down of worm infection rate due to the worms impact on Internet traffic and infrastructure. In [43], a mathematical model is presented, referred to as the Analytical Active Worm Propagation (AAWP) model, which characterizes the propagation of worms that employ random scanning.

In classical simple epidemic model, each host stays in one of two states: susceptible or infectious. The model assumes that once a host is infected by a virus, it will stay in

infectious state forever. Thus state transition of any host can only be: susceptible → infectious [44]. The classical simple epidemic model for a finite population is

$$\frac{dJ(t)}{dt} = \beta J(t)[N - J(t)] \tag{4.1}$$

where $J(t)$ is the number of infected hosts at time t; N is the size of population; and β is the infection rate. At beginning, $t = 0$, $J(0)$ hosts are infectious and the other $N - J(0)$ hosts are all susceptible.

4.2 Threats to Database Security

The three security services, availability, integrity, and confidentiality, counter threats to the security of a system [45]. According to these three services, Bertino [46] divides threats to database security into three broad categories: *incorrect data modification*, *data unavailability*, and *unauthorized data observation*. Incorrect modifications of data, either intentional or unintentional, result in an incorrect database state. Any use of incorrect data may result in heavy losses for the organization. When data is unavailable, information crucial for the proper functioning of the organization is not readily available when needed. Unauthorized data observation results in the disclosure of information to users not entitled to gain access to such information. Access control mechanisms support confidentiality. Intrusion tolerant systems triggers mechanisms that prevent the intrusion from generating access control mechanisms failure. To evaluate the survivability of an intrusion tolerant system, I will only consider incorrect data modification threats, data unavailability threats in this thesis.

Database systems can be attacked on several levels: transaction level, operating system (OS) level, and hardware level. The targets of transaction-level attacks are data items in the database. [47] shows that most threats to database security are from insiders. In addition, all of the cases involved attacks that affected the security of the organizations' data, while only 22% involved attacks that affected the security of the organizations' information systems/networks [48]. In this thesis, I will only consider transaction-level attacks, by which data items are compromised through malicious transactions.

The "SQL Injection" attack is a real example which has become one of the most serious threats to database applications. It is a class of code injection attacks in which specially crafted input strings result in illegal queries to a database. It applies to the popular Microsoft Internet Information Server/Active Server Pages/SQL Server platform. The idea of "SQL Injection" attack is to convince the application to run SQL code that is not intended. The technique of "SQL Injection" attack is to exploit a security vulnerability occurring in the database layer of an application. In this example, we could find the attacker can act as a Database Administer (DBA) to modify almost any data item in the database by submitting a malicious transaction. So in this thesis, I assume that the attacker can damage any data item in the database.

4.3 Damage Spreading

A successful transaction-level attack may seriously impair the integrity and availability of a database in depth. Every good transaction, which is *dependent upon* [25]

malicious transactions directly or indirectly, will propagate the damage and turn itself to be a bad transaction.

Definition. *In a history, a transaction T_i is dependency upon another transaction T_j if*

1. *there exists a data item x such that T_i reads x after T_j updates it,*
2. *and there is no other transaction that updates x between T_i and T_j.*

The *dependent upon* relation indicates the path along which damage propagates. Consider the following history over (B_1, G_1, G_2, G_3, G_4)

H: $r_{B_1[x]}$, $w_{B_1[x]}$, c_{B_1}, $r_{G_1[x]}$, $w_{G_1[x]}$, $r_{G_3[z]}$, $w_{G_3[z]}$, c_{G_3}, $r_{G_1[y]}$, $w_{G_1[y]}$, c_{G_1}, $r_{G_2[x]}$, $w_{G_2[x]}$, $r_{G_2[v]}$, $w_{G_2[v]}$, c_{G_2}, $r_{G_4[v]}$, $w_{G_4[v]}$, $r_{G_4[w]}$, $w_{G_4[w]}$, c_{G_4}

where c is commit.

Definition. *In a history, transaction T_1 affects transaction T_2 if the ordered pair (T_1, T_2) is in the transitive closure of the dependent upon relation. A good transaction G_i is suspect if some bad transaction B_i affects G_i.*

In this example, the damage caused by malicious transaction B_1 on data item x is propagated to new data items y, v, and w by some good transactions G_1, G_2, and G_4, which are dependent on B_1 directly or indirectly. Transaction G_3 does not propagate the damage since it is not dependent on B_1, and data item z keeps clean. Figure 4.1 shows the propagation path in the database.

A real example could be an attack on an online shopping system in which the attacker could change the price of certain item. Before the attack is detected successfully, every transaction submitted by legitimate users who want to buy this item is infected. As a result, the balance in each user's account is damaged by the infected transaction.

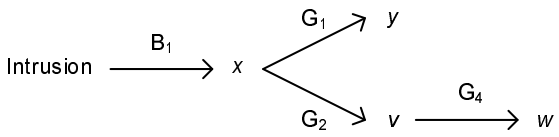


Fig. 4.1. Damage Propagation Path

4.4 Propagation Model

In this section, I will propose a stochastic (Markovian) damage propagation model in continuous time. Based on the traditional epidemic model, I assume that each data item has two only states: susceptible or infective. The model study the damage propagation without the effect of human interaction or damage recovery, which means once infected by an intrusion directly or indirectly, the data item will not be repaired, removed, or quarantined. It remains in the infective state. The effect of damage recovery and quarantine will be studied in Chapter 5.

Furthermore, I assume that the database size is a constant. In the case of database application, incoming transactions may insert new data items or delete old ones in the database. Usually the attack can be found by either the Intrusion Detection system (IDS) or Database Administrator (DBA) in a relatively short period of time. During this period, the change of database size is negligible.

Suppose that the user transactions occur successively in time, so that the intervals between successive events are independent and distributed according to an exponential distribution with parameter λ. Let the number of user transactions in the interval $(0, t]$

be denoted by $T(t)$. Then the stochastic process $\{T(t) \mid t \geq 0\}$ is a Poisson process with mean rate λ. Here λ is called the average transaction arrival rate in this thesis.

Suppose the database has a homogeneous population, which means that every data item in the database has the same probability to be *read* or *written* by a incoming transaction. Let I be the number of data items infected at time t and I_0 be the initial number of infected data items. I assume that the intrusion damages several data items at the very beginning and no further attack happens during the propagation. Let r be the average read size of a incoming user transaction, where the read size is the number of data items in its read set.

When $r \geq I$, the probability of a incoming transaction will read at lease one infected data item is:

$$\begin{aligned} p_r &= \frac{\binom{I}{1}\binom{N-I}{r-1} + \binom{I}{2}\binom{N-I}{r-2} + \cdots + \binom{I}{I}\binom{N-I}{r-I}}{\binom{N}{r}} \\ &= \frac{\sum_{k=1}^{I}\binom{I}{k}\binom{N-I}{r-k}}{\binom{N}{r}} \end{aligned} \tag{4.2}$$

When $r < I$, the probability is:

$$
\begin{aligned}
p_r &= \frac{\binom{I}{1}\binom{N-I}{r-1} + \binom{I}{2}\binom{N-I}{r-2} + \cdots + \binom{I}{I}\binom{N-I}{r-r}}{\binom{N}{r}} \\
&= \frac{\sum_{k=1}^{r}\binom{I}{k}\binom{N-I}{r-k}}{\binom{N}{r}} \qquad (4.3)
\end{aligned}
$$

When $r > N - I$, obviously the probability $p_r = 1$.

Given that $\binom{I}{k} = 0$ if $k > n$ or $k < 0$, we can combine the above three equations as:

$$
p_r = \frac{\sum_{k=1}^{r}\binom{I}{k}\binom{N-I}{r-k} - \sum_{k=I+1}^{r}\binom{I}{k}\binom{N-I}{r-k}}{\binom{N}{r}} - \qquad (4.4)
$$

According to **Vandermonde's identity**, we can get

$$
\begin{aligned}
\sum_{k=1}^{r}\binom{I}{k}\binom{N-I}{r-k} &= \sum_{k=0}^{r}\binom{I}{k}\binom{N-I}{r-k} - \binom{N-I}{r} \\
&= \binom{N}{r} - \binom{N-I}{r} \qquad (4.5)
\end{aligned}
$$

So that

$$
\begin{aligned}
p_r &= \frac{\sum_{k=1}^{r}\binom{I}{k}\binom{N-I}{r-k} - \sum_{k=I+1}^{r}\binom{I}{k}\binom{N-I}{r-k}}{\binom{N}{r}} \\
&= \frac{\binom{N}{r} - \binom{N-I}{r} - \sum_{k=I+1}^{r}\binom{I}{k}\binom{N-I}{r-k}}{\binom{N}{r}} \\
&= 1 - \frac{\binom{N-I}{r}}{\binom{N}{r}} - \frac{\sum_{k=I+1}^{r}\binom{I}{k}\binom{N-I}{r-k}}{\binom{N}{r}} \qquad (4.6)
\end{aligned}
$$

where $\binom{N-I}{r} = 0$ when $r > N - I$ and $\sum_{k=I+1}^{r}\binom{I}{k}\binom{N-I}{r-k} = 0$ when $r < I$.

As I described in Section 4.3, if an incoming transaction reads at least one infected data item, the transaction will propagate the damage to its write set. Let w be the average write size of an incoming transaction, where the write size is the number of data items in its write set. Here I assume that there is no overlap between the read set and the write set of an incoming transaction and $r, w > 0$.

The probability of the infected transaction infecting g good data items is:

$$
p_{w_g} = \frac{\binom{N-I}{g}\binom{I}{w-g}}{\binom{N}{w}} \qquad (4.7)
$$

In all, the probability of an incoming transaction reading at least one infected data item and spread the damage to g good data item is:

$$
\begin{aligned}
p_{S_g} &= p_r p_{w_g} \\
&= (1 - \frac{\binom{N-I}{r}}{\binom{N}{r}} - \frac{\sum_{k=I+1}^{r} \binom{I}{k}\binom{N-I}{r-k}}{\binom{N}{r}})(\frac{\binom{N-I}{g}\binom{I}{w-g}}{\binom{N}{w}}) \qquad (4.8)
\end{aligned}
$$

We assume that $\{(X,Y)(t) : t \geq 0\}$ is a homogeneous Markov chain in continuous time with state space satisfying

$$X(t) + Y(t) = N$$

where $X(t)$ and $Y(t)$ denote the number of susceptible and infective data items at time t, with initial conditions $(X,Y)(0) = (N-I, I)$ where $I \geq 1$, $N-1 \geq X(t) \geq 0$, and $N \geq Y(t) \geq I$. The transition probabilities are

$$
\begin{aligned}
&Pr\{(X,Y)(t+\delta t) = (i-g, j+g) \mid (X,Y)(t) = (i,j)\} = \lambda p_{S_g} \\
&= \lambda(1 - \frac{\binom{i}{r}}{\binom{N}{r}} - \frac{\sum_{k=j+1}^{r} \binom{i}{r-k}\binom{j}{k}}{\binom{N}{r}})(\frac{\binom{i}{g}\binom{j}{w-g}}{\binom{N}{w}}) \qquad (4.9)
\end{aligned}
$$

with $(X,Y)(t) = (0,N)$ being an absorbing state.

We now analyze the Markov chain of the numbers of infective data items $Y(t) : t \geq 0$. The transition probabilities are:

$$Pr\{(Y)(t+\delta t) = (j+g) \mid (Y)(t) = (j)\} = p_{j,(j+g)} = \lambda p_{S_g}$$

$$= \lambda(1 - \frac{\binom{N-j}{r}}{\binom{N}{r}} - \frac{\sum_{k=j+1}^{r}\binom{N-j}{r-k}\binom{j}{k}}{\binom{N}{r}})(\frac{\binom{N-j}{g}\binom{j}{w-g}}{\binom{N}{w}}) \tag{4.10}$$

with $(Y)(t) = N$ being an absorbing state.

Let $\mathbf{Q}(t) = [q_{ij}]$ be the generator matrix of the homogeneous Markov chain. Let $\mathbf{P}_j(t) = P\{Y(t) = j\}$ denote the unconditional probability that the Markov chain will be in state j at time t, and the row vector $\mathbf{P}(t) = [P_1, P_2, \cdots, P_n]$ represent the transient state probability vector of the CTMC. The transient behavior of the CTMC can be described by the Kolmogorov differential equation:

$$\frac{d\mathbf{P}(t)}{dt} = \mathbf{P}(t)\mathbf{Q} \tag{4.11}$$

where $\mathbf{P}(0)$ represents the initial probability vector (at time $t = 0$).

Once state probabilities have been computed, measures of interest are usually obtained as weighted averages of these quantities. Assume the number of infective data items j is the weight r_j attached to state j. Let $Z(t) = r_{Y(t)}$ be the weight of the Markov chain at time t. Then the expected instantaneous number of infective data items at time t is

$$E[Z(t)] = \sum_{j=1}^{N} r_j P_j(t)$$
$$= \sum_{j=1}^{N} j P_j(t) \quad (4.12)$$

Table 4.1. Notations in Chapter 4

Notation	Explanation
N	Database size, number of data items in the database
I	Number of infective data items at time t
I_0	Initial number of infected data items
r	Average read size, number of data items in the read set
w	Average write size, number of data items in the write set
λ	Average transaction arrival rate
p_r	Probability of a incoming transaction reading at lease one infected data item
p_{w_g}	Probability of a infected transaction damaging g good data items
p_{S_g}	Probability of a incoming transaction damaging g good data items
X(t)	Number of susceptible data items at time t
Y(t)	Number of infective data items at time t
Z(t)	Weight of Y(t) at time t

4.5 Simulation Experiments

In this section, I study the damage propagation in database by running simulation experiments. Instead of applying models to calculate the number of infected data items, I focus on studying the impact of different parameters, e.g. transaction arrival rate, database size, on the damage spreading. The model with homogeneous population is applied to study the impact of different parameters.

4.5.1 Impacts of Transaction Arrival Rate

In this part, I simulate 3 scenarios with different workloads. Simulation results are shown in Figure 4.2. In Figure 4.2(a), it shows the damage spreading at the start stage, while a long term propagation is shown in Figure 4.2(b). In this way, we could have a clear image of the damage spreading in both early stage and long term.

The high transaction arrival rate not only increases the database workload, but also increases the probability that a good transaction will touch an infected data item. As can be seen in Figure 4.2(b), nearly all data items in the database are infected in 100 minutes when transaction arrival rate is 10 tps (transaction per second), while the number of infected data item is under 400 (4%) when transaction arrival rate is 5 tps and the number of infected data item is only around 5 when transaction arrival rate is 1 tps. This suggests that, if the intrusion happens, it is better to slow down the incoming transactions. This will help us to slow down the damage propagation in the database. On the other hand, this will also lighten the database workload. The database will not keep busy. The recovery system will have more resources to use to quarantine or repair the damage.

4.5.2 Impacts of Transaction Size

In this part, I study the impacts of transaction size on the damage propagation in the database. The transaction size means the transaction's *read* and *write size*. Obviously, for the transaction with a larger *read size*, the probability that the transaction will read or touch an infected data item is higher. For the transaction with a larger *write size*, it will infect more good data items. As can be seen in Figure 4.3, the transaction

with a larger size ($r = 10, w = 5$) will spread the damage quickly. It can affect all data items in the database in 100 minutes. Compared with the impacts of transaction arrival rate, the impact of transaction size is more significant.

Also, we could find that the transaction with a smaller size ($r = 10, w = 2$) will spread the damage slowly. Around 400 data items (4%) are infected in 100 minutes. This gives us some guidelines on database design. It is better to design transactions with small read and write set. For the large transaction, we can try to break it down to smaller ones. This will help to decrease the damage spreading significantly.

4.5.3 Impacts of Database Size

Database size is another factor affecting the damage spreading in a database system. Obviously, if a database has a lot of data items, the probability that an incoming transaction will read an infected data item is very small. In contrast, for a small database, it is easy for an incoming transaction to read an infected data item. As shown in Figure 4.4(b), almost all the data items in a small database with 5,000 data items are infected in 100 minutes. However, for larger databases ($N = 10k$, $N = 50k$), the damage propagation is not significant. Only around 500 data items (5%) are infected in 100 minutes when $N = 10k$.

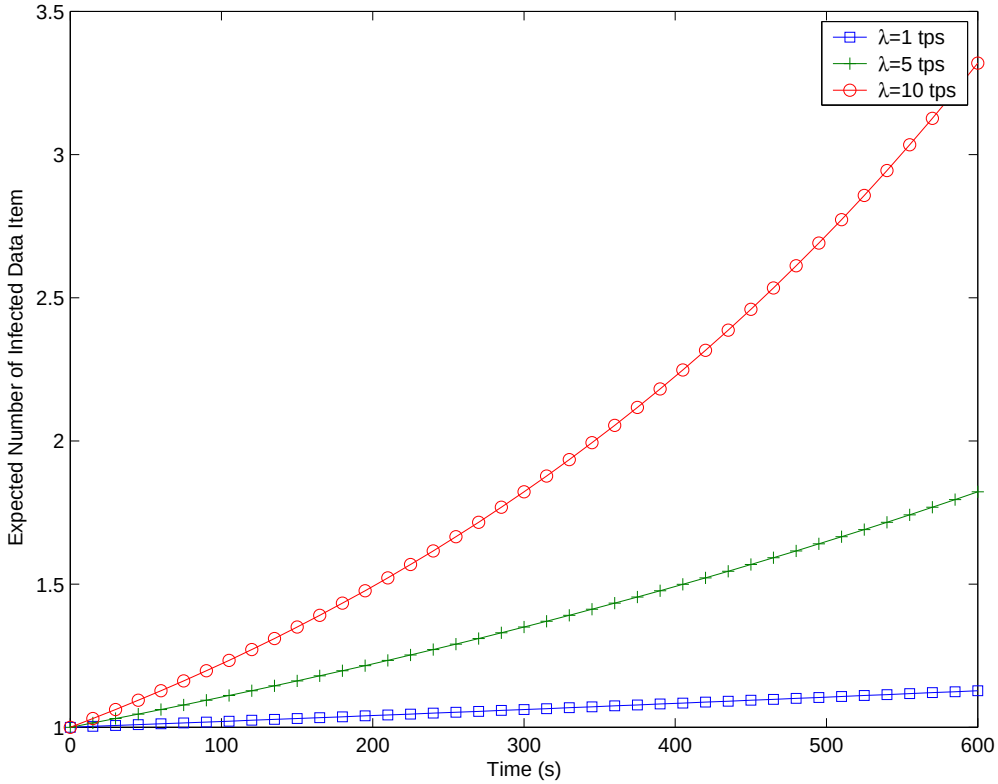
4.5.4 Impacts of Initial Infected Number

In the damage propagation models, I assume that the intrusion damages several data items at the very beginning and no further attach happens. The initial infected

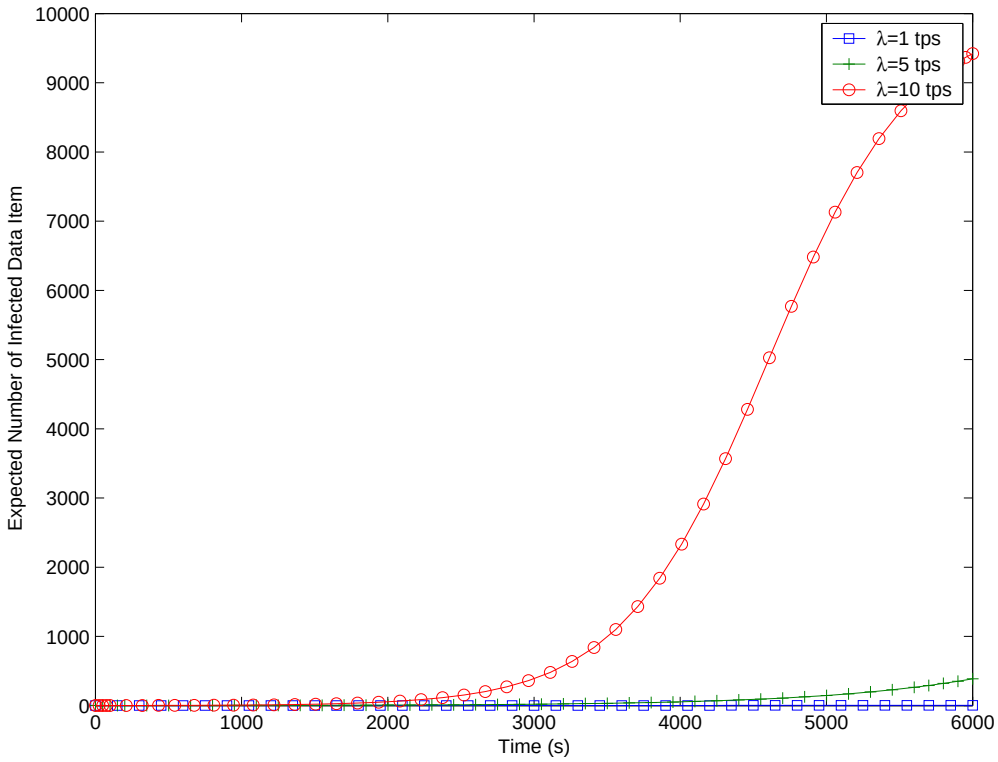
number is the number of data items that are infected by the initial intrusion. In this part, I study its impact on damage spreading.

