The Design of an Adaptive Intrusion Tolerant Database System

Pramote Luenam
Dept. of Info. Systems
UMBC, Baltimore, MD
pluena1@umbc.edu

Peng Liu
Dept. of Info. Systems
UMBC, Baltimore, MD
pliu@umbc.edu

Abstract

This paper presents the design of an adaptive intrusion tolerant database system, called AITDB. The goal of AITDB is to provide database applications with a stabilized level of data integrity and availability in face of attacks. Using a rule-based adaptation mechanism and a set of reconfiguration operators, AITDB automatically adapts itself to the dynamic changes of environment according to a specific set of adaptation criteria, which are determined based on the current situation of the damage (i.e., data integrity level), the attacks, the workload, the availability level, the performance, and the cost.

1 Introduction

A DBMS must provide the security, integrity, concurrency, and recovery controls to protect the database against a variety of possible threats that are either intentional or accidental. However, it is difficult, perhaps impossible to build a DBMS that can protects the database from every unauthorized use, misuse, or abuse. Many security techniques are proposed, developed, deployed, and evaluated both in academic research and commercial DBMS development, for example, login authentication, access and inference controls [1,4,5,10], encryption, multilevel secure databases [11,12] and multilevel secure transaction processing [2]. These techniques focus on preventing users from gaining access beyond their authorities. Such a preventive approach is very limited to handle successful attacks, or intrusions, which can seriously jeopardize the integrity and availability of databases. An intrusion tolerant database system, on the other hand, arms a prevention-centric secure database system with the ability to *survive* intrusions, hence it evolves a prevention-centric secure database system into a defense-in-depth resilient database system.

In our previous research [7,8], we have designed and implemented *ITDB*, an intrusion tolerant database system (prototype), using "off-the-shelf" components. ITDB illustrates intrusion tolerance design principles in three ways: (1) using multiple intrusion tolerance phases to achieve defense-in-depth; (2) using isolation and multi-phase damage containment to tolerate a not so good intrusion detector; (3) on-the-fly intrusion tolerance transparent to applications. ITDB focuses on (a) the data integrity and availability loss caused by authorized but malicious transactions, which according to the fact that most attacks are from insiders should be the major threat to database systems, and (b) transaction level intrusion tolerance mechanisms. Although ITDB does not directly address processor, OS, or DBMS level attacks, existing lower level database intrusion tolerance mechanisms such as those proposed in [9,13] can be easily integrated to enable ITDB to survive lower lever attacks.

In this paper, we present the design of an adaptive intrusion tolerant database system called AITDB. AITDB is designed to enhance the cost-effectiveness of ITDB through dynamic reconfiguration. Using a rule-based adaptation mechanism and a set of reconfiguration operators, AITDB automatically adapts itself to the dynamic changes of environment according to a specific set of adaptation (or tuning) criteria, which are determined based on the current situation of the damage (i.e., the data integrity level), the attacks, the workload, the availability level, the performance, and the cost.

The rest of the paper is organized as follows. In this section, we present the overview of ITDB and justify the need for adaptive intrusion tolerance. Section 2 presents the AITDB architecture. Section 3 presents the AITDB adaptation model. Section 4 describes AITDB rule-based reconfiguration mechanism. In section 5, we describe the major reconfiguration operators of AITDB. In Section 6, we present the system design of AITDB. Finally, we conclude the paper in Section 7.

1.1 ITDB Overview

AITDB is developed in the context of ITDB. ITDB is a transaction-level intrusion tolerant database system

that can survive attacks. The architecture of ITDB architecture is shown in Figure 1.

In order to build ITDB on top of an "off-the-shelf" DBMS, we developed a transaction proxy component, called the *Mediator* (MD). The MD not only captures a lot of information (or trails) about transactions' behavior but also help enforce some important intrusion tolerance policies such as isolation and damage containment. Using the transaction logs, the trails captured by the Mediator, and some other relevant proofs, the *Intrusion Detector* (ID) detects malicious or bad transactions. If a bad transaction is active when being identified, the transaction will be aborted by the Mediator. However, if a bad transaction is already committed when being identified, the bad transaction's identifier will be sent to the *Damage Assessor* (DA).

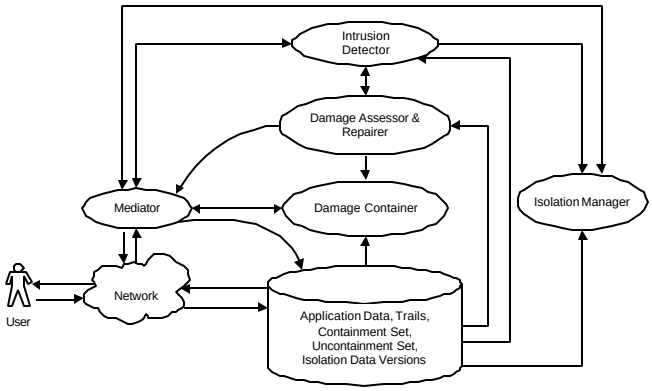

Figure 1: ITDB system architecture

The DA and the *Damage Repairer* (DR) *assess* the damage caused by the intrusion and performs on-the-fly damage repair. The growing transaction logs (and other trails) are scanned to locate the set of data objects corrupted directly or indirectly by the intrusion. Then, each damaged data object *x* will be repaired by a specific *cleaning* transaction (composed by the DR), which will restore *x*'s value to its latest undamaged version. ITDB continues executing new transactions during the repair process.

Since during the *detection latency* and *assessment latency* a lot of damage could be spread from a damaged data object to some other originally undamaged objects through the read and write operations of *good* transactions, ITDB integrates two more components, namely the *Damage Container* (DC) and the *Isolation Manager* (IM), to control the extent of damage spreading. The DC takes a novel *multi-phase damage containment* approach which first instantly contains the damage the might have been caused by an intrusion as soon as the intrusion is identified, then tries to uncontain the objects that are previously contained by mistake. Multi-phase damage containment can ensure that no damage will spread during the assessment latency, although with some availability lost. However, the DC can do nothing to reduce the damage caused during the detection latency. In contrast, the IM can reduce the damage caused during the detection latency (thus it indirectly reduces the damage caused during the assessment latency) by redirecting the access of a *suspicious* transaction (that is very likely to cause damage later on) to a virtually separated database. Isolation immunizes the database from the damage caused by the set of suspicious transactions without sacrificing substantial availability, since if an isolated user turns out to be innocent, most - if not all – of his or her updates can be *merged* back to the real database.

1.2 The need for adaptivity

ITDB components can behave in many different ways. At one point of time, the *resilience* of an ITDB system is primarily affected by (1) the current attacks; (2) the current workload; (3) the current data integrity level; (4) the current data availability level; and (5) the current *behavior* of the system. We call the first 4 factors the *environment* of the ITDB system. It is clear that given the same environment, two ITDB systems with different behaviors can yield very different levels of resilience. This suggest that one ITDB system behavior is only good for a limited set of environments. To achieve the maximum amount of resilience, an ITDB system must *adapt* itself to its environment. Through AITDB, (1) we can adapt ITDB to (different) application semantics. (2) We can significantly improve the cost-effectiveness of ITDB. (3) We can prevent dramatic performance degradation due to system environment changes.

2 Aitdb architecture

As mentioned earlier, AITDB is designed to have a self-tuning capability, which can adaptively adjust its behavior according to different adaptation criteria. For this purpose, we introduce four additional components to ITDB: the *Self-Stabilization Manager* (SSM), the *Reconfiguration Executor*, the *Emergency Analyzer*, and the *Listener*, as illustrated in Figure 2. The SSM is a new component while the Listener, the Emergency Analyzer, and the Reconfiguration Executor are additional threads added to the original components of ITDB.

In this section, we describe the function and purpose of these additional components. AITDB exploits two categories of system parameters to do reconfiguration: *control parameters*, which control the system behavior, and *monitor parameters*, which specify the environment. When an AITDB system starts, the SSM retrieves all the system parameters (together with their values) from the Monitor_Parameter

table. The parameters are initialized and then temporarily stored into the Parameters_Table. Using a specific rule-based adaptation mechanism which we will address shortly, the SSM analyzes the parameters and makes a decision on how to adjust the control parameters to improve the overall system cost-resilience. Then reconfiguration messages containing the suggested new values for these control parameters are prepared and sent to the corresponding components. Reconfiguration messages have the following format:

Message_Type; Sender; Receiver; Parameter_Name; Value; Measurement_Unit; Action

Each message contains seven fields. Each field is separated by semicolon. The first field, Message type, indicates the type of the message. Its value determines the format of the remaining fields. The Sender field indicates the name of the sending component. The Receiver field indicates the name of the destination component. The Parameter_Name field indicates the parameter that need to be changed. The Value field is the new value of the parameter suggested by the SSM. The Measurement_Unit field indicates the unit of measurement. Possible values are RANGE, PERCENT, UNIT and LEVEL. The last field, Action, specifies how the operation will be applied to the parameter.