The results in Figure 4.5 show that impact of initial infected number is not significant. As can be seen, only around 3,000 data items (30%) are infected in 100 minute when the number of initial infected data item is 10. When the number of initial infected data item is 1, only around 400 data items (4%) are infected.

The reason is that the damage in database is propagated passively by incoming good transactions. It is not like the spreading of virus or worm which actively copy themselves and propagate. The incoming transaction has a more significant effect on the damage spreading in the database. As shown in Figure 4.2 and 4.3, the transaction arrival rate and transaction size impact the damage propagation significantly.

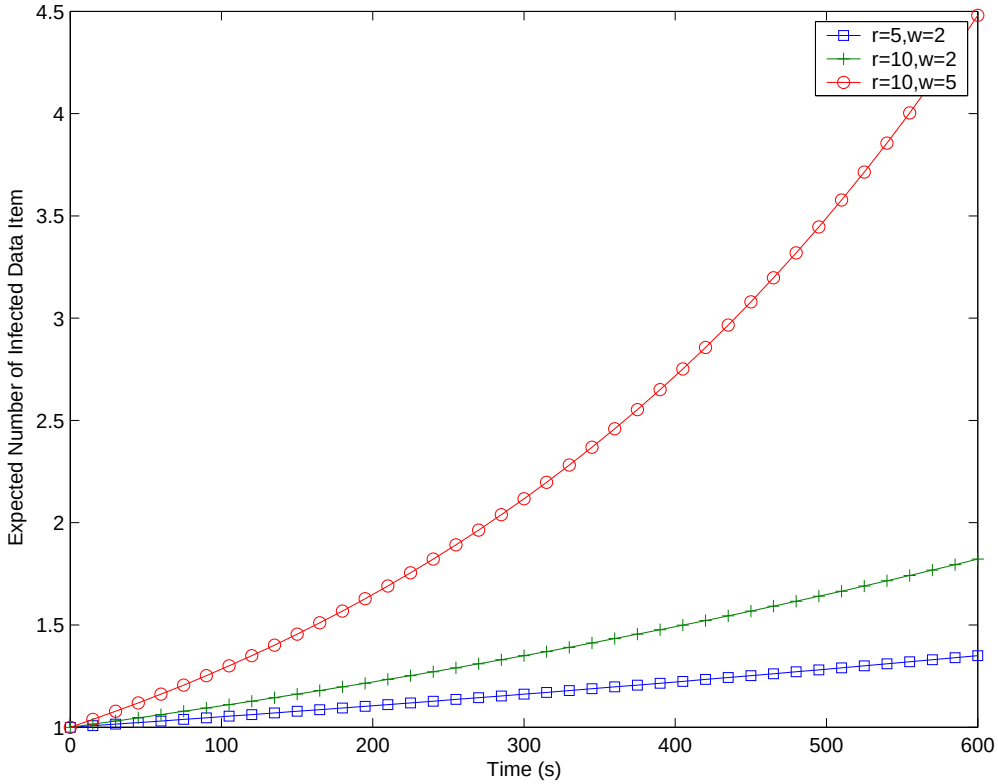


(a) $T = 10min$

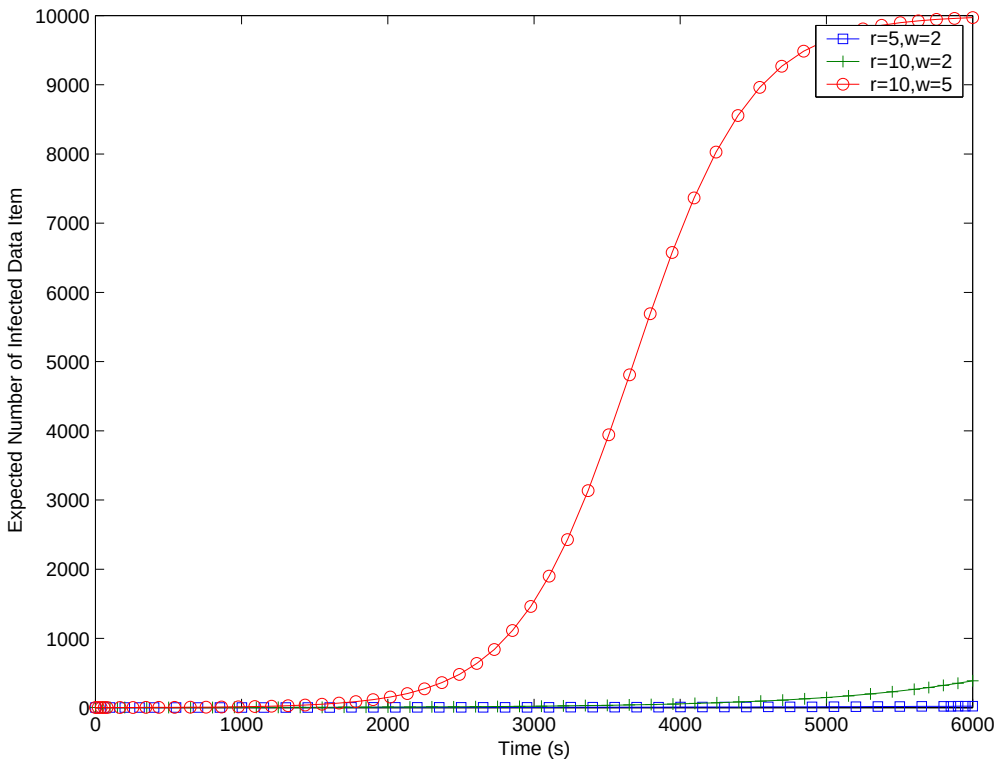


(b) $T = 100min$

Fig. 4.2. Effect of Transaction Arrival Rate λ ($r = 10$; $w = 2$; $N = 10,000$; $I_0 = 1$)

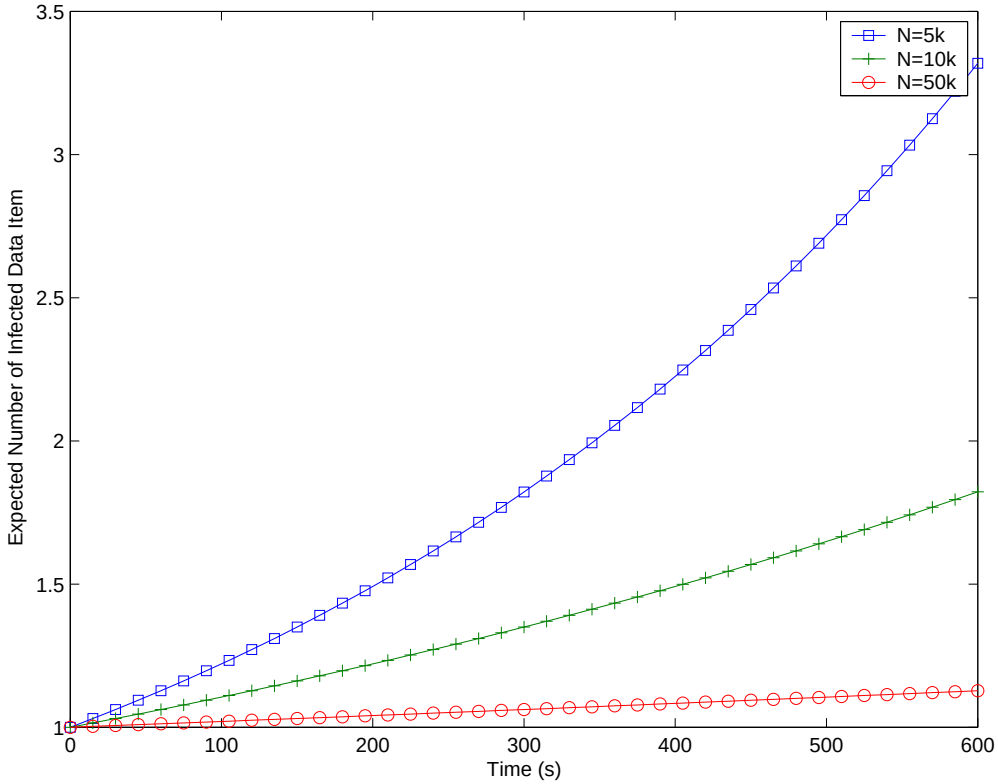


(a) $T = 10min$

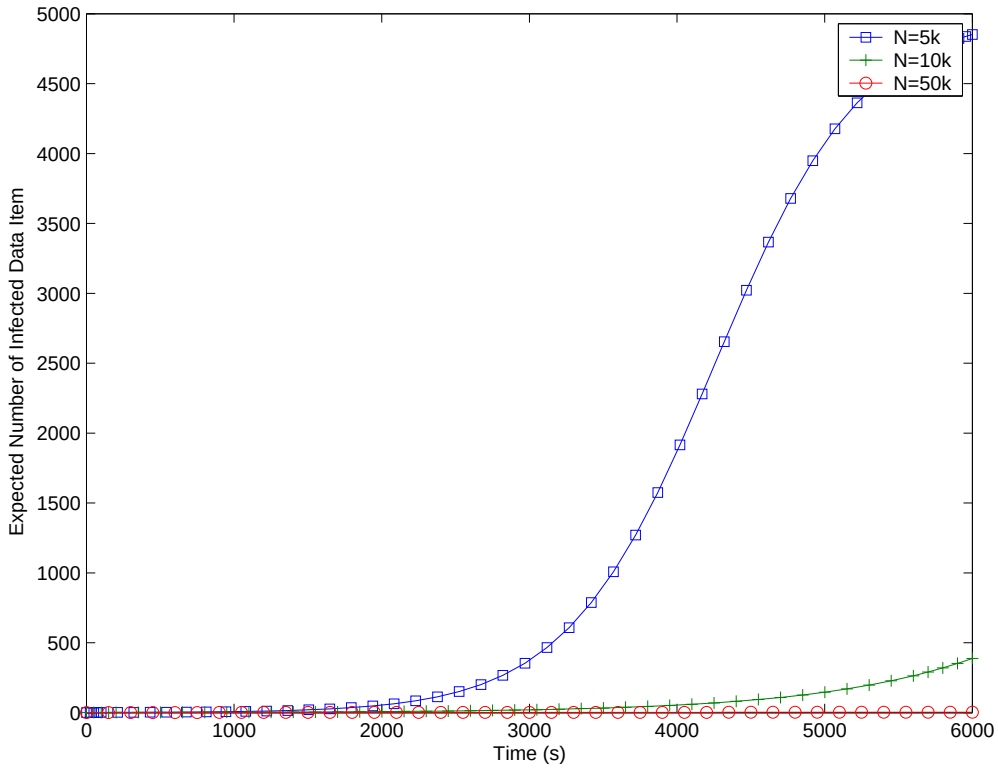


(b) $T = 100min$

Fig. 4.3. Effect of Transaction Size r, w ($\lambda = 5$; $N = 10,000$; $I_0 = 1$)

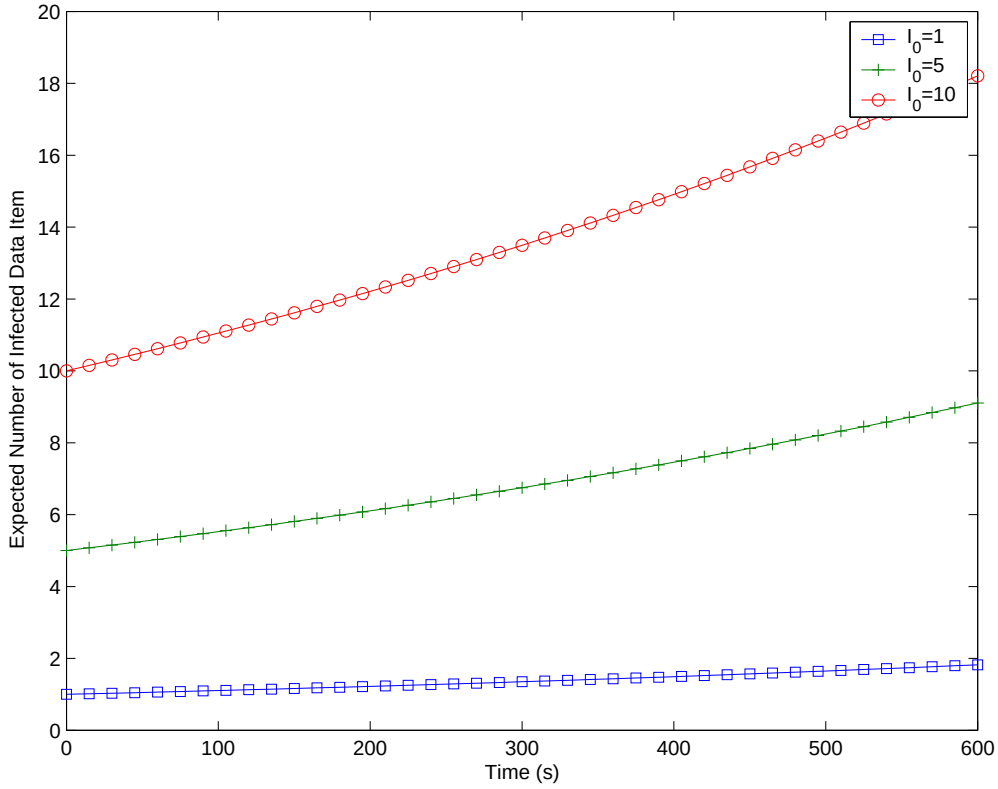


(a) $T = 10min$

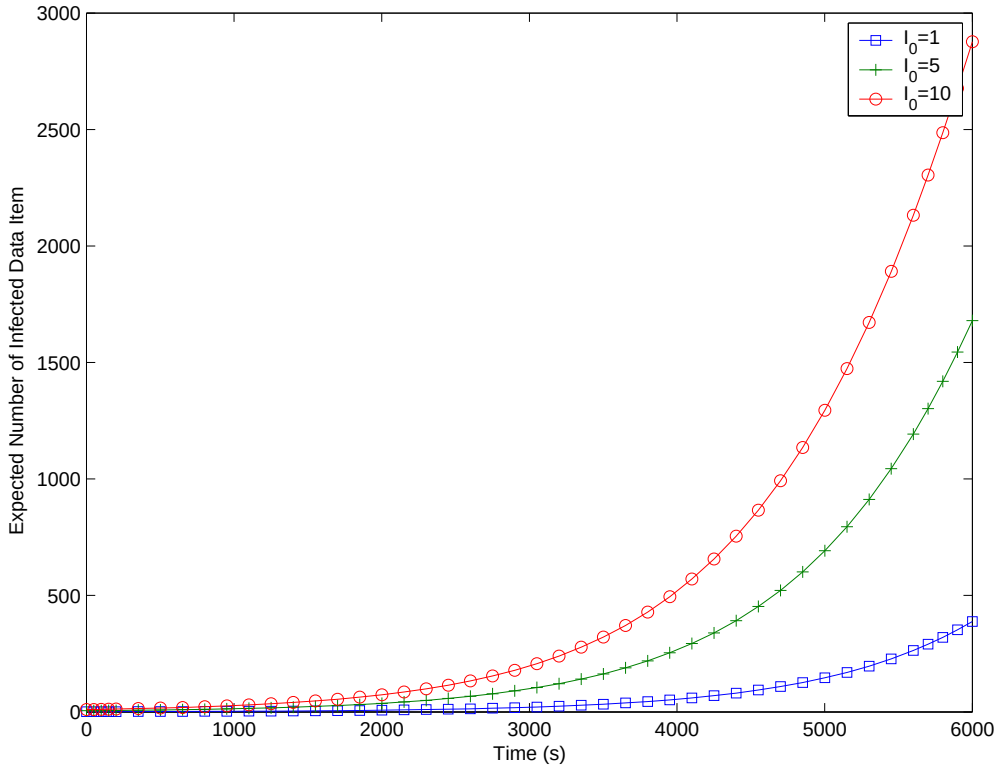


(b) $T = 100min$

Fig. 4.4. Effect of Database Size N ($\lambda = 5$; $r = 10$; $w = 2$; $I_0 = 1$)



(a) $T = 10min$



(b) $T = 100min$

Fig. 4.5. Effect of Initial Infected Number I_0 ($\lambda = 5$; $r = 10$; $w = 2$; $N = 10,000$)

Chapter 5

Modeling Intrusion Tolerant Database Systems

To analyze and evaluate the survivable capability of an intrusion tolerant system, a quantitative evaluation model is required. A variety of modeling techniques are applied in the research of survivability study.

5.1 Stochastic versus deterministic models

Computational models are generally either stochastic or deterministic. The parameters and variables in a deterministic model are not subject to random fluctuations, so that the system is at any time entirely defined by the initial conditions chosen. A complex deterministic model can predict the outcome, when the computer system is modeled in great detail.

A stochastic model takes into consideration the presence of some randomness in one or more of its parameters or variables. The predictions of the model therefore do not give a single point estimate but a probability distribution of possible estimates.

The strength of deterministic modeling is that with a good scientific understanding a good model can be constructed and it will be possible to see how it is working. The weakness is very labor intensive to develop the model. A large number of parameters will need to be collected, while a stochastic model can run with fewer parameters.

Also, deterministic models are quite limited in the stochastic behavior that they can express. All possible system states can not be captured by deterministic models. Stochastic models are much more comprehensive. They allow explicit modeling of complex relationships, and their transition structure encodes important sequencing information. Historically, stochastic methods have been explored in the context of mathematical models that specify probabilistic assumptions about time durations and transition behavior.

In this thesis, I will apply stochastic models to explore the complex relationships and transition structure of an intrusion tolerant database system.

5.2 Theoretical Preliminaries

Markov chains are important classes of stochastic processes. We describe a Markov chain as a series of states of a system, $X = \{x_1, x_2, \ldots, x_r\}$, that has the Markov property. The process starts in one of these states and moves from one state to another, or it may have stayed in the same state. Each move is called a step. If the chain is currently in state x_i, then it moves to state x_j at the next step with a probability denoted by $p_{ij}(n)$, and this probability is conditionally independent of every prior state given the current state.

The probabilities $p_{ij}(n)$ are called transition probabilities from state i to state j and from time n to time $n+1$. If $p_{ij}(n)$ does not depend on n, we say that the Markov Chain is time homogeneous. In practice, it is very difficult to estimate the parameters for a non-time-homogeneous Markov chain, therefore we always restrict ourselves to the time homogeneous chains unless otherwise mentioned.

The process can remain in the state it is in, and this occurs with probability p_{ii}. An initial probability distribution, defined on X, specifies the starting state. Usually this is done by specifying a particular state as the starting state.

Formally, a Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property, namely that, given the present state, the future and past states are independent.

$$\Pr(X_{n+1} = x | X_n = x_n, \ldots, X_1 = x_1) = \Pr(X_{n+1} = x | X_n = x_n) \tag{5.1}$$

The possible values of X_i form a countable set S called the state space of the chain.

As described in Chapter 3, it is conditionally independent of every prior state for the intrusion tolerant database system to detect the intrusions, quarantine the suspicious data, and repair the damage. The process has the Markov property. In the following sections, I will try to model the system as a Markov chain.

A Markov chain is a discrete state random process in which the only state that influences the next state is the current state. To be more precise, it can be classified as Discrete Time Markov Chain (DTMC) and Continuous Time Markov Chain (CTMC). Since the process of ITDB protecting the database system is a continuous time random process, I will model the ITDB system as a homogeneous finite state Continuous Time Markov chain (CTMC)

Markov chains are often described by a directed graph, where the edges are labeled by the probabilities of going from one state to the other states. In the following sections,

I will also use the directed graphs labeled by the probabilities to show the Markov chain models I proposed.

5.3 Basic State Transition Model

Figure 5.1 shows the basic state transition model of the intrusion tolerant database system. Traditional computer security leads to the design of systems that rely on resistance to attacks. If the strategies for resistance fail, the system is brought from good state G into the infected state I during the penetration and exploration phases of an attack. If the attack is detected successfully, intrusion tolerance system picks up where attack resistance leaves off. The system enters the quarantine state M. In this state, all suspicious data items are quarantined. After marking all the damage made by the attack, undamaged items are released and the system enters the recovery state R. Repair process will compensate all the damage and the system returns to the good state G. The four phases which are attack penetration, error detection, attack quarantine, damage assessment and error recovery, describe the basic phenomena that each intrusion tolerant system will encounter. These can and should be the basis requirement for the design and implementation of an intrusion tolerant database system.

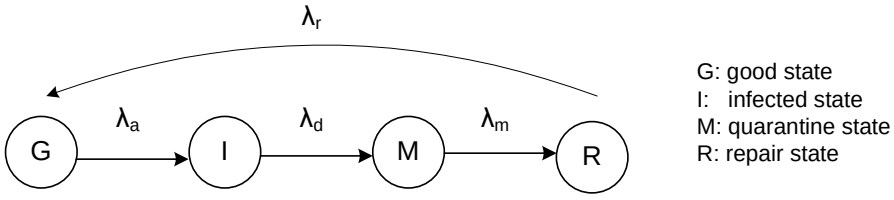


Fig. 5.1. Basic State Transition Model

Parameters in Figure 5.1 are summarized here:

- λ_a is the transition probability between State G and State I;
- $1/\lambda_a$ is the mean time to attacks (MTTA), the expected time for the system to be corrupted;
- λ_d is the transition probability between State I and State M;
- $1/\lambda_d$ is the mean time to detect (MTTD), the expected time for the intrusion to be detected;
- λ_m is the transition probability between State M and State R;
- $1/\lambda_m$ is the mean time to mark (MTTM), the expect time for the system to mark "dirty" data items;
- λ_r is the transition probability between State R and State G;
- $1/\lambda_r$ is the mean time to repair (MTTR), the expect time for the system to repair damaged data items.

5.4 Model with Intrusion Detection System

As an important part of an intrusion tolerant system, the Intrusion Detection System (IDS) is largely ignored in intrusion tolerant system modeling. [35] assumes that the IDS can report intrusion without delay or false alarm. Some work only consider part of IDS parameters, like true positive [31]. In this part, I will integrate a comprehensive model of IDS into the whole system.

False Alarm Rate (FAR) and detection probability are widely used to evaluate the performance of an IDS in either networking [49] or database field [17]. Detection latency, also called *detection time* in [50], is another metric to evaluate an IDS. I define detection latency as the the duration that elapses from the time when an attack compromises a database system successfully to the time when the IDS identifies the intrusion. All these three metrics are included in my model.

Let T_a and T_{fa}, respectively, denote the times to intrusion and the time to the failure of the IDS. If the IDS fails before the intrusion, then a false alarm is said to have occurred. Let A denote the time to failure occurrence (of either the detector or the system). Clearly,

$$A = min\{T_A, T_{fa}\} \tag{5.2}$$

I assume that T_a and T_{fa} are mutually independent and exponentially distributed with parameter λ_a and α, respectively. Then, clearly, A is exponentially distributed with parameter $\lambda_a + \alpha$ and

$$F_A(t) = 1 - e^{-(\lambda_a+\alpha)t} \tag{5.3}$$

A successful attack may seriously impair the integrity and availability of a database in depth. After the intrusion, it takes a finite time T_d (*detection latency*) to detect the intrusion. I assume that the time to identify one successful intrusion is exponentially distribution with parameter λ_d.

$$F_D(t) = 1 - e^{-\lambda_d t} \tag{5.4}$$

For the imperfect detection, I assume that all attacks will be identified by the database administrator eventually. I use state MD and MR to represent the *undetected state* and *manual repair state* respectively. Let us assume that the detection probability of an IDS to identify a successful intrusion is d. The transition probability that the system transfers from state I to state MD is $(1-d)$. I assume that the time to manually identify a successful intrusion is exponentially distributed with parameter λ_{md} and the time to manually repair infected data items is exponentially distributed with parameter λ_{mr}. The state transition model considering the deficiencies of the IDS is presented in Figure 5.2.

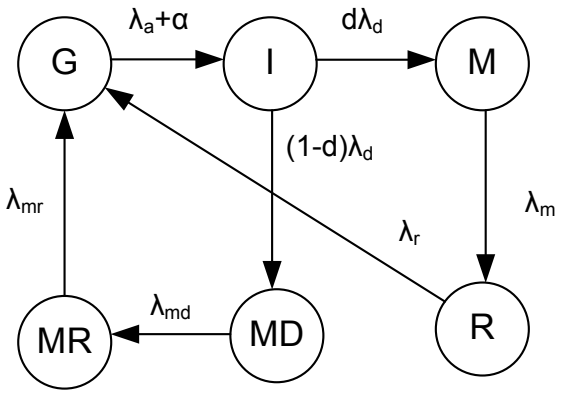


Fig. 5.2. State Transition Model with IDS

5.5 Comprehensive State Transition Model

The damage keeps propagating the effect of the intrusion during the detection phase. Longer detection delay may bring wider damage and take more time to repair. The purpose of the IDS is to reduce the its detection latency T_d. Although damage spreading is a normal phenomenon, little work puts effort on studying the effect of damage propagation on damage assessment and compensation in their intrusion tolerant system models. In this section, I want to take the first step to study the effect of detection delay on damage propagation, which may affect damage assessment and repair correspondingly.

Let T_{di} denotes the time between the infection of $(i-1)$th and ith data item. Obviously,

$$T_d = \sum_{i=1}^{k} T_{di} \tag{5.5}$$

where k is the number of infected data items during the detection latency. Let assume that T_{di} exponentially distributed with parameter λ_{di} and

$$F_{Di}(t) = 1 - e^{-\lambda_{di} t} \tag{5.6}$$

As soon as the intrusion is identified, the quarantine phase instantly quarantines the damage that might have been caused by an intrusion. At the same time, the damage assessment process begins to scan the quarantined data items and locate the infected

ones. I assume that the time to scan one infected data items is exponentially distributed with parameter λ_m.

$$F_M(t) = 1 - e^{-\lambda_m t} \tag{5.7}$$

After all infected data items are identified via the damage assessment process, the repair system begins to compensate the damage caused by the intrusion. I assume the time to repair one infected data item is exponentially distributed with parameter λ_r

$$F_R(t) = 1 - e^{-\lambda_r t} \tag{5.8}$$

Let $(I : k)$ denotes the *infect state* with k infected data items in the database, and $(M : k)$ denotes the *mark state* with k infected data need to be located. Figure 5.3 shows the comprehensive state transition model of ITDB.

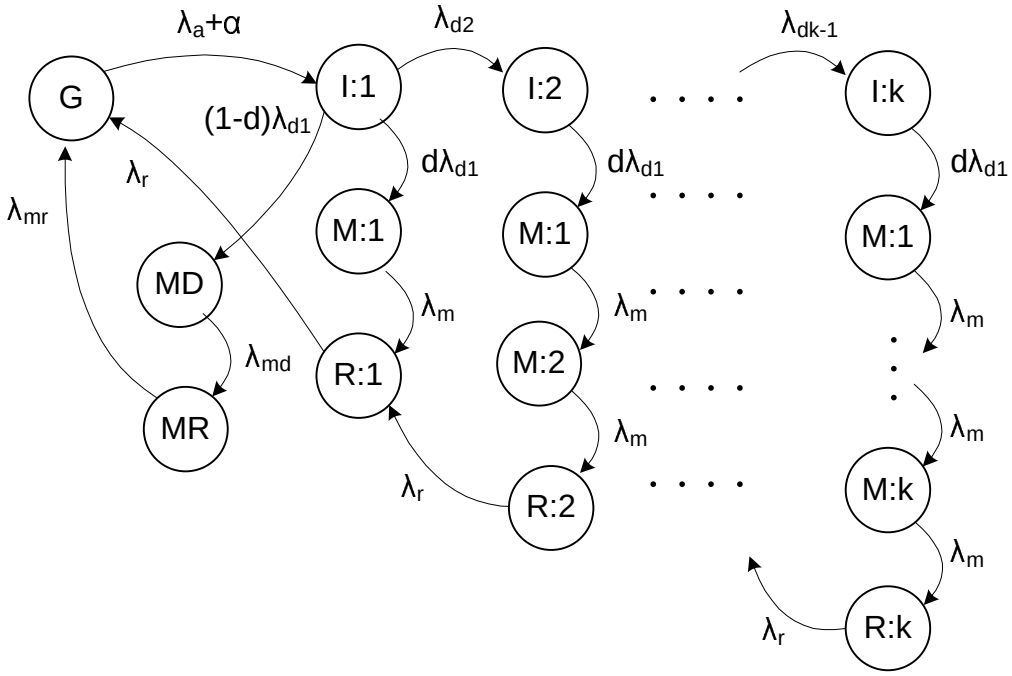


Fig. 5.3. Comprehensive State Transition Model

Chapter 6

Survivability Evaluation

Evaluation criteria in trustworthy data processing systems are often referred to as *survivability* or *trustworthiness*. Survivability refers to the capability of a system to complete its mission, in a timely manner, even if significant portions are compromised by attacks or accidents [13]. However, in the context of different types of systems and applications, it can mean many things. This brings a difficulty in the measurement and interpretation of survivability. For a database system, survivability is the quantified ability of a system or subsystem to maintain the integrity and availability of essential data and information, such as account and loan data, and services, such as transaction validation and processing. And also a survivable database system could maintain the performance of essential services facing attacks.

In this chapter, based on the models established in Chapter 5, quantitative metrics are proposed to facilitate evaluating the survivability of intrusion tolerant database systems from several aspects.

6.1 State Transition Model Analysis

In the previous chapter, several Markov chains were proposed to model the ITDB system. In this section, I analyze the established models.

Let $\{X(t), t \geq 0\}$ be a homogeneous finite state continuous time Markov chain (CTMC) with state space S and generator matrix $\mathbf{Q} = [q_{ij}]$. Let $P_i(t) = P\{X(t) = i, i \in S\}$ denote the unconditional probability that the CTMC will be in state i at time t, and the row vector $\mathbf{P}(t) = [P_1, P_2, \cdots, P_n]$ represent the transient state probability vector of the CTMC. The transient behavior of the CTMC can be described by the Kolmogorov differential equation:

$$\frac{d\mathbf{P}(t)}{dt} = \mathbf{P}(t)\mathbf{Q} \tag{6.1}$$

where $\mathbf{P}(0)$ represents the initial probability vector (at time $t = 0$).

In addition, cumulative probabilities are sometimes of interest. Let $L(t) = \int_0^t P(u)du$; then, $L_i = (t)$ represents the expected total time the CTMC spends in state i during the interval $[0, t)$. $L(t)$ satisfies the differential equation:

$$\frac{d\mathbf{L}(t)}{dt} = \mathbf{L}(t)\mathbf{Q} + \mathbf{P}(0), \tag{6.2}$$

where $\mathbf{L}(0) = 0$.

The steady-state probability vector $\pi = lim_{t\to\infty}\mathbf{P}(t)$ satisfies:

$$\pi\mathbf{Q} = 0 \quad \sum_{i\in S} \pi_i = 1 \tag{6.3}$$

By solving the Equation 6.1, 6.2 and 6.3, we can get the some survivable metrics of an intrusion tolerant system.

6.2 Evaluation Metrics

In my model, survivability is quantified in terms of *Integrity* and *Availability*. According to survivability, I define integrity in a way different from integrity constrains. In this thesis, I define Integrity as follow:

Definition. ***Integrity*** *is defined as a fraction of time that all accessible data items in the database are clean.*

High Integrity means that the intrusion tolerant system can serve the users good or clean data at a high probability. Obviously, all data items are clean and accessible in state G. When attacks occur, some data items will be affected. So in state I, part of accessible data items are "dirty". After the intrusion is identified, the ITDB can quarantine all the damaged data until finished the repair process. Since the ITDB does selective quarantine and on-the-fly repair, the database system is still available, and accessible data are clean during the quarantine, damage assessment, and repair process.

Consider the model in Figure 5.1, state space $S = \{G, I, M, R\}$. The generator matrix $\mathbf{Q}$ for the basic state transition model in Section 5.3:

$$\mathbf{Q} = \begin{bmatrix} -\lambda_a & \lambda_a & 0 & 0 \\ 0 & -\lambda_d & \lambda_d & 0 \\ 0 & 0 & -\lambda_m & \lambda_m \\ \lambda_r & 0 & 0 & -\lambda_r \end{bmatrix} \tag{6.4}$$

The generator matrix P for my model in Figure 5.3 is big and complicated. Here I show a simple generator matrix where $k = 2$ for the model in Figure 5.3:

$$P = \begin{pmatrix}
-(\lambda_a + \alpha) & \lambda_a + \alpha & & & & & & & \\
 & -\lambda_{d1} & d\lambda_{d1} & & & & & (1-d)\lambda_{d1} & \\
 & & -(\lambda_{d2} + \lambda_s) & \lambda_{d2} & & \lambda_s & & & \\
 & & & -\lambda_s & \lambda_s & & & & \\
 & & & & -\lambda_s & & \lambda_s & & \\
\lambda_r & & & & & -\lambda_r & & & \\
 & & & & & \lambda_r & -\lambda_r & & \\
 & & & & & & & -\lambda_{md} & \lambda_{md} \\
\lambda_{mr} & & & & & & & & -\lambda_{mr}
\end{pmatrix} \tag{6.5}$$

By solving the Equation 6.2 and 6.3, we can get:

$$
\begin{aligned}
\pi_G &= \frac{1/\lambda_a}{1/\lambda_a + 1/\lambda_d + 1/\lambda_m + 1/\lambda_r} \\
&= \frac{MTTA}{MTTA + MTTD + MTTM + MTTR} \\
\pi_I &= \frac{1/\lambda_d}{1/\lambda_a + 1/\lambda_d + 1/\lambda_m + 1/\lambda_r} \\
&= \frac{MTTD}{MTTA + MTTD + MTTM + MTTR} \\
\pi_M &= \frac{1/\lambda_m}{1/\lambda_a + 1/\lambda_d + 1/\lambda_m + 1/\lambda_r} \\
&= \frac{MTTM}{MTTA + MTTD + MTTM + MTTR} \\
\pi_R &= \frac{1/\lambda_r}{1/\lambda_a + 1/\lambda_d + 1/\lambda_m + 1/\lambda_r} \\
&= \frac{MTTR}{MTTA + MTTD + MTTM + MTTR}
\end{aligned}
\tag{6.6}
$$

From the Definition 1, we can get the Integrity for the basic state transition model in Section 5.3:

$$
\begin{aligned}
I &= \pi_G + \pi_M + \pi_R \\
&= \frac{MTTA + MTTM + MTTR}{MTTA + MTTD + MTTM + MTTR}
\end{aligned}
\tag{6.7}
$$

Similarly, we can get the Integrity for the comprehensive state transition model I proposed in Section 5.5:

$$I = \pi_G + \sum_{i=1}^{k} \pi_{M_i} + \sum_{i=1}^{k} \pi_{R_i} \tag{6.8}$$

Availability [10] is defined as the fraction of time that the system is providing service to its users. Since the ITDB does on-the-fly repair and will not stop its service facing attacks, its availability is nearly 100%, which can not show the performance of ITDB clearly. To better evaluate the survivability of ITDB, I define another type of availability, Rewarding-availability:

Definition. ***Rewarding-availability (RA)*** *is defined as a fraction of time that the all clean data items are accessible.*

If the clean data can not be accessed, it is a loss of service to users. Rewarding-availability means that the system not only can serve its users, but also does not deny the request for the clean data. ITDB will release all the quarantined clean data items after damage assessment. For the basic state transition model in Section 5.3, the Rewarding-availability is:

$$\begin{aligned} RA &= \pi_G + \pi_R \\ &= \frac{MTTA + MTTR}{MTTA + MTTD + MTTM + MTTR} \end{aligned} \tag{6.9}$$

The Rewarding-availability for the comprehensive state transition model in Section 5.5 is:

$$RA = \pi_G + \sum_{i=1}^{k} \pi_{R_i} \tag{6.10}$$

Chapter 7

Semi-Markov Survivability Model

The Markov chain models proposed in Chapter 6 assume that the distributions for transitions between states are exponential. However, in some real cases the attack and repair time are not exponentially distributed. Semi-Markov process is an extension of Markov chains. In Markov environment, the distributions for transitions between states have to be negative exponential functions. Instead in the semi-Markov case the distributions can be nonexponential functions. That is the main advantage of semi-Markov processes. In addition, in Markov chain environment the time transition is given. But in the reality, the transition between two states in a mechanical system sometimes happens after a random duration. This is one reason why the semi-Markov chain model fits better than the Markov one in real problems.

In many engineering problems, especially in dependability, reliability, availability, maintainability, safety, performability analysis [51, 52, 53, 54, 55], semi-Markov processes are used. However, little effort has been made to apply semi-Markov process to model the survivability of a security system. In this chapter, I will model the survivability of intrusion tolerant database systems whose stochastic behavior is given by a semi-Markov process with a finite state space.

The numerical computation of semi-Markov process is unequally hard for the time-dependent and for the steady-state probabilities. The calculus of steady-state probabilities is very easy and requires only the computation of the mean time spent in each state and of the invariant distribution of the embedded Markov chain in the semi-Markov process. Concerning the time-dependent probabilities, the computation is more difficult than for Markov systems because of convolution products, i.e., matrices inversion in the convolution sense.

7.1 Theoretical Preliminaries

In this section, the main results regarding continuous time homogeneous and non-homogeneous semi-Markov processes are reported. The notation adopted follows that of [56].

Let $E = \{1, ..., m\}$ be the state space and let (Ω, F, P) be a probability space. Let us also define the following random variables:

$$X_n : \Omega \to E, \quad T_n : \Omega \to \mathbb{N}$$

where X_n represents the state at the $n-th$ transition and T_n represents the time of the $n-th$ transition. The (X, T) process is called "*homogeneous Markov renewal process*".

The associated *homogeneous semi-Markov* kernel Q is defined by:

$$\begin{aligned} Q_{ij}(t) &= P[X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_0, ..., X_n; T_0, ..., T_n] \\ &= P[X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i] \end{aligned} \tag{7.1}$$

As is well known,

$$p_{ij} = \lim_{t \to \infty} Q_{ij}(t) \quad i, j \in E$$

and $P = [p_{ij}]$ is the transition probability matrix of the embedded Markov chain.

$$S_i(t) = P[T_{n+1} - T_n \leq t \mid X_n = i]$$

gives the probability that the process will leave state i in a time t. Of course,

$$S_i(t) = \sum_{j \neq t}^{m} Q_{ij}(t) \tag{7.2}$$

Now it is possible to define the conditional distribution functions (CDF) of the waiting time in each state i, given the state subsequently occupied:

$$H_{ij}(t) = P[T_{n+1} - T_n \leq t \mid X_n = i, X_{n+1} = j].$$

The related probabilities are obtained by:

$$H_{ij}(t) = \begin{cases} \frac{Q_{ij}(t)}{p_{ij}}, & if \quad p_{ij} \neq 0; \\ U_1(t) & if \quad p_{ij} = 0 \end{cases} \tag{7.3}$$

where

$$U_1(t) = 1 \quad \forall t \geq 0$$

We suppose, without loss of generality, that the waiting time also has a probability density function.

Now it is possible to define the homogeneous semi-Markov process (HSMP) Z.

$$Z = (Z_t \quad t \in \mathbb{N}) \tag{7.4}$$

representing, for each time t, the state occupied by the process.