For example, in order to inform the Mediator to change the Suspicion Level Threshold control parameter to the forth level, the SSM can send the following reconfiguration message:

NORMAL; SSM; Mediator; Suspicion Level Threshold; 4; LEVEL; CHANGETO;

To retrieve a reconfiguration message, a Listener will be added to each ITDB component. The Listener is a thread that is responsible for checking if a new reconfiguration message arrives for the component. When a new message arrives, the Listener will notify and pass the message to the Reconfiguration Executor. The Reconfiguration Executor is another thread that will be added to several components such as the Intrusion Detector and the Mediator. The thread is responsible for reconfiguring the component by changing the values of its control parameters from the old values to the new values.

The SSM periodically polls the parameters from the ITDB components. The polling time interval is specified in the Interval_Time table (i.e., every 30 seconds). However, in some emergency cases, AITDB might need to react instantly. For example, when a transaction is identified malicious with the anomaly degree over the maximum threshold. For such an event, the Intrusion Detector should not wait until the time interval has lapsed and it should immediately report this event to the SSM. To support this feature, we add a new thread, namely the Emergency Analyzer, to each component to analyze, detect, and report emergency events. Similar to the SSM, the Emergency Analyzer also uses a rule-based mechanism to determine which events should be treated as an emergency.

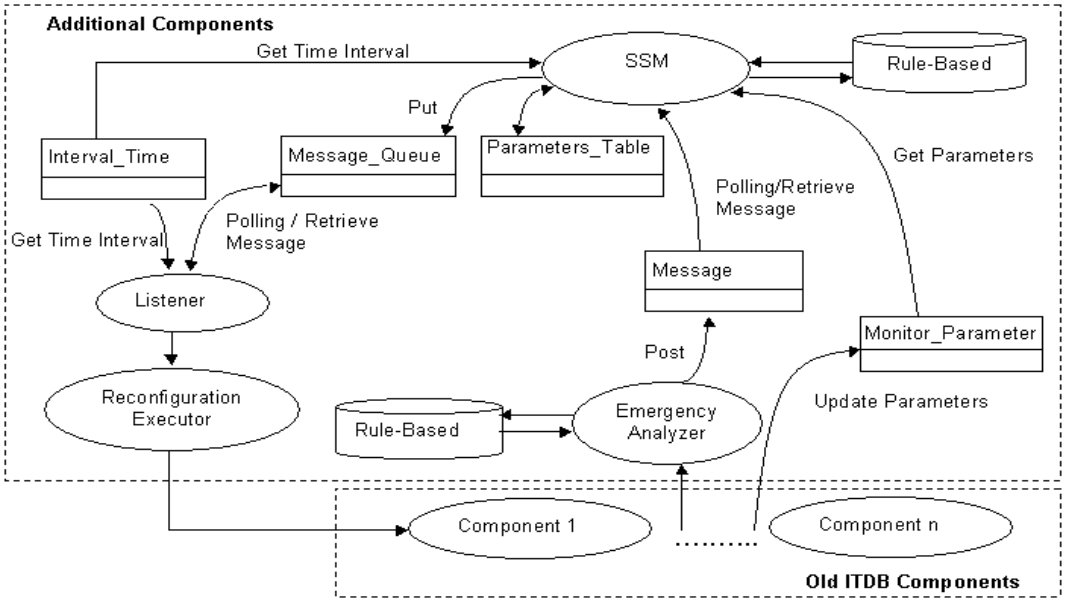

Figure 2: AITDB system architecture

3 THE Adaptation MODEL

3.1 Adaptation Criteria

AITDB does reconfiguration according to two adaptation criteria: (1) the (resulted) levels of data integrity and availability, and (2) the cost-effectiveness. Since the goal of AITDB is to provide database applications with a stabilized level of data integrity and availability, it is reasonable to use the current levels of data integrity and availability and the changes of these levels over a period of time to measure the *resilience* of an AITDB system. In particular, AITDB uses four trustworthiness *metrics* to measure the effectiveness of adaptation operations: (1) level of data integrity (denoted LI), which is indicated by the percentage of the damaged data objects (to all the data objects); (2) level of data availability from the perspective of containment (denoted LDA), which is indicated by the percentage of the data objects contained by the Damage Container; (3) level of data availability from the perspective of false alarms (denoted LTA), which is indicated by the percentage of the good transactions that are mistakenly rejected due to false alarms to all the good transactions; (4) level of data availability from the perspective of isolation (denoted LIA), which is indicated by the percentage of the innocent transactions that are backed-out during merging processes. Note that containment, false alarms, and isolation can all cause availability loss since contained data items are temporarily not accessible, false alarms can cause innocent transactions to be denied, and the merge after an isolated user is proven innocent could roll back transactions.