The transition probabilities are defined in the following way:

$$\phi_{ij}(t) = P[Z_t = j \mid Z_0 = i] \tag{7.5}$$

which are obtained by means of the following evolution equations:

$$\phi_{ij}(k) = (1 - S_i(t))\delta_{ij} + \sum_{l=1}^{m} \int_0^t Q_{il}(\tau)\phi_{lj}(t-\tau)d\tau. \tag{7.6}$$

where δ_{ij} represents the Kronecker δ.

The (X, T) process will be called "*non-homogeneous Markov renewal process*". The *non-homogeneous semi-Markov* kernel Q is defined by:

$$Q_{ij}(s,t) = P[X_{n+1} = j, T_{n+1}t \mid X_n = i, T_n = s] \tag{7.7}$$

Also in this case,

$$p_{ij}(s) = \lim_{t\to\infty} Q_{ij}(s,t) \quad i,j \in E \tag{7.8}$$

and $P(s) = [p_{ij}(s)]$ is the transition matrix of the embedded non-homogeneous Markov chain.

$$S_i(s,t) = P[T_{n+1} \leq t \mid X_n = i, T_n = s],$$

represents the probability that the process will leave state i from s to t. Of course,

$$S_i(s,t) = \sum_{j\neq i}^{m} Q_{ij}(s,t) \tag{7.9}$$

The conditional distribution functions (CDF) of the waiting time in each state i , given the state subsequently occupied, are defined by:

$$H_{ij}(s,t) = P[T_{n+1} \leq t \mid X_n = i, X_{n+1} = j, T_n = s].$$

The related probabilities are obtained by:

$$H_{ij}(s,t) = \begin{cases} \frac{Q_{ij}(s,t)}{p_{ij}(s)}, & if \quad p_{ij}(s) \neq 0; \\ U_1(t) & if \quad p_{ij}(s) = 0 \end{cases} \tag{7.10}$$

Also in this case, we suppose that the waiting time has a probability density function. In non-homogeneous case the transition probabilities of the Equation 7.4 are defined in the following way:

$$\phi_{ij}(s,t) = P[Z_t = j \mid Z_s = i],$$

which are obtained by means of the following evolution equations:

$$\phi_{ij}(u,k) = (1 - S_i(s,t))\delta_{ij} + \sum_{l=1}^{m} \int_s^t Q_{il}(s,\tau)\phi_{lj}(\tau,t)d\tau. \tag{7.11}$$

7.2 Semi-Markov Model Analysis

Suppose that the semi-Markov process is irreducible and positive recurrent, with steady-state probability π on E. Let $P = (p_{ij})$, where $p_{ij} = \lim_{t\to\infty} Q_{ij}(t)$, be the transition probability matrix of the Markov chain (J_n)corresponding to the semi-Markov process, and let v be its stationary distribution.

Let h_i be the mean sojourn time in state i, where

$$h_i = \mathbb{E}[S_1 \mid J_0 = i] = \int_0^\infty (1 - H_i(t))dt$$

and set $\mathbf{h} = (h_1, h_2, ..., h_s)'$. We suppose here that for all $i \in E$, $h_i < \infty$

The relationship between π and v is given by

$$\pi = \frac{1}{v\mathbf{h}} diag(v)\mathbf{h} \tag{7.12}$$

In addition, we know that

$$vP = v, \quad \sum_{i \in E} v_i = 1 \tag{7.13}$$

By solving the above equation, we can get the steady-state probability of occupying a particular state in the semi-Markov chain.

7.3 Evaluation Metrics

7.3.1 Integrity

Consider the comprehensive transition model presented in Chapter 5 with state space E, I partition the state set into two set: I_0 and I_1, where

$$I_0 = \{G, M_i, R_i\} \quad E = I_0 \cup I_0 \quad I_0 \cap I_0 = \varnothing$$

The subset I_0 contains all "integrity" states in which all accessible data items are clean, while subset I_1 contains all states in which not all accessible data items are clean.

The Integrity for the comprehensive state transition model is

$$I = \pi_G + \sum_{i=1}^{k} \pi_{M_i} + \sum_{i=1}^{k} \pi_{R_i} \tag{7.14}$$

7.3.2 Rewarding-Availability

Consider the comprehensive transition model presented in Chapter 5 with state space E, the state set is partitioned into two set: RA_0 and RA_1, where

$$RA_0 = \{G, M_i, R_i\} \quad RA_i = \{I_i\}$$

The subset RA_0 contains all "rewarding-available" states in which all clean data items are accessible, while subset RA_1 contains all states in which not all clean data items are accessible.

The Rewarding-availability for the comprehensive state transition model is:

$$RA = \pi_G + \sum_{i=1}^{k} \pi_{R_i} \tag{7.15}$$

Chapter 8

The Testing Environment

To validate the models I proposed in the previous chapters, a real testing environment is needed to conduct comprehensive validation experiments. However, most existing work on security system modeling is based on simulation, which have limitation in modeling realistic workloads and real system dynamics. In [31],[33],[35], a security intrusion and the response of different intrusion tolerant systems are modeled as random processes. However, no empirical experiments are conducted to validate the proposed model. The performance evaluation is only based on simulation results. Little prior work has been done to build up real testbed systems to evaluate a security system. In [57], an intrusion detection evaluation testbed is established, which can support real-time, automated and quantitative evaluations of ID systems and other information assurance (IA) technologies. But this system focuses more on a network system, not a database system. In [58, 59], real-time database testbed is developed to evaluate the database system performance. However, the testbed is not designed to evaluate the survivability of a database system and no security-related experiment is conducted in this testbed.

In this chapter, I take the first step to build a real-time testbed to validate the models I proposed. First, I implement a real-time multi-user relational database system which provides an operational platform for my research. The TPC-c benchmark [60] was

chosen as background transaction. A representative intrusion tolerant database system, ITDB, is implemented as an empirical example.

8.1 Background Transaction and Generation

Background transactions were generated in the testbed for a variety of reasons. These transactions made it possible to measure the performance of evaluated intrusion tolerant database systems. We can also use these transactions to deter the development of limited non-robust intrusion tolerant systems that simply trigger when a particular transaction type occurs. The synthesized nature of the transactions allows widespread and relatively unrestricted access to the items in the database. The approach taken in the testbed was to generate realistic transactions that are roughly similar to transactions in a real world database system. In addition, these transactions are designed to produce transaction arriving rate similar to workload in a real world database system.

8.1.1 TPC-C Benchmark

TPC Benchmark C is an on-line transaction processing (OLTP) benchmark. TPC-C is more complex than previous OLTP benchmarks such as TPC-A because of its multiple transaction types, more complex database and overall execution structure. TPC-C involves a mix of five concurrent transactions of different types and complexity either executed on-line or queued for deferred execution. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. Table 8.1 summarizes the transaction mix rates among my background transactions.

Table 8.1. Summary of TPC-C's Transactions

Transaction	Min. %	Profile	Description
New-Order	-	Mid-weight, read-write, online response time requirement	Initiates an order for an average of 10 items.
Payment	43%	Light-weight, read-write, online response time requirement	Updates the customer's balance and reflects the payment on the district and warehouse sales statistics.
Order-Status	4%	Mid-weight, read-only, online response time requirement	Queries the status of a customer's last order.
Delivery	4%	Mid-weight, read-write, relaxed response time requirement	Processes a batch of 10 new orders, one for each district for a given warehouse.
Stock-Level	4%	Heavy, read-only, relaxed response time requirement	Counts the number of items in the last 20 orders in a district that fall below the stock threshold.

In the TPC-C business model, a wholesale parts supplier (called the Company below) operates out of a number of warehouses and their associated sales districts. The TPC benchmark is designed to scale just as the Company expands and new warehouses are created. However, certain consistent requirements must be maintained as the benchmark is scaled. Each warehouse in the TPC-C model must supply 10 sales districts, and each district serves 3,000 customers. An operator from a sales district can select, at any time, one of the five operations or transactions offered by the Company's order-entry system. Like the transactions themselves, the frequency of the individual transactions are modeled after realistic scenarios. The following diagram illustrates the warehouse, district, and customer hierarchy of TPC-C's business environment

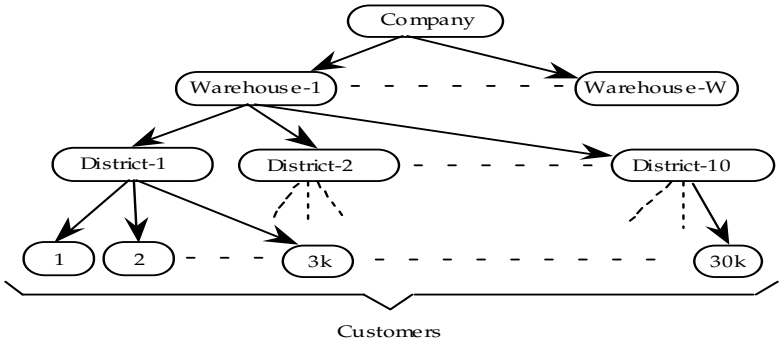


Fig. 8.1. TPC-C Business Environment

The most frequent transaction consists of entering a new order which, on average, is comprised of ten different items. Each warehouse tries to maintain stock for the 100,000 items in the Company's catalog and fill orders from that stock. However, in reality, one warehouse will probably not have all the parts required to fill every order. Therefore, TPC-C requires that close to ten percent of all orders must be supplied by another warehouse of the Company. Another frequent transaction consists in recording a payment received from a customer. Less frequently, operators will request the status of a previously placed order, process a batch of ten orders for delivery, or query the system for potential supply shortages by examining the level of stock at the local warehouse. A total of five types of transactions, then, are used to model this business activity.

The TPC-C database is comprised of nine separate and individual tables with a wide range of record and population sizes. The TPC-C Database Schema is shown in Figure 8.2. TPC-C is measured in transactions per minute (tpmC).

I chose TPC-C as transaction application for several reasons. First, TPC-C provides a standard transaction model, which simulates a complete computing environment

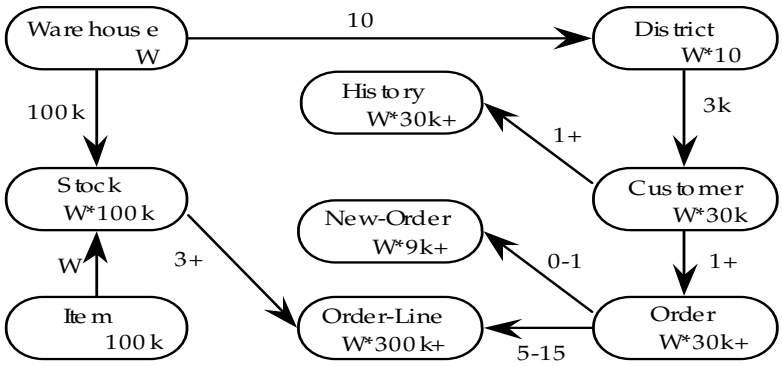


Fig. 8.2. TPC-C Database Schema

where users execute transactions against a database. Second, TPC-C transactions are highly correlated as shown in Figure 8.3[1]. So the damage may be spread widely by some types of transactions, like New-Order, Payment, and Delivery Transactions. Third, the number of warehouses, w, illustrates the database scaling. I set the warehouse number to be 5 ($w = 5$). In this way, the application will handle millions of records. Table 8.2 shows the data population. My database contains roughly 2.6 million data records.

Table 8.2. Database Population

Relationship	Cardinality
Warehouse	5
District per warehouse	10
Customer per district	3,000
Order per customer	1
Stock per warehouse	100,000

[1]For more details about the TPC-C Transactions Dependent Relations, please refer to Appendix B

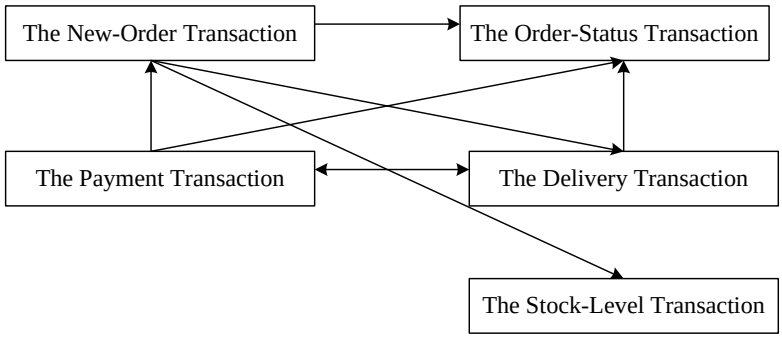


Fig. 8.3. TPC-C Transactions Dependent Relations

8.1.2 Transaction Generation

I created a simulator process, called Transaction Simulator, to compose and submit TPC-C transactions. The transaction input arguments are generated by the simulator process automatically. Using the Transaction Simulator, we can instrument the parameters used in the experiments, like the size of transactions, the number of transactions, and the number of concurrent users.

Malicious transactions are also generated by the Transaction Simulator. I simulate a malicious transaction by composing a transaction with an abnormal input. For example, to simulate a malicious Payment transaction, I compose the transaction with a large payment amount.

8.2 ITDB implementation

The current version of ITDB has around 30,000 lines of (multi-threaded) C++ code and Oracle PL/SQL code. Each ITDB component is implemented as a set of C++ objects that have a couple of CORBA calling interfaces through which other components

can interact with the component and the reconfiguration can be done. I use ORBacus V4.0.3 as the ORB. Finally, ITDB assumes applications use OCI calls, a standard interface for Oracle, to access the database, and ITDB proxies transactions at the OCI call level. The reason that ITDB does not proxy transactions at the TCP/IP or the SQL * NET level, which are more general, is because the exact packet structure of SQL * NET is confidential. Figure 8.4 shows how I implement the ITDB system.

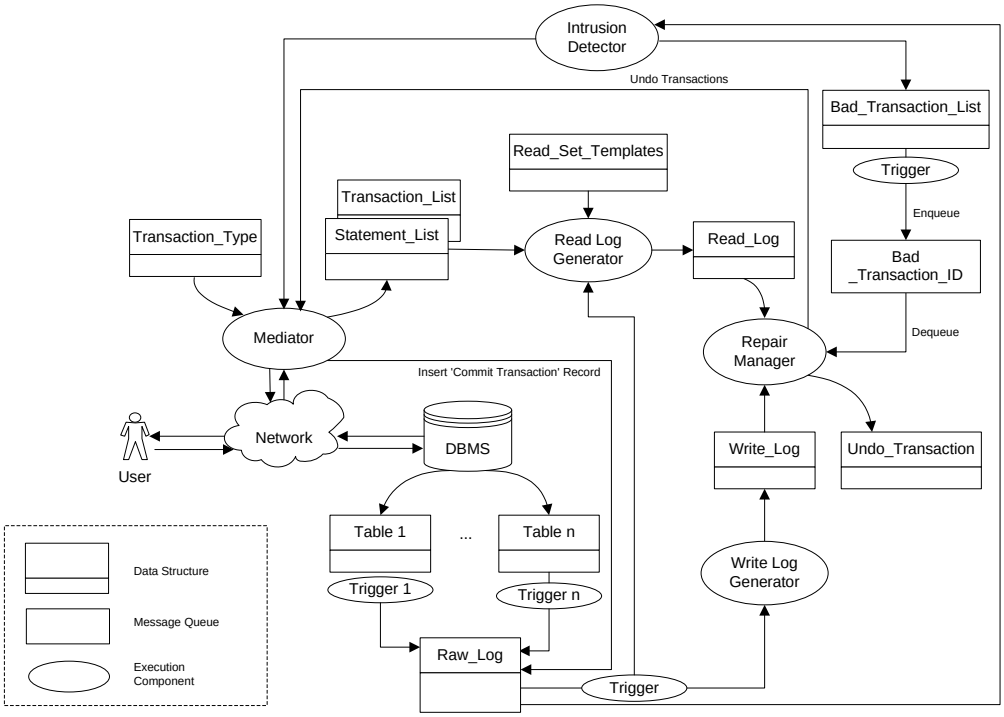


Fig. 8.4. ITDB implementation

8.3 Intrusion Detection System Simulation

In this section, I study some parameters which represent the performance of an Intrusion Detection Systems (IDS). By changing the values of these parameters within a certain range in different experiments, I can simulate different capabilities of IDSs to a large extent.

There are many works on evaluating intrusion detection systems [61, 62]. ITDB builds its intrusion detection subsystem by applying existing anomaly detection techniques. So here I will not put efforts into evaluating the performance of intrusion detection system, instead I will focus on evaluating the impact of intrusion detection deficiencies on survivability. In my experiments, I choose performance parameter values of IDS based on the typical performance of some real IDSs [63, 27]. Actually, for each IDS performance parameter, I have tested an extended range of values (larger than the typical range observed in real world) to challenge the survivability of ITDB under even more deficient IDSs.

Intrusion detection, which is the art of detecting inappropriate, incorrect, or anomalous activity, is a critical topic in security area. NIST define intrusion detection as "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network" ([64]).

There are several parameters which can evaluate or characterize the performance of IDS. False Positives (FP) and True Positives (TP) are essential to this process. False

Positives refer to the innocent events that an IDS erroneously classifies as intrusive. True Positives are intrusion attempts an IDS reported correctly. The decrease of False Positive and the increase of True Positive are a critical objective in intrusion detection. In this thesis, I will use False Alarm Rate (FAR) and Detection Rate (DR), instead of False Positive Rate and True Positive Rate, as shown in Table 8.3.

Table 8.3. IDS Parameters

Parameters	Meaning
Miss Rate	False Negative Rate
False Alarm Rate (FAR)	False Positive Rate
Detection Rate (DR)	True Positive Rate
Normal Rate	True Negative

Another important parameter to evaluate an IDS, Detection Latency (DL), has been overlooked by many researchers. Detection Latency denotes the delay for IDS to recognize intrusion and raise alarms. Long delay may increase the damage spreading exponentially and bring heavy workloads to the recovery subsystem. Having a shorter Detection Latency is another critical objective for intrusion detection.

The performance parameters of an IDS are not independent of each other. For anomaly detection techniques, intrusion detection is based on monitoring an activity and comparing it with established normal behaviors. A long detection latency will give IDS more time to analyze the activity. In this way, if the system prolongs Detection Latency, False Alarm Rate will typically decrease and Detection Rate will typically increase. On the other hand, the risk of leaving some intrusions undetected will typically increase if Detection Latency is shortened.

8.4 Evaluation Testbed

The test-bed consists of a single database server that serves two clients. Oracle Database *9i* runs on a Dell workstation as the server. The transaction proxy, intrusion detection, attack repair, and damage quarantine subsystems of ITDB also run on the server. The Transaction Simulators, which are used to generate user transactions, run on two clients. All the computers are connected by a 10/100Mbps switch LAN. Figure 8.5 shows a conceptual view of the evaluation testbed that I created. The detailed setting of the experimental testbed is shown in Table 8.4.

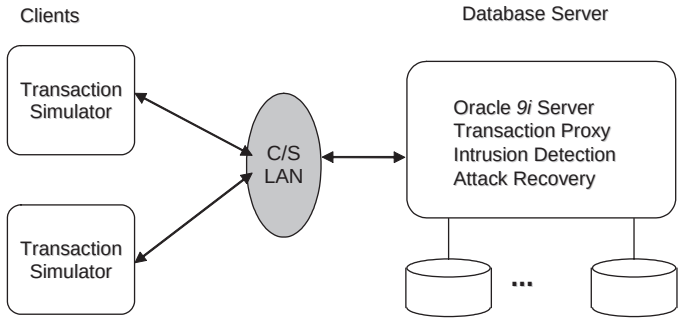


Fig. 8.5. Evaluation Testbed Structure

Table 8.4. System specification

	Server	Client 1	Client 2
CPU	Pentium 4 3.2GHz	Pentium 4 2.0GHz	Pentium 4 2.0GHz
Memory	1GB	512MB	512MB
Storage	150GB	40GB	40GB
OS	Window XP Pro	Window XP Pro	Window XP Pro

Table 8.5 gives the values of the simulation parameters that all of my experiments have in common (except where otherwise noted). Parameters that vary from experiment to experiment are not listed in Table 8.5 and will instead be given with the description of the relevant experiments.

Table 8.5. Evaluation Parameter Setting

Parameter	Meaning	Value
DBMS	Database Management System	Oracle 9i
db_size	Number of objects in database	14,789 blocks (2.6M rows)
block_ size	Size of block	8192 Bytes
max_size	Size of largest transaction	20 row read, 20 row write
min_size	Size of smallest transaction	3 row read, 4 row write
write_prob	Pr(write X| read X)	70%
num_terms	Number of terminals	2 terminals
num_cpu	Number of CPUs	1 CPU
num_disk	Number of disks	1 disk

In choosing parameter values for my experiments, I want to choose database and transaction sizes that would jointly yield a "cluster" of operations that would allow the interesting effects to be observed without requiring impossibly long simulation time.

Chapter 9

Empirical Validation

Model validation is possibly the most important step in the model building sequence. It is also one of the most overlooked. In the survivability research literature [13, 30, 33, 31], model validation is largely missing. Validation can ensure that the models meet their intended requirements in terms of the methods employed and the results obtained. The goal of model validation is to make the model useful in the sense that the model represents the system being modeled and provides accurate information about the system. Before the CTMC and semi-Markov models proposed in the previous chapters models can be accepted and used to support decision making, they need to be validated.

A model is considered valid for a set of experimental conditions if its accuracy is within its acceptable range. This generally requires that the model's output variables of interest are identified. If the variables of interest are random variables, the properties and functions of the random variables such as means are usually primary interest and used in determining model validity. In my models, the steady state probability of occupying a particular state is the primary interest. In this chapter, I will compare the steady state probabilities of my models with a set of measured ITDB behaviors facing attacks and validate the models I proposed.

Note that we can not absolutely validate a model over the complete domain. Instead, tests and experiments are conducted until sufficient confidence is obtained that a model can be considered valid for its intended application.

9.1 CTMC Model Validation

I validate the models by comparing the steady state probability of occupying a particular state computed from the model to the estimated probability from the observed data. The probabilities from the measured data are estimated as the ratio of the length of time the system was in that state to the total length of the period of observation. The steady state probabilities for the CTMC models are computed using Equation 6.3 in Chapter 6:

$$\pi \mathbf{Q} = 0 \quad \sum_{i \in S} \pi_i = 1$$

where $\mathbf{Q}$ is the generator matrix and π is the steady state probability vector of the CTMC. Transition probabilities in the generator matrix $\mathbf{Q}$ are listed in Table 9.1:

Table 9.1. Transition Probabilities List

Parameters	Meaning
λ_a	Attack Hitting Rate
α	False Alarm Rate
d	Detection Probability
λ_{d_i}	Detection Rate
λ_{m_i}	Mark Rate
λ_{r_i}	Repair Rate
λ_{md}	Manual Detection Rate
λ_{mr}	Manual Repair Rate

For above parameters, some are completely determinate in my experiments. For example, the behaviors of attackers, human interaction and the properties of IDS can be controlled in the experiments. The value of λ_a, α, d, λ_d, λ_{mr} and λ_{md} can be set.

Some parameters in my models are the characters of ITDB, which can not be controlled in the experiments. We need to estimate the value of these parameters. Numerical methods for parameter estimation is useful, but the method of maximum likelihood is considered to be more robust and yields estimators with good statistical properties from a statistical point of view. The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data. Maximum likelihood estimation methods are versatile and apply to most models and to different types of data. In addition, they provide efficient methods for quantifying uncertainty through confidence bounds. The methodology for maximum likelihood estimation is quite simple. In this section, I will use the method of maximum likelihood to produce estimators of parameters we can not control.

Assume we observed k scan events and repair events, the total scan time is M_k, and the total repair time is R_k. The maximum likelihood estimators of λ_{m_i}, λ_{r_i} are

$$\widetilde{\Lambda}_{M_i} = \frac{k}{M_k}$$
$$\widetilde{\Lambda}_{R_i} = \frac{k}{R_k} \tag{9.1}$$

I test the CTMC models by two sets of experimental conditions. The comparison results between the steady state probabilities computed from the CTMC model and the

estimated probabilities from the observed data are shown in Table 9.2 and 9.3. The results are expressed as percentage.

Table 9.2. Comparison of CTMC State Occupancy Probabilities (Scenario #1)

State	Value from CTMC Model	Observed Value	Difference (%)
G	80.46	80.83	0.46
I	11.00	10.78	2.00
M	2.11	2.06	2.37
R	4.50	4.60	2.22
MD	1.10	1.03	6.36
MR	0.55	0.59	7.27

Table 9.3. Comparison of CTMC State Occupancy Probabilities (Scenario #2)

State	Value from CTMC Model	Observed Value	Difference (%)
G	63.60	64.48	1.38
I	16.00	15.42	3.62
M	4.09	3.92	4.16
R	8.73	8.29	5.04
MD	4.80	4.51	6.04
MR	2.40	2.49	3.75

Table 9.4 summarizes the values of parameters in the validation experiments.

It can be seen that the computed values from the CTMC models can match the actual observed values quite closely. The difference is within 9%. This validates the model building methodology and Markov chain. So the CTMC models can be taken to model the real-system reasonably well.

9.2 Semi-Markov Chain Model Validation

Using the same methodology as above, I validate the semi-Markov chain models by comparing the steady state probability of occupying a particular state computed from

Table 9.4. CTMC Model Validation Experimental Conditions

Parameters	Value
Attack Hitting Rate, λ_a	1 (Scenario #1); 2 (Scenario #2)
Detect Rate, λ_{d_i}	10 (Scenario #1); 15 (Scenario #2)
Mark Rate, λ_{m_i}	27
Repair Rate, λ_{r_i}	22
Manual Detection Rate, λ_{md}	10
Manual Repair Rate, λ_{mr}	20
False Alarm Rate, α	10% (Scenario #1); 20% (Scenario #2)
Detection Probability, d	90% (Scenario #1); 80% (Scenario #2)

the model to the estimated probability from the observed data. The probabilities from the measured data are estimated as the ratio of the length of time the system was in that state to the total length of the period of observation. The steady state probabilities for the semi-Markov models are computed using Equation 7.12 and 7.13 in Chapter 7:

$$\pi = \frac{1}{v\mathbf{h}} diag(v)\mathbf{h}$$

$$vP = v$$

$$\sum_{i \in E} v_i = 1$$

where π is steady-state probabilities of semi-Markov chain, $\mathbf{h}$ is the mean sojourn time in each state, P is the transition probability matrix of the Markov chain corresponding to the semi-Markov process, and v is its stationary distribution.

In addition to the parameters for the corresponding Markov chain discussed in Section 9.1, we still need to estimate or set the values of the mean sojourn time in each state. h_G, h_{I_i}, h_{MD}, and h_{MR} are completely determinate. I can control their values in my experiments. Parameters h_{M_i} and h_{R_i} are the character of ITDB system which can

not be controlled in the experiments. Maximum likelihood estimation method is applied to produce estimators of parameters h_{M_i} and h_{R_i}.

The semi-Markov chain model proposed in Chapter 7 is tested by two sets of experimental conditions. The comparison results between the steady state probabilities computed from the semi-Markov chain model and the estimated probabilities from the observed data are shown in Table 9.5 and 9.6. The results are expressed as percentage.

Table 9.5. Comparison of semi-Markov State Occupancy Probabilities (Scenario #1)

State	Value from semi-Markov Model	Observed Value	Difference (%)
G	80.64	80.83	0.24
I	10.89	10.78	1.01
M	2.09	2.06	1.44
R	4.57	4.60	0.66
MD	1.14	1.03	3.74
MR	0.56	0.59	5.36

Table 9.6. Comparison of semi-Markov State Occupancy Probabilities (Scenario #2)

State	Value from semi-Markov Model	Observed Value	Difference (%)
G	63.88	64.48	0.94
I	15.72	15.42	1.91
M	4.04	3.92	2.97
R	8.53	8.29	2.81
MD	4.68	4.51	3.63
MR	2.43	2.49	2.47

The results in Table 9.5 and 9.6 show that the computed values from the semi-Markov chain model can match the actual observed values quite closely. The difference is within 6%. This validates the semi-Markov chain model. In addition, we can find the difference between the observed values and computed values from the semi-Markov models is smaller than the difference between the observed values and computed values

from the Markov models. This indicates that the semi-Markov models fit better than the Markov models in modeling the survivability of an intrusion tolerant database system. Table 9.7 summarizes the values of parameters in the validation experiments.

Table 9.7. Semi-Markov Model Validation Experimental Conditions

Parameters	Value
Attack Hitting Rate, λ_a	1 (Scenario #1); 2 (Scenario #2)
Detect Rate, λ_{d_i}	10 (Scenario #1); 15 (Scenario #2)
Mark Rate, λ_{m_i}	27
Repair Rate, λ_{r_i}	22
Manual Detection Rate, λ_{md}	10
Manual Repair Rate, λ_{mr}	20
False Alarm Rate, α	10% (Scenario #1); 20% (Scenario #2)
Detection Probability, d	90% (Scenario #1); 80% (Scenario #2)
State G Sojourn Time, h_G	5
State I Sojourn Time, h_I	1(Scenario #1); 2(Scenario #2)
State M Sojourn Time, h_M	3
State R Sojourn Time, h_R	1
State MD Sojourn Time, h_{MD}	10
State MR Sojourn Time, h_{MR}	20

Chapter 10

Results

In this section, I use ITDB [8] as an example to study intrusion tolerant database systems' survivability metrics I proposed in Chapter 7. Instead of evaluating the performance of a specified system, I focus on the impact of different system deficiencies on the survivability in the face of attack. Experiments run using different system settings and workloads. The analysis presented here is designed to compare the impact of different parameters of intrusion detection subsystems, such as False Alarm Rate, Detection Latency; and different workload parameters, such as Attack Rates on the relative survivability metrics of ITDB.

10.1 Impacts of Attack Intensity

The attack intensity can challenge the survivability of a security system, especially an intrusion tolerant system. Heavy attack intensity keeps the security system busy on defending and repairing the damage. The system may be crashed before the security system recovers the damage. As an intrusion tolerant system, it allows intrusions, but a key problem is whether it can tolerate different attack intensity. To answer this question, in this part, I will study the impacts of attack rate on survivability of ITDB.

I compare the steady state probabilities of different system configuration of ITDB under different attack rates. In Figure 10.1(a), an example of a good intrusion tolerant

system is shown. Here, a good system is one which has a high detection probability, low false alarm rate, short detection latency, fast damage assessment and repair system. In another word, the transition probability d, λ_{m_i}, and λ_{r_i} are large. The waiting time h_{m_i}, h_{r_i} are small.

As can be seen, the heavy attacks have little impact on the survivability of ITDB. The damage assessment and repair subsystems can locate and mask the intrusion quickly. As a result, the steady state probabilities of state I, R, and M are very low. The integrity and rewarding-availability remain at a high level (> 0.8). The only impact of high attack rate is that the probability of ITDB staying at state I is increased. This does not hurt the survivability of ITDB.

An example of a bad system is shown in Figure 10.1(b). In contrast to a good intrusion tolerant system, I define a bad system as one which has a low detection probability, high false alarm rate, long detection latency, slow damage assessment and repair system. In other words, the transition probabilities d, λ_{m_i}, and λ_{r_i} are small. The waiting time h_{m_i}, h_{r_i} are large.

The high attack rate increases the work load for damage marking and repairing subsystems. As a result, steady state probabilities of state R ($\pi_R > 0.3$) and state M go up quickly. This keeps ITDB busy on analyzing and masking the heavy attacks. However, the system integrity is not impacted by the attacks significantly. The reason is that the ITDB applies the damage quarantine strategy. This enables the ITDB having the capability to provide clean information to users even facing heavy attacks.

10.2 Impact of False Alarms

False alarm is a key factor to evaluate the performance of an IDS. ITDB adopts the behavior-based intrusion detection techniques. The high false alarm rate is often cited as the main drawback of behavior-based detection techniques since the entire scope of the behavior of an information system may not be covered during the learning phase. A high false alarm rate may bring extra workload to the recovery subsystem and waste some system resources. Will ITDB tolerate the relatively high false alarm rates? To answer this question, I will evaluate the impact of false alarms on the steady state of ITDB in this part.

Figure 10.2(a) shows the variation of steady state probabilities when ITDB is under light attacks ($\lambda_a = 1$). ITDB maintains the integrity (> 0.85) and rewarding-availability (> 0.6) at a high level, even though facing a nearly 100% false alarm rate. This indicates that the system can tolerate a high false alarm rate under light attacks. Also the steady state probabilities of state I, M, R are at a very low level (< 0.1). This indicates that the system can quarantine, locate, and repair the attacks efficiently and quickly.

Another case that ITDB is under heavy attacks ($\lambda_a = 5$) is shown in Figure 10.2(b). As can be seen, high false alarm brings pressure on ITDB. The steady state probability of state I, π_D, is higher than the probability state G, π_G, when false alarm rate is higher than 60%. The heavy attacks and extra load brought by false alarms increase the steady state probabilities of state I, M, and R. These mean that ITDB spends much more time on state I and keeps busy analyzing and repairing the damage.

The rewarding-availability decreases as the damage quarantine and assessment process becomes longer. At the same time, the system still can maintain the integrity (> 0.85) at a high level. This indicates that the probability that the system can provide clean data to some users is high.

10.3 Impact of Detection Probability

Detection probability is another important metric used to measure the performance of an intrusion detector. In this section, I will study the impact of detection probabilities on the survivability under different detection probabilities.

Figure 10.3(a) shows that ITDB is under light attack ($\lambda_a = 1$). When the detection rate is 0%, the system totally depends on manual detection. Since the manual detection requires human intervention, it takes a relatively long time to detect the intrusion manually. As a result, ITDB has a high probability (> 0.4) of staying at state MD and a low probability of staying at state G when $d = 0$. The integrity and rewarding-availability are also at a low level (≈ 0.5).

The steady state probability of state MD goes back to 0 when the detection probability is 100%. The steady state probabilities of state M and R go up while the detection probability is increasing. This indicates that, with more attacks identified by the IDS, the system will spend more time on damage assessment and recovery. Since the manual repair is much slower than the repair subsystem of ITDB, the rewarding-availability and integrity go up while the detection probability is increasing.

When ITDB faces a heavy attack as shown in Figure 10.3(b), low detection rate hurts the performance of ITDB significantly. The steady state probability of state G,

π_G is lower than 0.5 when the detection probability is 0%. As a result, the integrity and rewarding-availability is also very low when the intrusion detection can not find the intrusion. However, as can be seen in Figure 10.3(b), the ITDB still can maintain the system. With the detection rate is increasing, the integrity and rewarding-availability go up smoothly. This indicates that the ITDB can tolerate the low detection probability when facing a heavy attack.

Compared with the false alarms, the impact of detection probability on the survivability of an intrusion tolerant database system is more severe. The variance of integrity and rewarding-availability is less than 0.2 when the detection probability changes from 0% to 100%, while the variance is nearly 0.4 when changing false alarm rate from 0% to 100%. One reason is that the high false alarms will bring extra load to the security system to quarantine and repair unaffected data items, while low detection probability will bring more work for the administrator to mark and repair the damage manually. If the system can identify and recover the damage faster than manual operation, the impact of low detection probability is more severe and more dangerous to the survivability. This result encourages us to consider more on improving the detection probability for the future intrusion tolerant system development.

10.4 Transient Behaviors

Much of the theory developed for solving Markov chain models is devoted to obtaining steady state measures, that is, measures for which the observation interval is "sufficiently large" ($t \rightarrow \infty$). These measures are indeed approximations of the behavior of the system for a infinite, but long, time interval, where long means with respect to the

interval of time between occurrences of events in the system. However, in some cases the steady state measures are not good approximations for the measures during a relatively "short" period of time.

Before reaching the steady state, the system will go through a transient period. If the damage quarantine and recovery systems are not efficient enough, the system may never reach steady states, or take a very long time. The cumulative time distribution of quarantine and repair states will be dominant. Even though the steady state probability of good state is high, obviously we can not satisfy the system's performance. The limitation of steady state measures motivates me to observe the transient behaviors of different intrusion tolerant systems in this part. Because of the computation complexity of semi-Markov chain on transition behaviors, I apply CTMC model to study the transition behavior of ITDB instead of semi-Markov chain model. Figure 10.4 and 10.5 show the comparison results. I start the system from state G, which means $P_G(0) = 1$.

A better system's behaviors are shown in Figure 10.4. I assume that a better intrusion tolerant system has a good IDS, which can detect intrusion quickly and have a high detection rate and a low false alarm rate. Damage assessment and repair systems can locate and mask the intrusion quickly. As can be seen in Figure 10.4(a), a better system reaches steady state quickly. The probability of staying at state G is high, while the probabilities of staying at another states, like state I, R, and M, are very low. From Figure 10.4(b), we can also find that the cumulative time distribution of staying at state G is dominant, which means the system will spend most of time at good state. Since the damage assessment and repair system can accomplish their tasks quickly, the cumulative time distribution of state I, R, and M are low.

In Figure 10.5, I give an example of a poor system, which has a slow assessment and repair system. Compared with Figure 10.4, we can find that it takes a longer time for the system to reach steady states. The cumulative time of state G is not dominant. The system spends more time on damage assessment and repair.

10.5 Overhead on System Performance

To achieve good survivability, some functionalities introduced by ITDB may bring performance degradation to the database system. In this section, I will study how significantly ITDB affects the system performance and which component of ITDB is the major cause of performance degradation. The deficiencies of intrusion detection may also be a factor affecting system throughput during the damage quarantine and recovery processes. The impact of intrusion detection deficiencies on system performance will also be studied in this section. Based on these results, we can learn how to optimize ITDB so that the system performance can be further improved.

I apply the TPC-C benchmark to evaluate the run-time performance of ITDB on the top of Oracle database server. The performance metric used by TPC-C is *transactions per minute* (tpmC). Instead, I use response time (millisecond per transaction) to measure the performance degradation. Because the response time for each transaction is very short, I measure ITDB's performance through average response time to increase the accuracy. The Transaction Simulator is used to compose and submit transactions to the database server in batch-style.

The response time is composed of several parts:

- Time used by the Transaction Simulator to prepare each SQL statement in transactions and submit them;
- Time used by the Transaction Proxy Subsystem to log each user transaction and forward it to the DBMS;
- Time used by the DBMS to process each statement;
- Time used by the damage quarantine and repair process when intrusions happen.

10.5.1 Transaction Proxy Overhead

The transaction proxy subsystem is an important part of ITDB. It provides services to clients for accessing database, and captures some information about submitted transactions' behaviors. In this part, I will measure the system performance decrease caused by transaction proxy subsystem.

Two important functions of the transaction proxy subsystem are transferring transactions to database and keeping operation logs of transaction. It is necessary to measure the performance impacts of these two parts separately. Then we can analyze which is the main reason to decrease the performance and then we focus our research to improve the system performance.

First, I will submit transactions without proxy, meaning I submit transactions directly to the database. Second, I will submit transactions through proxy, but the proxy will not generate any log. Finally, I will submit transactions through proxy, and it also generates logs.

The results in Figure 10.7(c), 10.6(b), and 10.6(c) show that system performance is decreased when I send transactions to database system through transaction proxy subsystem, although the decrease is not significant. The proxy part decreases the performance by 11%. The logs part decreases the system performance by 31%. These show that the logging part is the main component that impacts the system performance in the ITDB's transaction proxy subsystem, although it is not very significant.