To measure the cost-effectiveness of an AITDB system, we use three cost-effectiveness *metrics*: (1) level of system workload (denoted LSW), which is indicated by the degree of workload decreasing; (2) level of the system effectiveness (denoted LSE), which is indicated by a group of effectiveness to cost ratios; (3) level of the system attacks (denoted LSA), which is indicated by how intense the attacks are.

Both the trustworthiness and the cost-effectiveness metrics embody themselves through the values of the set of monitor parameters, which are listed in Table 1, where the effects of every monitor parameter on these metrics are specified. AITDB measures the adaptation criteria metrics through these monitor parameters.

3.2 Adaptation model

In AITDB, the environment is monitored through the set of monitor parameters. And the reconfiguration is performed through several key *reconfiguration operators*. These reconfiguration operators, when triggered by the environment changes, will do the reconfiguration by changing the values of the set of control parameters in such a way that the adaptation criteria could be maximized. These new control parameters will determine the behaviors of the ITDB components during the next AITDB adaptation interval. And these adjusted behaviors will affect the values of the monitor parameters during the next AITDB adaptation interval (as a payback). The whole adaptation process is shown in Figure 3.

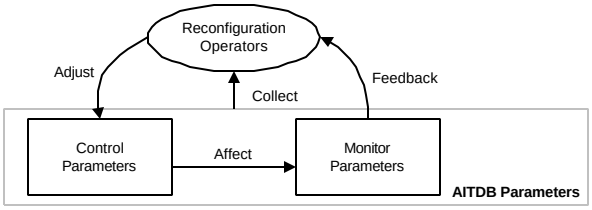

Figure 3: Adaptation Model

Parameter	Description	Related Component	Category	Effect
Transaction Submission Rate (TSR)	The average number of transactions submitted to the database within a time period.	MD	LSW	TSR↑ → LSW↑
Malicious Users to Trustworthy Users Ratio (MTR)	The percentage of the malicious users to the innocent users.	MD	LI	MTR↑→ LI↓
Transaction Processing Cost (TPC)	The average transaction processing time within a particular period.	MD	LSE	TPC↑→ LSE↓
Number of Aborted Transactions (NAT)	The total number of transactions aborted by the Mediator.	MD	LSA	NAT↑→ LSA↑
Detection Latency (DL)	The average delay time for detecting an intrusion.	ID	LSE	DL↑→ LSE↓
Intrusion Detection Cost (IDC)	The average time spent for detecting an intruder (starting from the beginning of the transaction until the malicious behavior is detected).	ID	LSE	IDC↑→ LSE↓

Suspicious Transactions to Malicious Transactions Ratio (SMR)	The ratio between the suspicious transactions and the malicious transactions.	ID	LSE	SMR↑→ LSE↓
Suspicion Rate (SPR)	The average number of the suspicion transactions detected by the Intrusion Detector within a particular period.	ID	LSA	SPR↑→ LSA↑
False Alarm Rate (FAR)	The percentage of the number of suspicious transactions that are found innocent later to the number of all suspicious transactions.	ID	LTA	FAR↑→ LTA↓
Affected Item Size (AIS)	The total number of affected items at a particular time.	DA & DR	LI	AIS↑→ LI↓
Number of Affected Transactions (NAF)	The average number of affected transactions in a specific period.	DA & DR	LSW	NAF↑→ LSW↑
Affected Items Rate (AIR)	The percentage increase (or decrease) in the number of affected items between two periods. AIR can be used to determine the trend of the damage spreading.	DA & DR	LSA	AIR↑→ LSA↑
Repair Cost (RPC)	The average time required for the repair operation.	DA & DR	LSE	RPC↑→ LSE↓
Average Number of Isolated Users (ANI)	The average number of suspicious users who are isolated by the Isolation Manager within a particular period.	IM	LSW	ANI↑→ LSW↑
Merging Size (MS)	The number of objects kept in the merged back history at a particular time.	IM	LSW	MS↑→ LSW↑
Merging Back Ratio (MBR)	The ratio between merged back objects and all database objects	IM	LSW	MBR↑→ LSW↑
Isolated to Normal Transactions Ratio (INR)	The ratio between the isolated transactions and the normal transactions.	IM	LIA	INR↑→ LIA↓
Back-out Ratio (BOR)	The percentage of the back-out transactions to the Merging Size.	IM	LIA	BOR↑→ LIA↓
Merging Cost (MC)	The average time required for the merging back operation per transaction.	IM	LSE	MC↑→ LSE↓
Isolation Cost (IC)	The average processing time required for each isolated user (from the beginning until merged back or discarded) within a particular time period.	IM	LSE	IC↑→ LSE↓
Containment Set Size (CSS)	The number of objects kept in the containment set.	DC	LSW	CSS↑→ LSW↑
Blocked Transaction Ratio (BTR)	The percentage of transactions denied when accessing contained objects due to the contained operation.	DC	LDA	BTR↑→ LDA↓
Containment Cost (CC)	The average time spent for processing each object in the containment operation (starting from the time an object is contained until it is uncontained).	DC	LSE	CC↑→ LSE↓
Data Integrity level (DIL)	Data Integrity Level = 1 – the percentage of the known damaged objects.	Database	LI	DIL↑ → LI↑
Transaction Size (TSS)	The average number of read and write operations in a transaction.	Database	LSW	TSS↑→ LSW↑
Transaction Execution Time (TET)	The average time spent for executing each transaction within a particular time period.	Database	LSE	TET↑→ LSE↓

Table 1: List of AITDB monitor parameters

4 Reconfiguration Rules

Rules are used by the SSM (i.e., the set of reconfiguration operators) to specify and program AITDB reconfiguration *policies*. In our rule-based tuning processes, we start by examining the events as well as the relevant system parameters that might affect the system's behavior and cost-effectiveness. When an interesting event arrives, we first check if the reconfiguration firing conditions (associated with the event) are satisfied or not. These conditions indicate the amount of resilience or cost-effectiveness degradation that could lead to system crash (in terms of security). If these conditions are satisfied, we then instantly respond to the event with a set of changed control parameters (i.e., a new defense behavior). Currently, AITDB rules are provided by the system-security-officer based on his or her experiences. Each AITDB rules, after being generated, are transformed into a decision chart that is reviewed by experts to examine the logic and feasibility of its implementation. Figure 4 illustrates a sample rule used in AITDB

When: Get the Average Response Time report from the Mediator
If: The Average Response Time from the Mediator > 20 Second (High) and
The Number of Malicious Users < 2 (Low) and
The Percentage of Affected Transactions < 5% (Low) or
The Percentage of Isolated Transactions < 5% (Low)
Then: 1. Switch Multiphase Damage Containment to One Phase; 2. Adjust the Suspicious Anomaly Threshold to Lower Level

Figure 4: A sample reconfiguration rule

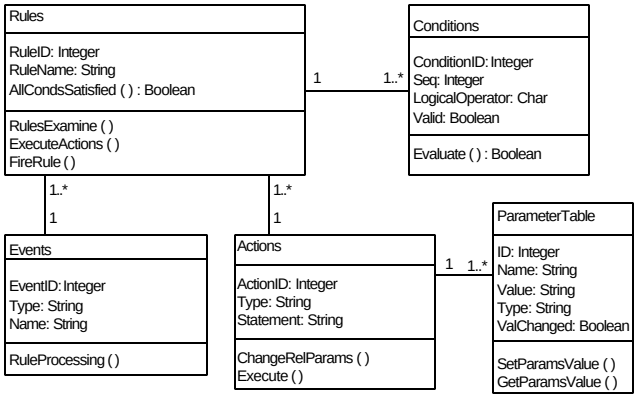

Figure 5: Class Diagram of Rules

AITDB rules use an Event-Condition-Action model. They have the following form:

When <*event*> if <*conditions*> then <*actions*>

Actions are executed when a rule containing them is fired. A rule fires whenever all of its conditions are satisfied. The class diagram shown in Figure 5 presents the relationships, methods, and attributes of the classes used in our rule-based reconfiguration mechanism.