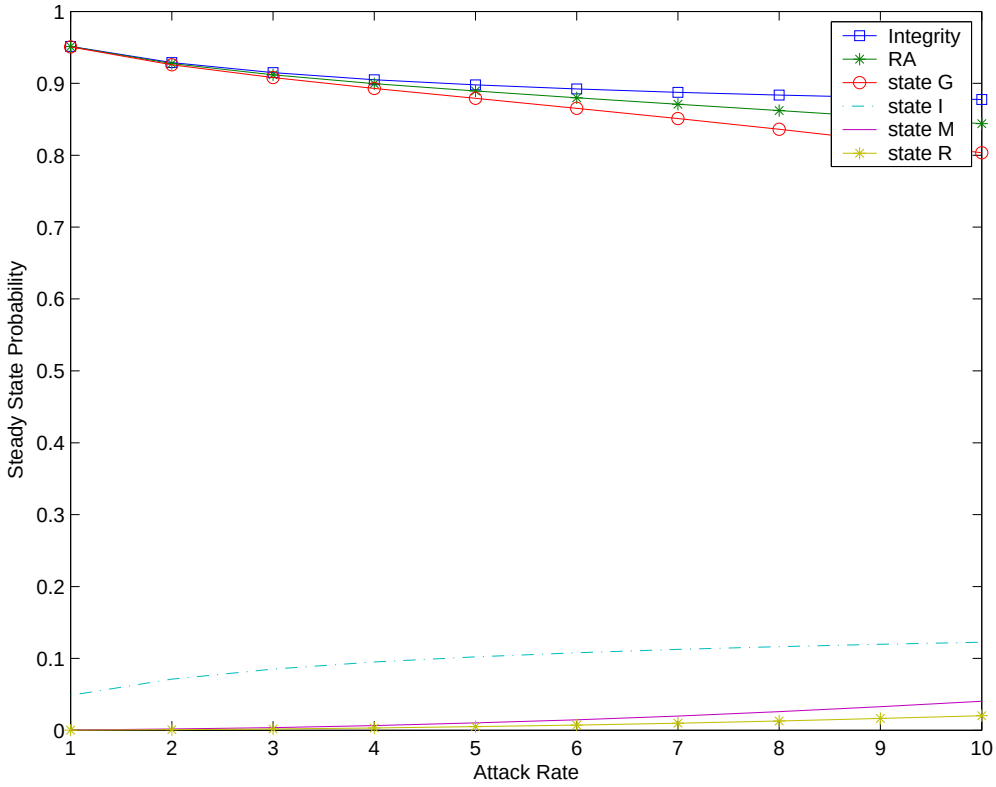
ITBD keeps several logs on submitted transactions, such as read operation logs, write operation logs, and transaction logs. These logs will be used for intrusion detection and attack recovery. The efficiency of keeping logs is important to the overall ITDB system. The experiment results suggest that further research should focus on developing a more efficient log system.

10.5.2 Damage Quarantine and Repair Overhead

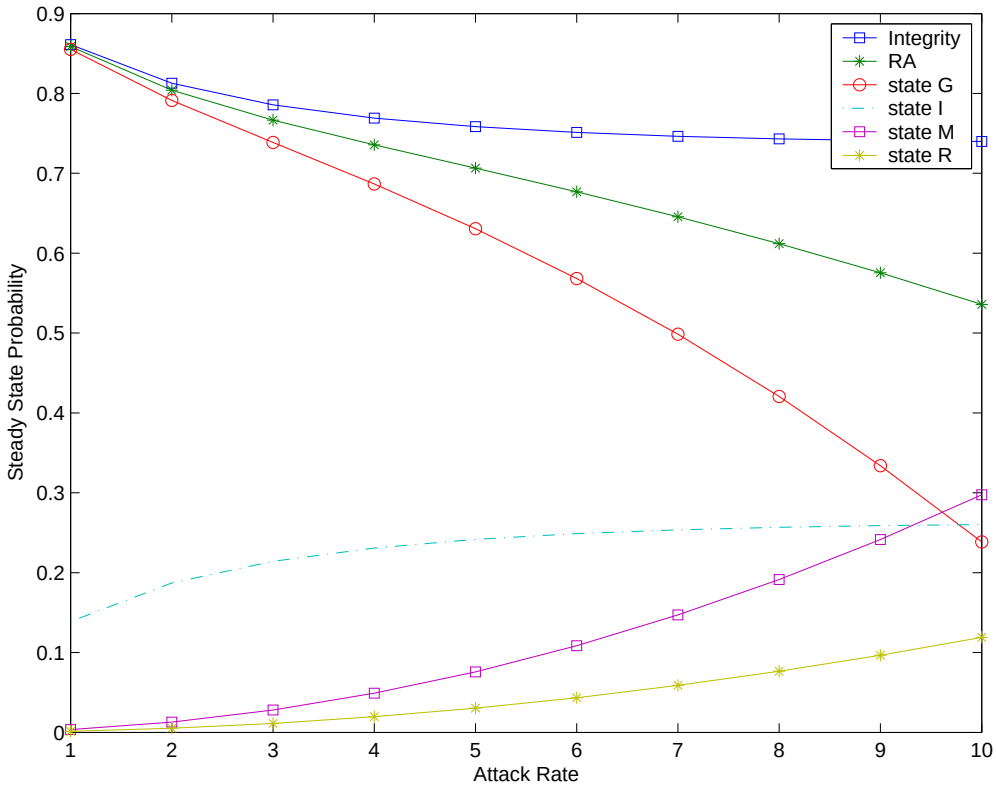
The damage quarantine and the repair subsystems are triggered when the IDS raises intrusion alarms. After all the alarms have been processed and their damage has been repaired, these two subsystems will be deactivated. In this section, I am interested in the system throughput degradation caused by these two subsystems.

Figure 10.7 shows the system's response time during the process of damage quarantine and repair under different detection latency. As can be seen, the overhead of this process is significant. However, by enforcing the damage quarantine algorithm, the attack recovery subsystem can repair the damage at a faster speed. The duration of this

process, namely the quarantine time window, is very short. So the average overhead caused by the process is not very significant.

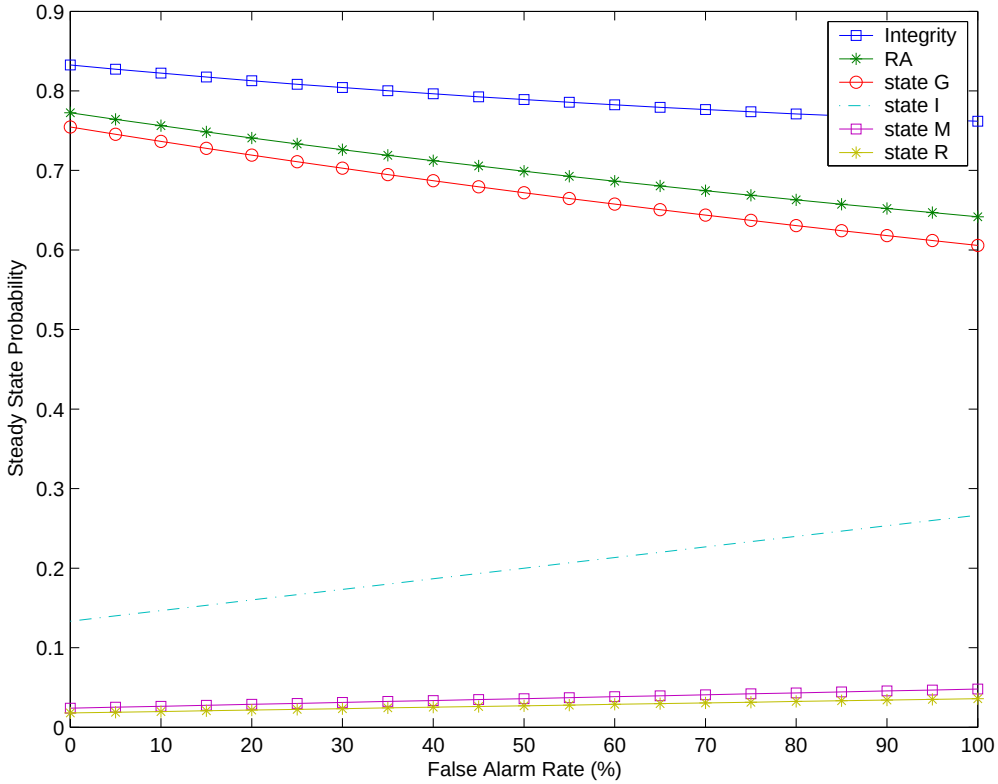


(a) good system ($d = 99\%$, $\alpha = 0.1$, $\lambda_d = 20$, $h_m = 3$, $h_r = 1$)

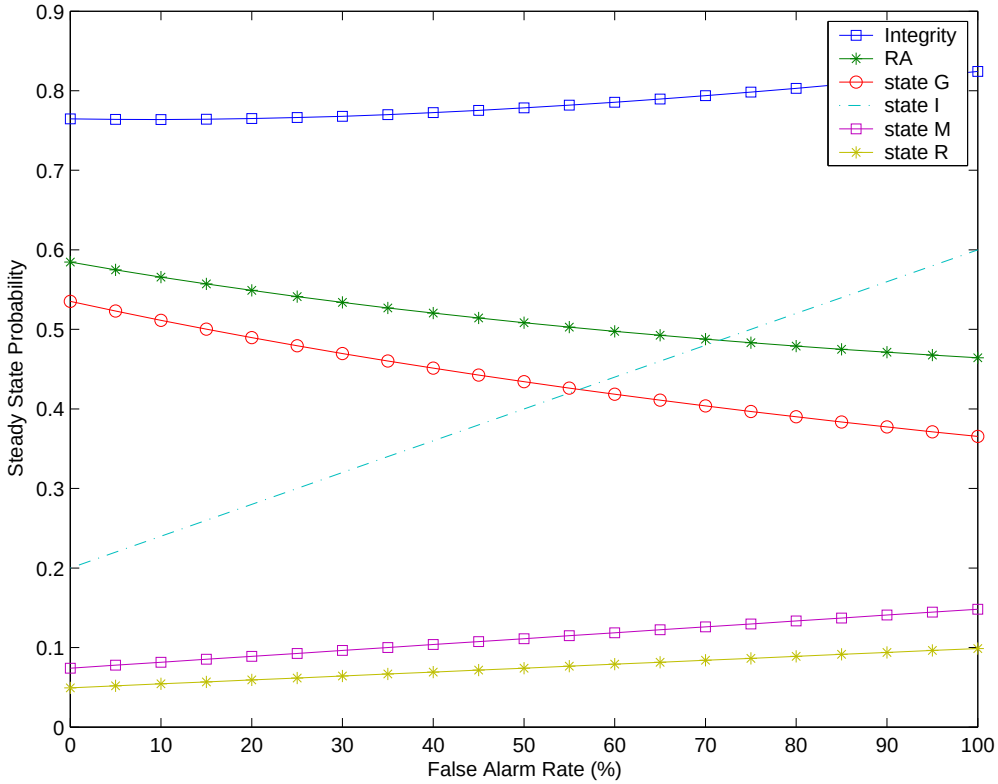


(b) poor system ($d = 80\%$, $\alpha = 0.5$, $\lambda_d = 10$, $h_m = 5$, $h_r = 3$)

Fig. 10.1. Impact of Attack Intensity

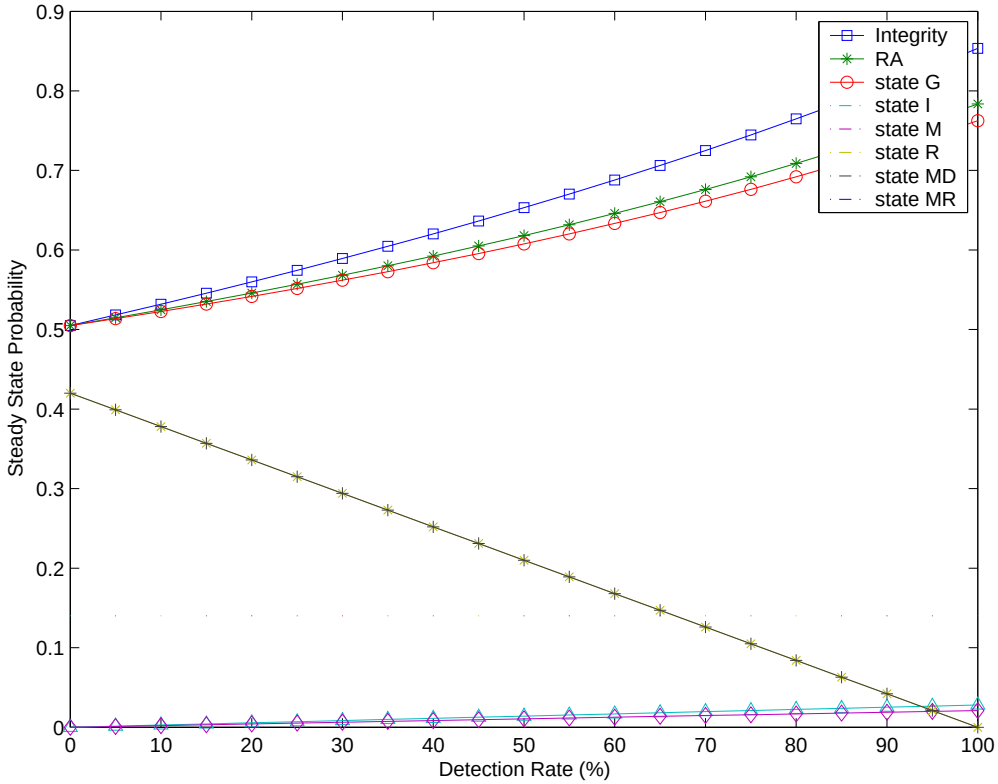


(a) Light Attack ($\lambda_a = 1$, $\lambda_d = 15$, $d = 90\%$, $h_m = 3$, $h_r = 1$)

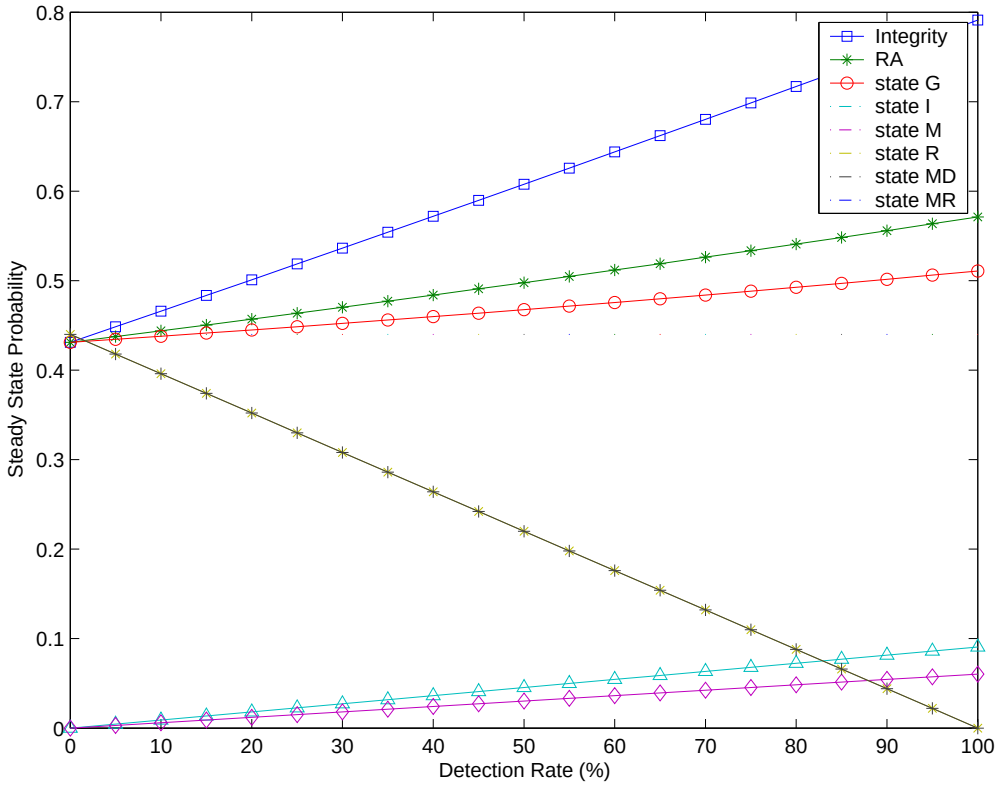


(b) Heavy Attack ($\lambda_a = 5$, $\lambda_d = 15$, $d = 90\%$, $h_m = 3$, $h_r = 1$)

Fig. 10.2. Impact of False Alarm Rate

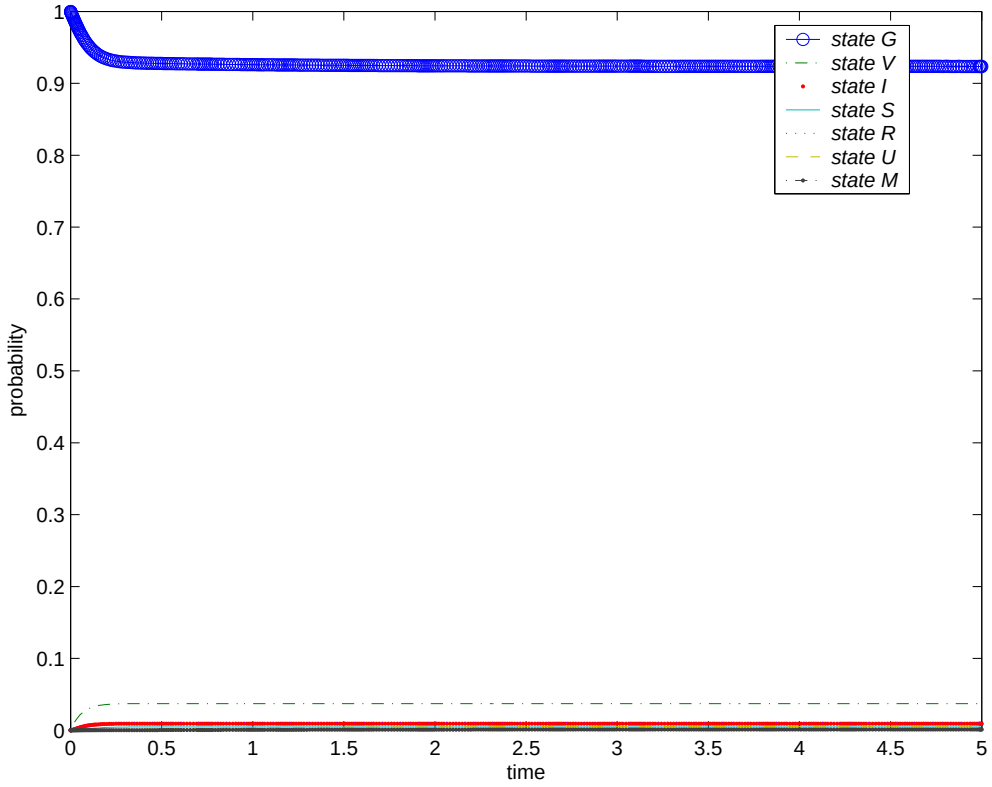


(a) Light Attack ($\lambda_a = 1$, $\alpha = 0.2$, $\lambda_d = 15$, $h_m = 3$, $h_r = 1$)

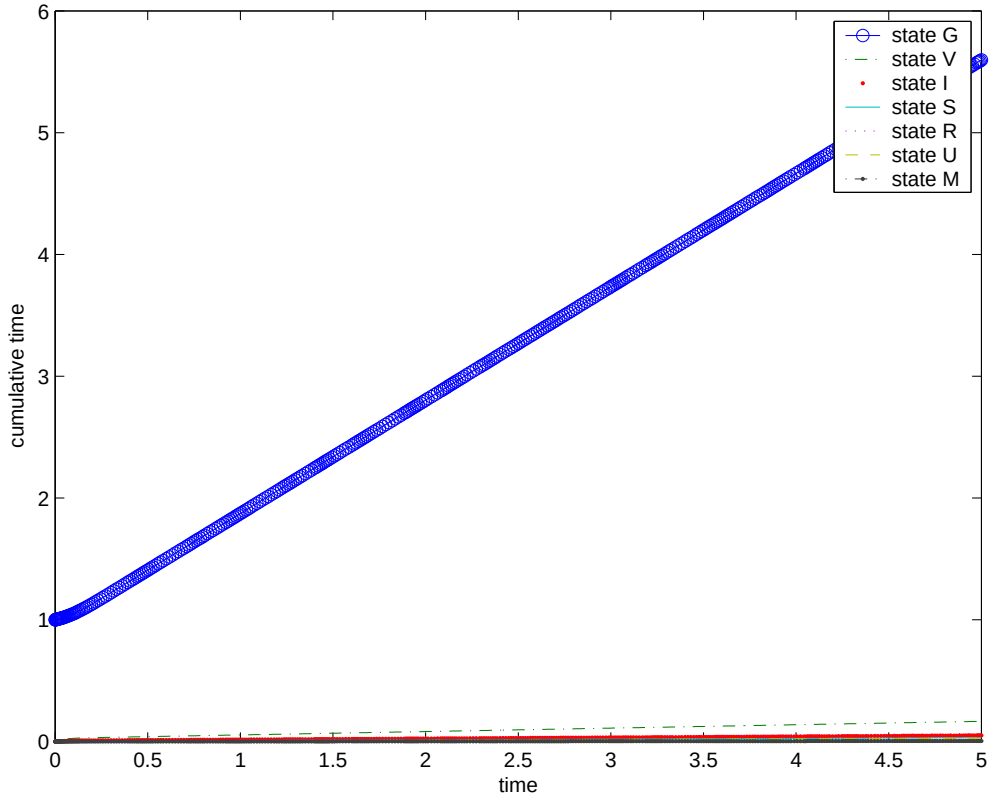


(b) Heavy Attack ($\lambda_a = 5$, $\alpha = 0.2$, $\lambda_d = 15$, $h_m = 3$, $h_r = 1$)

Fig. 10.3. Impact of Detection Rate

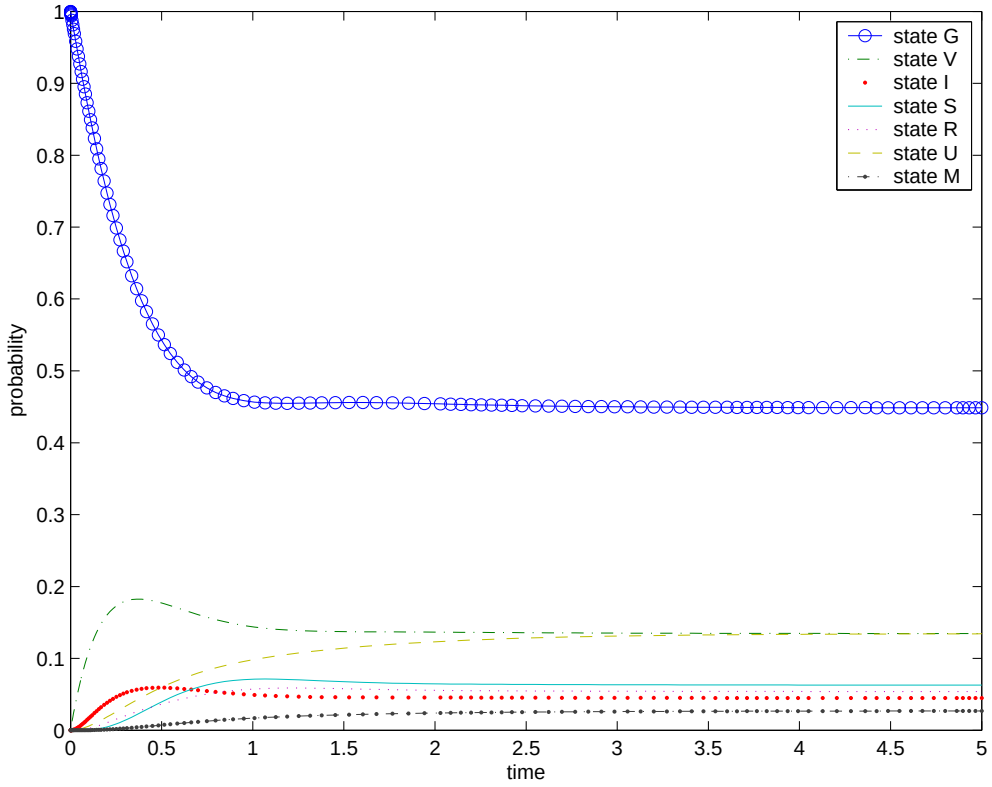


(a) transient probabilities of a good system

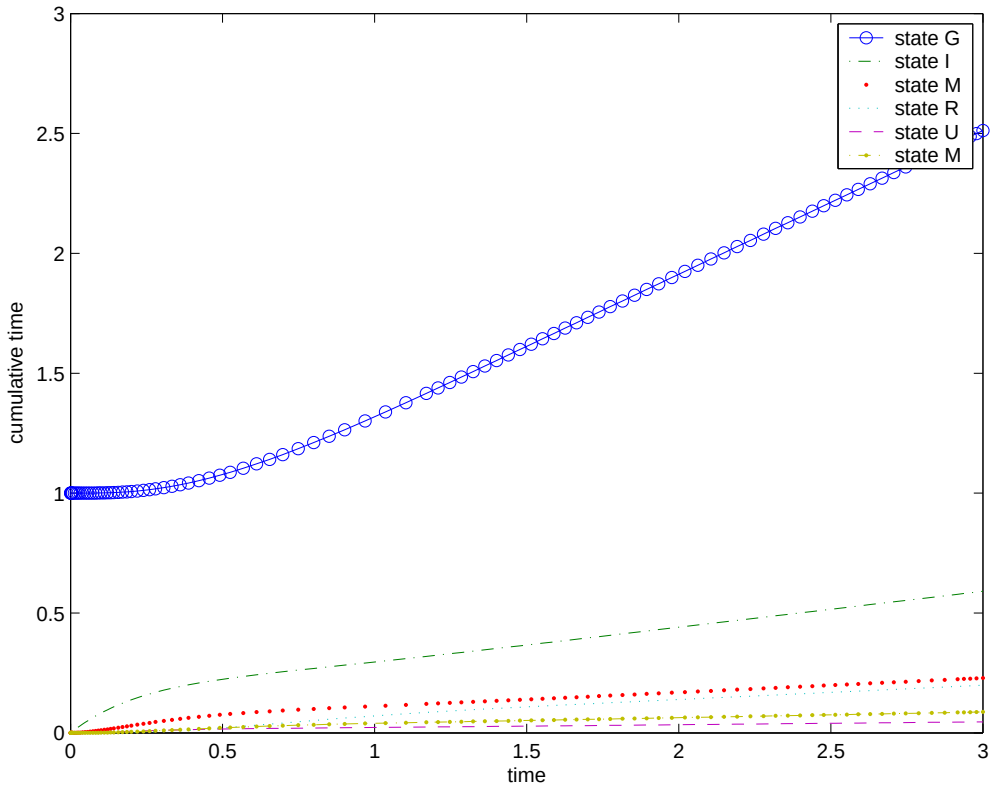


(b) cumulative time distribution of a good system

Fig. 10.4. Transient Behaviors of a good system

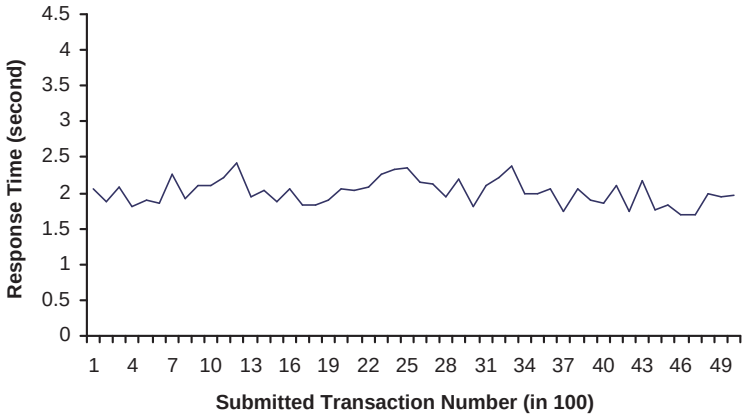


(a) transient probabilities of a poor system

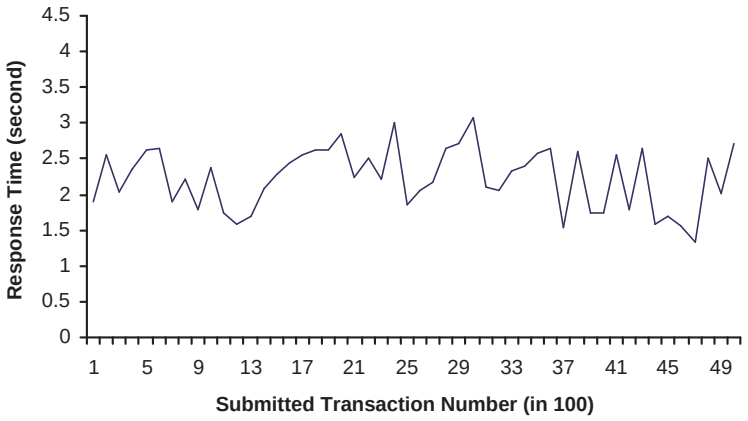


(b) cumulative time distribution of a poor system

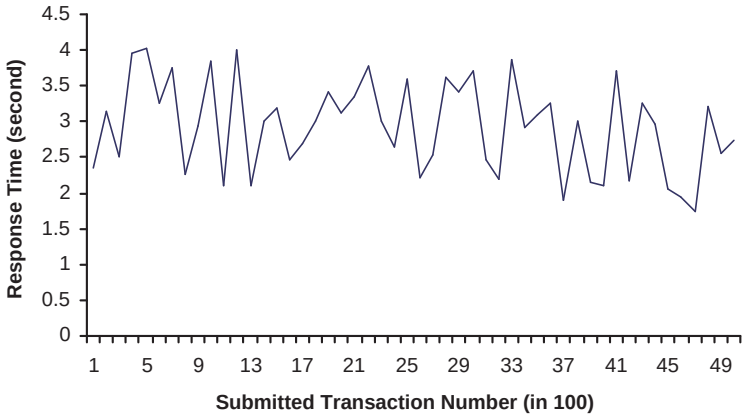
Fig. 10.5. Transient Behaviors of a poor system



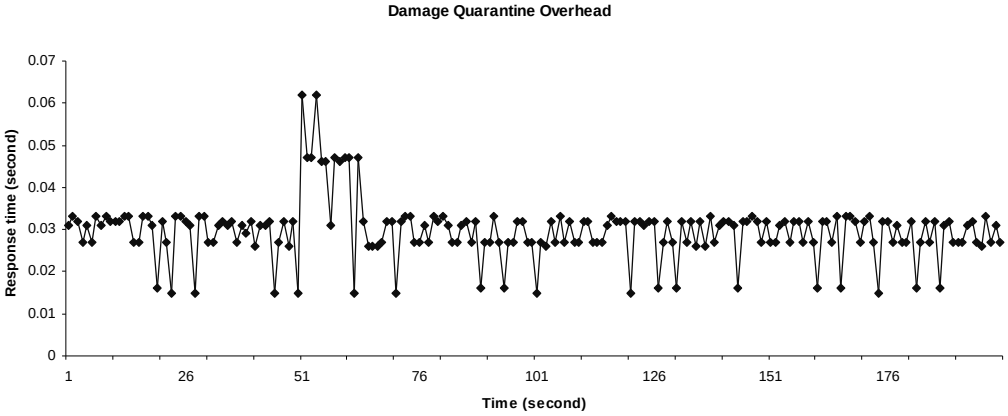
(a) Response Time per 100 Transactions without transaction proxy subsystem. The average response time per100 transactions (without proxy) is 2.011 second. The maximum response time of 100 transactions (without proxy) is 2.422 second.



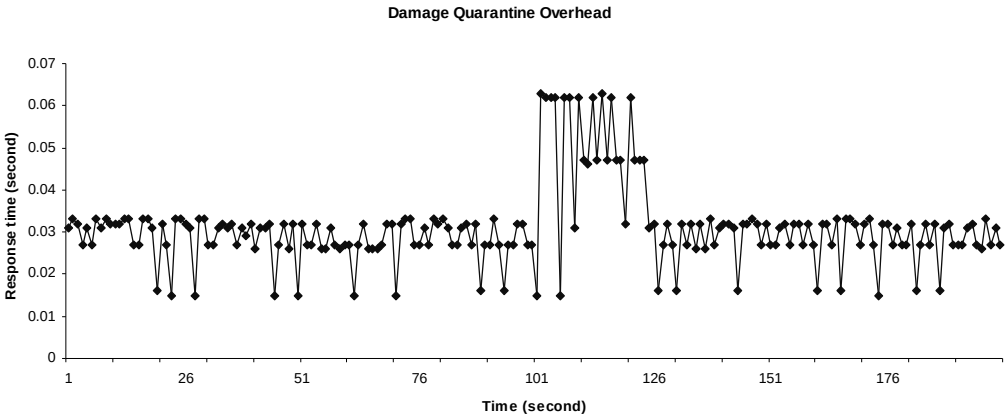
(b) Response Time per 100 Transactions with transaction proxy. The average response time per100 transactions (without proxy) is 2.231 second. The maximum response time of 100 transactions (without proxy) is 3.079 second.



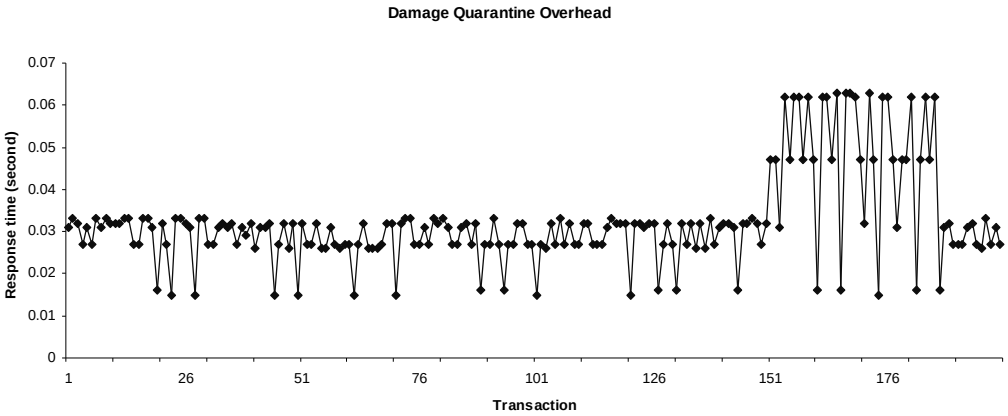
(c) Response Time per 100 Transactions with transaction proxy and logs. The average response time per100 transactions (without proxy) is 2.926 second. The maximum response time of 100 transactions (without proxy) is 4.015 second.



(a) DL = 50 second



(b) DL = 100 second



(c) DL = 150 second

Fig. 10.7. System Response Time

Chapter 11

Conclusion and Future Works

11.1 Conclusion

In this thesis, I focus on modeling the survivability of an intrusion tolerant database system in the presence of attacks.

Before we study the intrusion tolerant system, we need to have a better understanding of the attack behavior and its degree of spreading in a database system. Based on the classical epidemic SIS model, a stochastic (Markovian) damage propagation model in continuous time is proposed. This model leads to a better understanding and prediction of the scale and speed of database damage propagation. The impact of different parameters on damage spreading is studied by simulation experiments.

I extended the classic reliability/availability model to a new survivability model. Comprehensive state space approaches are applied to study the complex relationships and their transition structure encoding sequencing response of intrusion tolerant database systems facing attacks. A basic intrusion tolerant database system model and a comprehensive state transition model of ITDB are presented in this thesis. Mean Time to Attack (MTTA), Mean Time to Detection (MTTD), Mean Time to Marking (MTTM), and Mean Time to Repair (MTTR) are defined as basic measures of survivability. Quantitative metrics integrity and rewarding-availability are defined to evaluate the survivability of intrusion tolerant database systems.

The Markov chain models assume that the distributions for transitions between states are exponential. However, in some real cases the attack and repair time are not exponentially distributed. Semi-Markov process is an extension of Markov chain, which allows the distributions for transitions between states can be nonexponential functions. I extend the model to a semi-Markov process which fits better than the Markov one in real problems.

A real intrusion tolerant database system, ITDB, is implemented to conduct empirical experiments to validate the proposed state-space models. Experiment results show the validity of the proposed survivability models.

To further validate the security of ITDB, I performed an empirical survivability evaluation, where maximum-likelihood methods are applied to estimate the values of the parameters used in my state transition models. The impacts of existing system deficiencies and attack behaviors on the survivability are studied using defined quantitative measures. The efficiency of ITDB is also evaluated using TPC-C benchmark.

My evaluation results indicate that (1) ITDB can provide essential database services in the presence of attacks, and (2) maintain the desired essential survivability properties without being seriously affected by various system deficiencies and different attack intensity. (3) Performance measurements show that ITDB system is cost-effective system.

11.2 Future Works

11.2.1 Extending Damage Propagation Models

Based on the classical epidemic model, a stochastic (Markovian) damage propagation model in continuous time is proposed. The model focuses on the database with a homogeneous population. Compared with complicated database structures, the damage propagation model still need to be improved.

In the future, I would like to extend my current damage propagation model. The genic epidemic model can be borrowed to simulate the relational structures in databases. I also need to establish models to simulate the complicated transaction structures.

11.2.2 Modeling complicated attack behaviors

To evaluate the survivability of an intrusion tolerant database system, it is very important to model the attack behaviors. There are some other distribution functions which can describe the attack behaviors. The hypo-exponential distribution is a phase-type distribution. It consists of a number exponential "phases" or "stages" in series and parallel, respectively. It may be used to model transition that may involve multi-stage activities. Weibull distribution is flexible to mimic the behavior of other statistical distributions such as the normal and the exponential. In the future, I would like to study different attack behaviors by applying the semi-Markov models.

11.2.3 Studying Transition Behaviors

I extended the CTMC models to semi-Markov process models to simulate the nonexponential distributions. However, because of the calculation complexity of semi-Markov chain on transition behaviors, I did not apply the semi-Markov chain models to study the transition behaviors of ITBD. In the future, I would like to apply the established semi-Markov models to study the complicated transition behaviors.

11.2.4 Integrating Models

In Chapter 4, a stochastic damage propagation model in continuous time is proposed. The model help us to have a better understanding and prediction of the scale and speed of database damage propagation. However, I have not integrated this damage propagation model with the CTMC models proposed in Chapter 5 and the semi-Markov chain models proposed in Chapter 7. In the future, I would like integrate these models together. In this way, we will have more clear understanding of the impacts of damage propagation on the survivability of an intrusion tolerant database system and the impacts of intrusion tolerant database system on controlling the damage spreading in a database.

Bibliography

[1] H. Wang and P. Liu, "Modeling and evaluating the survivability of an intrusion tolerant database system," in *ESORICS*, 2006.

[2] P. Liu, H. Wang, and L. Li, "Real-time data attack isolation for commercial database applications," *J. Netw. Comput. Appl.*, vol. 29, no. 4, pp. 294–320, 2006.

[3] H. Wang, P. Liu, and L. Li, "Evaluating the impact of intrusion detection deficiencies on the cost-effectiveness of attack recovery." in *Information Security, 7th International Conference, ISC 2004, Palo Alto, CA, USA, September 27-29*, 2004, pp. 146–157.

[4] E. B. Fernandez, R. C. Summers, and C. Wood, *Database Security and Integrity.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1981.

[5] P. P. Griffiths and B. W. Wade, "An authorization mechanism for a relational database system," *ACM Trans. Database Syst.*, vol. 1, no. 3, pp. 242–255, 1976.

[6] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data.* New York, NY, USA: ACM Press, 2004, pp. 563–574.

[7] O. T. T. Project, "Top ten the most critical web application security flaws," http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, 2004.

[8] P. Liu, "Architectures for intrusion tolerant database systems." in *18th Annual Computer Security Applications Conference (ACSAC 2002), 9-13 December 2002, Las Vegas, NV, USA*, 2002, pp. 311–320.

[9] A. Smirnov and T. cker Chiueh, "A portable implementation framework for intrusion-resilient database management systems." in *2004 International Conference on Dependable Systems and Networks (DSN 2004), 28 June - 1 July 2004, Florence, Italy, Proceedings*, 2004, pp. 443–452.

[10] K. S. Trivedi, *Probability and statistics with reliability, queuing and computer science applications.* Chichester, UK, UK: John Wiley and Sons Ltd., 2002.

[11] R. A. Sahner, K. S. Trivedi, and A. Puliafito, *Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package.* Norwell, MA, USA: Kluwer Academic Publishers, 1996.

[12] D. M. Nicol, W. H. Sanders, and K. S. Trivedi, "Model-based evaluation: From dependability to security," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 48–65, 2004.

[13] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff, and N. R. Mead, "Survivability: Protecting your critical systems." *IEEE Internet Computing*, vol. 3, no. 6, pp. 55–63, 1999.

[14] D. Barbará, R. Goel, and S. Jajodia, "Mining malicious corruption of data with hidden markov models." in *Research Directions in Data and Applications Security,*

IFIP WG 11.3 Sixteenth International Conference on Data and Applications Security, July 28-31, 2002, Kings College, Cambridge, UK, 2002, pp. 175–189.