5 reconfiguration operators

In this section, we present three key reconfiguration operators of AITDB. Each operator is associated with an *adaptive* intrusion tolerance *facility* (or component) that is controlled by a specific set of control parameters. Each operator reconfigures the corresponding facility by changing the values of the control parameters of the facility. In particular, the Anomaly Level Threshold Selector is associated with the adaptive intrusion detector of AITDB. The TP-Speed Controller is associated with the adaptive Transaction Proxy (a part of the Mediator). And the Containment Tuner is associated with the adaptive Damage Container.

5.1 Anomaly Level Threshold Selector

In AITDB, the Intrusion Detector uses synthesized *anomaly level* of transactions (or sessions) to raise alarms, and there are two kinds of anomaly level thresholds: (1) suspicious anomaly level threshold (ST), and (2) malicious anomaly level threshold (MT). The suspicious anomaly level threshold is used by the Isolation Manager to determine whether or not a transaction is suspicious and should be isolated. The malicious anomaly level threshold is used by the Intrusion Detector to confidently report intrusions (although the detector can make mistakes). Both thresholds are control parameters that directly affect several AITDB monitor parameters such as suspicion rate, false alarm rate, the number of malicious transactions, and the intrusion detection cost. When both ST and MT are high, fewer intrusions are detected with fewer false alarms, fewer transactions are isolated, the intrusion tolerance cost is reduced, more data availability is there, but the data integrity level could be seriously decreased. When both are low, the data integrity level is increased, but (a) the intrusion tolerance cost is increased; (b) more false alarms are raised; and (c) the data availability is jeopardized. The job of the anomaly level threshold selector is to adaptively pick the right values for ST and MT in such a way that the tuned Intrusion Detector is more effective within the current environment. An example reconfiguration policy involving the anomaly level threshold selector is shown in Figure 4.

5.2 TP-Speed Controller

The detection latency and assessment latency can be longer when the attacks are intense or when the DBMS workload is too heavy. As a result, serious damage spreading can be caused during the two latencies. To avoid the problem, we may want to reduce the value of a control parameter, namely the *transaction submission rate*, by using the speed control mechanism that can be enforced by the Mediator. By adding an appropriate extra delay to each transaction submitted to the database server, the overall workload can be reduced. As a result, the detection latency and assessment latency can be shortened and the amount of damage spreading can be controlled. However, it should be noticed that the extra delays for transaction processing (TP) jeopardize the data availability and increases the amount of database performance penalty of AITDB. Figure 6 shows an example reconfiguration policy involving the TP-Speed Controller.

When: Get the Suspicion Rate from the Intrusion Detector
If: The Suspicion Rate > 10% (High) and
The Integrity Level < 80% (Low) and
The Transaction Submission Rate > 20 Transactions/Second (High)
Then: Ask the Mediator to increase the Transaction Delay Time by 5%

Figure 6: A sample rule using the speed controller

5.3 Containment Switcher

Traditional database damage containment is *one-phase*, that is, a damaged data object is contained only after the Damage Assessor finds that it is damaged. One-phase damage containment approach has a serious drawback, that is, it cannot prevent the damage caused on the objects that are corrupted but not yet located from spreading. ITDB uses a multi-phase damage containment approach to solve this problem. Multi-phase damage containment can guarantee that after the containing phase no damage will spread. However, since multi-phase damage containment could mistakenly contain a lot of undamaged data objects during the containing phase, substantial data availability loss could be temporarily caused. This motivates AITDB to do adaptive damage containment.

The simplest adaptive damage containment can be done by *switching* the system from multi-phase to one-phase or in the reverse way. In particular, when damage spreading is too serious, we should switch the system to multi-phase containment to increase the data integrity level. On the other hand, when damage spreading is not serious at all, we should switch the system back to one-phase containment to increase the data availability level. An example reconfiguration policy involving the Containment Switcher is shown in Figure 4. Better adaptive damage containment could be done if we support approximate damage containment where some damage is allowed to leak out to trade in more data availability. However, this is out of the scope of this paper.

6 System Design

This section provides the detailed overview of our AITDB design. Our design is object-oriented. The design is presented and organized into four major components: the Listener, the Reconfiguration Executor, the SSM, and the Emergency Analyzer. To describe the relationships and dependencies among these components, we use the diagram illustrated in Figure 7.