[15] C. Y. Chung, M. Gertz, and K. N. Levitt, "Demids: A misuse detection system for database systems." in *Integrity and Internal Control in Information Systems, IFIP TC11 Working Group 11.5, Third Working Conference on Integrity and Internal Control in Information Systems: Strategic Views on the Need for Control, Amsterdam, The Netherlands, November 18-19, 1999*, 1999, pp. 159–178.

[16] V. C. S. Lee, J. A. Stankovic, and S. H. Son, "Intrusion detection in real-time database systems via time signatures." in *IEEE Real Time Technology and Applications Symposium*, 2000, pp. 124–133.

[17] Y. Hu and B. Panda, "A data mining approach for database intrusion detection." in *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, March 14-17, 2004*, 2004, pp. 711–716.

[18] R. Graubart, L. Schlipper, and C. McCollum, "Defending database management systems against information warfare attacks," The MITRE Corporation, Tech. Rep., 1996.

[19] J. Knight, K. Sullivan, M. Elder, and C. Wang, "Survivability architectures: issues and approaches," in *DARPA Information Survivability Conference and Exposition*, 2000, pp. 157 – 171.

[20] V. Stavridou, B. Dutertre, R. Riemenschneider, and H. Saidi, "Intrusion tolerant software architectures," in *DARPA Information Survivability Conference and Exposition II*, 2001, pp. 230 – 241.

[21] D. Medhi and D. Tipper, "Multi-layered network survivability-models, analysis, architecture, framework and implementation: an overview," in *DARPA Information Survivability Conference and Exposition*, 2000, pp. 173 – 186.

[22] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, and P. K. Khosla, "Survivable information storage systems," *Computer*, vol. 33, no. 8, pp. 61–68, 2000.

[23] A. Goel, W. chang Feng, D. Maier, W. chi Feng, and J. Walpole, "Forensix: A robust, high-performance reconstruction system," in *ICDCSW '05: Proceedings of the Second International Workshop on Security in Distributed Computing Systems (SDCS) (ICDCSW'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 155–162.

[24] S. T. King and P. M. Chen, "Backtracking intrusions," *ACM Trans. Comput. Syst.*, vol. 23, no. 1, pp. 51–76, 2005.

[25] P. Ammann, S. Jajodia, and P. Liu, "Recovery from malicious transactions." *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1167–1185, 2002.

[26] P. Luenam and P. Liu, "Odar: An on-the-fly damage assessment and repair system for commercial database applications." in *DBSec*, 2001, pp. 239–252.

[27] A. Smirnov and T. cker Chiueh, "A portable implementation framework for intrusion-resilient database management systems." in *DSN*, 2004, pp. 443–452.

[28] TCSE, *Trusted Computer System Evaluation Criteria (Orange Book), DOD-5200.28-STD.* US Department of Defense, December 1985.

[29] CC, "Common criteria (cc), iso/iec 15408," www.commoncriteriaportal.org, 1998.

[30] J. C. Knight, E. A. Strunk, and K. J. Sullivan, "Towards a rigorous definition of information system survivability," vol. 1, no. 1, 2002, pp. 78–89.

[31] B. B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, and K. S. Trivedi, "A method for modeling and quantifying the security attributes of intrusion tolerant systems." *Perform. Eval.*, vol. 56, no. 1-4, pp. 167–186, 2004.

[32] C. E. Landwehr, "Formal models for computer security." *ACM Comput. Surv.*, vol. 13, no. 3, pp. 247–278, 1981.

[33] S. Singh, M. Cukier, and W. H. Sanders, "Probabilistic validation of an intrusion-tolerant replication system." in *DSN*, 2003, pp. 615–624.

[34] J. Zhang and P. Liu, "Delivering services with integrity guarantees in survivable database systems." in *Data and Applications Security XVII: Status and Prospects, IFIP TC-11 WG 11.3 Seventeenth Annual Working Conference on Data and Application Security, August4-6, 2003, Estes Park, Colorado, USA*, 2003, pp. 33–46.

[35] M. Yu, P. Liu, and W. Zang, "Self-healing workflow systems under attacks." in *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*, 2004, pp. 418–4025.

[36] P. Liu, J. Jing, P. Luenam, Y. Wang, L. Li, and S. Ingsriswang, "The design and implementation of a self-healing database system." *Journal of Intelligent Information Systems (JIIS)*, vol. 23, no. 3, pp. 247–269, 2004.

[37] J. O. Kephart and S. R. White, "Measuring and modeling computer virus prevalence," in *Proceedings of the 1993 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 1993, p. 2.

[38] J. C. Wierman and D. J. Marchette, "Modeling computer virus prevalence with a susceptible-infected-susceptible model with reintroduction." *Computational Statistics & Data Analysis*, vol. 45, no. 1, pp. 3–23, 2004.

[39] C. C. Zou, W. Gong, and D. F. Towsley, "Code red worm propagation modeling and analysis." in *ACM Conference on Computer and Communications Security*, 2002, pp. 138–147.

[40] N. T. Bailey and L. Bailey, *The Mathematical Theory of Infectious Diseases and its Applications*. Charles Griffin, 1975.

[41] H. Andersson and T. Britton, *Stochastic Epidemic Models and Their Statistical Analysis*. Springer, 2000.

[42] C. C. Zou, L. Gao, W. Gong, and D. F. Towsley, "Monitoring and early warning for internet worms." in *ACM Conference on Computer and Communications Security*, 2003, pp. 190–199.

[43] Z. Chen, L. Gao, and K. A. Kwiat, "Modeling the spread of active worms." in *INFOCOM*, 2003.

[44] J. C. Frauenthal, *Mathematical Modeling in Epidemiology.* Springer-Verlag, 1980.

[45] M. Bishop, *Computer Security: Art and Science.* Addison-Wesley Professional, 2002.

[46] E. Bertino and R. Sandhu, "Database security-concepts, approaches, and challenges," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 1, pp. 2–19, 2005.

[47] D. L. Carter and A. J. Katz, "Computer crime: An emerging challenge for law enforcement," *FBI Law Enforcement Bulletin*, December 1996.

[48] M. R. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore, "Insider threat study: Illicit cyber activity in the banking and finance sector," Tech. Rep., August 2004.

[49] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proceedings of the DARPA Information Survivability Conference and Exposition.* IEEE Computer Society Press", year = 2000,.

[50] W. Chen, S. Toueg, and M. K. Aguilera, "On the quality of service of failure detectors." *IEEE Trans. Computers*, vol. 51, no. 1, pp. 13–32, 2002.

[51] R. M. Smith, K. S. Trivedi, and A. V. Ramesh, "Performability analysis: Measures, an algorithm, and a case study," *IEEE Transactions on Computers*, vol. 37, no. 4, pp. 406–417, 1988.

[52] G. Ciardo, R. A. Marie, B. Sericola, and K. S. Trivedi, "Performability analysis using semi-markov reward processes," *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1251–1264, 1990.

[53] N. Limnios and G. Oprisan, *Semi-Markov Processes and Reliability.* Birkhauser;, 2001.

[54] A. Blasi, J. Janssen, and R. Manca, "Numerical treatment of homogeneous and non-homogeneous semi-markov reliability models," *Communications in Statistics - Theory and Methods*, vol. 33, no. 3, pp. 697 – 714, 2004.

[55] G. D'Amico, J. Janssen, and R. Manca, "Credit risk migration semi-markov models: a reliability approach," in *International Symposium on Applied Stochastic Models and Data Analysis (ASMDA 2005)*, 2005, pp. 950–957.

[56] E. Cinlar, *Introduction to Stochastic Processes.* Addison-Wesley Professional, 2002.

[57] L. Rossey, R. Cunningham, D. Fred, J. Rabek, R. Lippmann, J. Haines, and M. Zissman, "Lariat: Lincoln adaptable realtime information assurance testbed," in *IEEE Aerospace Conference*, 2002.

[58] B.-C. Jenq, W. H. Kohler, and D. Towsley, "A queueing network model for a distributed database testbed system," *IEEE Trans. Softw. Eng.*, vol. 14, no. 7, pp. 908–921, 1988.

[59] K. D. Kang, P. H. Sin, J. Oh, and S. H. Son, "A real-time database testbed and performance evaluation," in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2007.

[60] (2004) Tpc-c benchmark. [Online]. Available: http://www.tpc.org/tpcc/

[61] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and evaluating computer intrusion detection systems," *Commun. ACM*, vol. 42, no. 7, pp. 53–61, 1999.

[62] G. Vigna, W. Robertson, and D. Balzarotti, "Testing network-based intrusion detection signatures using mutant exploits," in *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security.* New York, NY, USA: ACM Press, 2004, pp. 21–30.

[63] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "Cost-based modeling for fraud and intrusion detection: results from the jam project," in *DARPA Information Survivability Conference and Exposition*, 2000, pp. 130 – 144.

[64] R. Bace and P. Mell, "Intrusion detection systems," NIST, Tech. Rep., 2001.

Appendix A

Two Pass Repair Algorithm

Ammann et al. proposed the two pass repair algorithm in [25]. The algorithm described below is composed of two passes. Pass one scans the log forward from the entry where the first bad transaction starts to locate every bad and suspect transaction. Pass two goes backward from the end of the log to undo all bad and suspect transactions.

Input: the log, the set **B** of bad transactions.
Output: a consistent database state in which all bad and suspect transactions are undone.
Initialization;
Let *commit_list* := {}, *undo_list* := **B**, *write_set* := {}, *tmp_write_set* := {}, *tmp_undo_list*;
`/* See Comment 1 */`
Locate the log entry where the first bad transaction B1 starts;
Scan forward until the end of the log. For each log entry;
if *the entry is for a transaction* T_i *in* **B then**
  if *the entry is a write record* $[T_i; x; v]$ **then**
    $write\_set := write\_set \cup \{x\}$;
  end
else
  case *the entry is a write record* $[T_i; x; v]$
    $tmp\_write\_set := tmp\_write\_set \cup \{(T_i, x)\}$; `/* See Comment 2 */`
  end
  case *the entry is a read record* $[T_i; x]$
    if T_i *is in the tmp_undo_list* **then**
      skip the entry;
    end
    if x *is in the write_set* **then**
      $tmp\_undo\_list := tmp\_undo\_list \cup \{T_i\}$; `/* See Comment 3 */`
    end
  end
  case *the entry is an abort record* $[T_i; abort]$
    delete all the data items of T_i from the *tmp_write_set*; **if** T_i *is in the tmp_undo_list* **then**
      delete T_i from the *tmp_undo_list*;
    end
  end
  case *the entry is a commit record* $[T_i; commit]$
    if T_i *is in the tmp_undo_list* **then**
      move T_i from the *tmp_undo_list* to the the *undo_list*;
      move all the data items of T_i from the *tmp_write_set* to the *write_set*;
    else
      delete all the data items of T_i from the *tmp_write_set*;
    end
  end
end
Scan backward from the end of the log to undo all the transactions in the *undo_list*;

Algorithm 1: Two Pass Repair Algorithm

Comments:

1. The *commit_list* consists of the transactions which commit after the first bad transaction. The *undo_list* consists of the bad and suspect transactions that should be undone. The *tmp_undo_list* is used to capture the set of in-repair good transactions that have read some dirty data. A transaction T is in-repair between the time we scan the record $[T, begin]$ and the time we scan the record $[T, commit]$ or the record $[T, abort]$. The *write_set* consists of the dirty data items. The use of *tmp_write_set* is explained in comment 2.

2. We need to keep track of the data items written by each in-repair good transaction because the transaction may be later on found suspect, and at that moment we need to add these data items to the *write_set*. There are basically two approaches to solve this problem. One, which is used in the algorithm, is to keep the write items in a temporary memory structure (namely, *tmp_write_set*); the other is to scan the log backward to figure out the write items later on when the transaction is found suspect (the backward scan can be efficient since all the write log entries of a transaction are chained together in the log). The first approach costs more memory space but is faster. The second approach costs less memory space but is slower since it may cause disk operations.Also, since we assume that the history to be repaired is strict, the following scenario, which happens in a history that is recoverable but not strict, will not occur:

$$r_{G_1}[x_1]w_{G_1}[x_1]r_{G_2}[x_1]w_{G_2}[x_1]r_{G_1}[y_1]w_{G_1}[y_1]c_{G_1}c_{G_2}$$

Suppose y_1 is in the *write_set* and x_1 and x_2 are not. When we encounter the entry $r_{G_2}[x_1]$, though G_2 is dependent upon G_1, we skip it according to the algorithm since x_1 is not in the *write_set*. Later when we encounter the entry $r_{G_1}[y_1]$, we will add G_1 to the *undo_list* since it reads an item in the *write_set*. But, at this point, G_2 will not be added to the *undo_list* though it has been affected by G_1.

3. We cannot put T_i into the *undo_list* because T_i may be later on found aborted.

Appendix B

Read and Write Set Templates of TPC-C Transactions

In TPC-C, the term database transaction as used in the specification refers to a unit of work on the database with full ACID properties, namely atomicity, consistency, isolation, and durability. A business transaction is composed of one or more database transactions. In TPC-C, a total of five types of business transactions are used to model the processing of an order (see [60] for the source codes of these transactions). The read and write Set templates of these transaction types are as follows.

B.1 New-Order Transaction

The New-Order transaction consists of entering a complete order through a single database transaction. The template for this type of transaction is ("+" denotes string concatenation):

1. **Input** =

   warehouse number(w_id), district number(d_id), customer number(c_id);

   a set of items(ol_i_id), supplying warehouses(ol_supply_w_id),

   and quantities(ol_quantity)

2. **Read_Set**=

   { Warehouse.w_id.W_TAX;

   District.(w_id+d_id).(D_TAX, D NEXT_O_ID);

Customer.(w_id+d_id+c_id).(C_DISCOUNT, C_LAST, C_CREDIT);

Item.ol_i_id.(I_PRICE, I_NAME, I_DATA);

Stock.(ol_supply_w_id+ ol_i_id).(S_QUANTITY, S_DIST, S_DATA,

S_YTD, S_ORDER_CNT, S_REMOTE_CNT) }

3. **Write Set**=

   { x = District.(w_id+d_id).D_NEXT_O_ID;

   New-Order.(w_id+d_id+x);

   Order.(w_id+d_id+x);

   R_1= ol_i_id;

   Order-Line.(w_id+d_id+x+R_1)}

B.2 Payment Transaction

The Payment transaction updates the customers balance, and the payment is reflected in the districts and warehouses sales statistics, all within a single database transaction. The templates for this type of transaction are:

1. **Input**=

   warehouse number(w_id), district number(d_id),

   customer number(c_w_id, c_d_id, c_id) or customer last name(c_last),

   and payment amount(h_amount)

2. **Read Set**=

   { Warehouse.w_id.(W_NAME, W_STREET_1, W_STREET_2, W_STATE, W_YTD);

District.(w_id+d_id).(D_NAME, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP, D_YTD);

[**Case 1**, the input is customer number:

Customer.(c_w_id+c_d_id+c_id).(C_FIRST, C_LAST, C_STREET_1, C_STREET_2, C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM, C_DISCOUNT, C_BALANCE, C_YTD_PAYMENT, C_PAYMENT_CNT, C_DATA);

Case 2, the input is customer last name:

Customer.(c_w_id+c_d_id+c_last).(C_FIRST, C_LAST, C_STREET_1, C_STREET_2, C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM, C_DISCOUNT, C_BALANCE, C_YTD_PAYMENT, C_PAYMENT_CNT, C_DATA)]}

3. **Write Set**=

{ Warehouse.w_id.W_YTD; District.(w_id+d_id).D_YTD;

[**Case 1**, the input is customer number:

Customer.(c_w_id+c_d_id+c_id).(C_BALANCE, C_YTD_PAYMENT, C_PAYMENT_CNT);

History.(c_id+c_d_id+c_w_id+d_id+w_id).*

Case 2, the input is customer last name:

Customer.(c_w_id+c_d_id+c_last).(C_BALANCE, C_YTD_PAYMENT,

C_PAYMENT_CNT);

History.(c_d_id+c_w_id+d_id+w_id).*]}

B.3 Order-Status Transaction

The Order-Status transaction queries the status of a customers most recent order within a single database transaction. The templates for this type of transaction are:

1. **Input**=

   customer number(w_id+d_id+c_id) or customer last name(w_id+d_id+c_last)

2. **Read Set**=

   { [**Case 1**, the input is customer number:

   Customer.(w_id+d_id+c_id).(C_BALANCE, C_FIRST, C_LAST, C_MIDDLE);

   Case 2, the input is customer last name:

   Customer.(w_id+d_id+c_last).(C_BALANCE, C_FIRST, C_LAST, C_MIDDLE)] ;

   x=Order.(w_id+d_id+c_id).O_ID;

   Order.(w_id+d_id+c_id).(O_ENTRY_D, O_CARRIER_ID);

   Order-line.(w_id+d_id+x).(OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY,

   OL_AMOUNT, OL_DELIVERY_D) }

3. **Write Set**=ϕ

B.4 Delivery Transaction

The Delivery transaction processes ten new (not yet delivered) orders within one or more database transactions. The templates for this type of transaction are:

1. **Input**= warehouse number(w_id), district number(d_id), and

   carrier number(o_carrier_id)

2. **Read Set**=

   {R_1=New-Order.(w_id+d_id).NO_O_ID;

   R_2=Order.(w_id+d_id+R_1).O_C_ID;

   Order.(w_id+d_id+R_1).(O_CARRIER_ID, OL_DELIVERY_D, OL_AMOUNT);

   Customer.(w_id+d_id+R_2).(C_BALANCE, C_DELIVERY_CNT)}

3. **Write Set**=

   {R_1=New-Order.(w_id+d_id).NO_O_ID;

   R_2=Order.(w_id+d_id+x).O_C_ID;

   Order.(w_id+d_id+R_1).O_CARRIER_ID;

   Customer.(w_id+d_id+R_2).(C_BALANCE, C_DELIVERY_CNT);

   New-Order.(w_id+d_id+R_1);

   Order-Line.(w_id+d_id+R_1).OL_DELIVERY_D}

B.5 Stock-Level Transaction

The Stock-Level transaction retrieves the stock level of the last 20 orders of a district. The templates for this type of transaction are:

1. **Input**= warehouse number(w_id), district number(d_id),

2. **Read Set** =

   {x=District.(w_id+d_id).D_NEXT_O_ID;

   R_1={$x-1$,...,$x-19$,$x-20$};

R_2=Order-Line.(w_id+d_id+R_1+ OL_NUMBER).OL_I_ID;

Stock.(w_id+R_2).S_QUANTITY}

3. **Write Set**=ϕ

Vita

EDUCATION

- Pennsylvania State University, University Park, PA
  Ph.D. Candidate in Information Sciences and Technology (IST)

- Jilin University, Jilin, China, P.R.
  M.S. in Electrical Engineering (EE)
  B.S. in Electrical Engineering (EE)

SELECTED PUBLICATION

- **Hai Wang** and Peng Liu
  Survivability Evaluation: Modeling Techniques and Measures
  Handbook of Research on Information Security and Assurance
  Proposal accepted, chapter in review

- **Hai Wang**, Peng Liu, and Lunquan Li
  Evaluating the Survivability of Intrusion Tolerant Database Systems and the Impact of Intrusion Detection Deficiencies
  International Journal of Information and Computer Security (IJICS)
  Accepted, to appear in January 2008

- **Hai Wang** and Peng Liu
  Modeling and Evaluating the Survivability of an Intrusion Tolerant Database System 11th European Symposium on Research in Computer Security (ESORICS 2006), September 2006 (Acceptance ratio 20%=32/160)

- **Hai Wang**, Peng Liu, and Lunquan Li
  Evaluating the Impact of Intrusion Detection Deficiencies on the Cost-Effectiveness of Attack Recovery
  7th Information Security Conference (ISC 2004), Lecture Notes in Computer Science 3225/2004, pages 146-157, September 2004 (Acceptance ratio 34%=36/106)

RESEARCH PROJECTS

- **Self-Healing Database Systems Project**, supported by NSF and DARPA
  Investigating innovative techniques for building highly-survivable database systems

- **QoIA Management Project**, supported by NSF
  Building a new trusted computing infrastructure that is able to continue delivering QoIA services in face of attacks and cost constraints