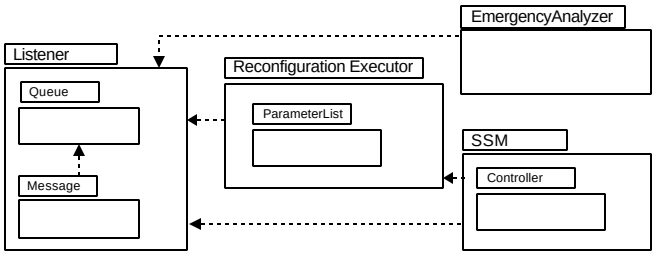

Figure 7: AITDB Component Design

The Listener is responsible for handling communications between components through a message queue. The Listener is an instance that is implemented by objects of class Message and class Queue. The Message class provides methods for checking and receiving incoming messages, and posting an outgoing message to any other component. An instance of the Queue class is a template, which turns to be instantiated as objects that will be used to store messages. The Queue class provides methods for placing messages into the queue and getting a message from the queue.

The Reconfiguration Executor thread is added into every component that needs to reconfigure itself. The control parameters, as well as their suggested new values, are passed from the Listener as an argument. This thread is implemented by objects of class Reconfiguration Executor and class ParameterList. The Reconfiguration Executor class uses an object created from a template class, the ParameterList. The Reconfiguration Executor class provides a method for changing the parameters to the optimized values. The ParameterList class is a template class that defines a generic container for storing collections of parameters. Each parameter associates a name with a value. Each parameter belongs to one specific ITDB component. The ParameterList class also provides useful methods for maintenance the data in the list.

The SSM retrieves parameters from all the components of ITDB. These parameters are then analyzed to determine the optimized values before being sent back to the corresponding components through a message queue. The SSM is an instance of class Controller. The Controller class provides methods for collecting parameters from ITDB components and the database as well.

The Emergency Analyzer handles emergency situations. It uses some relevant rules to decide if the situation is an emergency and should be reacted instantly. The Emergency Analyzer is an object of class EmergencyAnalyzer. The EmergencyAnalyzer class provides methods for collecting parameters from a component. It also provides a method to notify the SSM whenever an emergency has been identified.

7 Conclusion

This paper presents the design of an adaptive intrusion tolerant database system. AITDB is developed in the context of ITDB -- a transaction-level intrusion tolerant database system. The goal of AITDB is not only to enhance the cost-effectiveness of ITDB but also to provide applications with a stabilized level of data integrity and availability. AITDB is designed to have a self-tuning capability, which adaptively tunes itself to the dynamic changes of environment.

There are some future works for AITDB. First, we want to implement the prototype of AITDB to investigate our design goal. Second, we want to test and evaluate the effectiveness and performance of AITDB using real world data such as credit card transaction data and inventory management data. Finally, we want to apply such intelligent techniques as fuzzy logic or neural network to enhance the cost-effectiveness of AITDB.

Acknowledgement

Luenam and Liu were supported by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0575.

References

[1] M. R. Adam. Security-Control Methods for Statistical Database: A Comparative Study. ACM Computing Surveys, 21(4), 1989.

[2] V. Atluri, S. Jajodia, and B. George. Multilevel Secure Transaction Processing. Kluwer Academic Publishers, 1999.

[3] Carter and Katz. Computer Crime: An Emerging Challenge for Law Enforcement. FBI Law Enforcement Bulletin, 1(8), December 1996.

[4] P. P. Griffiths and B. W. Wade. An Authorization Mechanism for a Relational Database System. ACM Transactions on Database Systems, 1(3):242–255, September 1976.

[5] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In Proceedings of ACM SIGMOD International Conference on Management of Data, pages 474–485, May 1997.

[6] P. Liu and S. Jajodia. Multi-phase damage confinement in database systems for intrusion tolerance. Proceeding of 14th IEEE Computer Security Foundations Workshop, 2001. To appear.

[7] P. Liu, J. Jiwu, P. Luenam and S. Ingsriswang. Intrusion Tolerant Database Systems. Intrusion Tolerant Database Systems. Submitted for Conference Publication.

[8] P. Luenam and P. Liu. ODAR: An On-the-fly Damage Assessment and Repair System for Commercial Database Applications. Proc. 15th IFIP WG 11.3 Working Conference on Database and Application Security, July 15-18, Ontario, Canada. To appear.

[9] U. Maheshwari, R. Vingralek, and W. Shapiro. How to build a trusted database system on untrusted storage. In Proceedings of 4th Symposium on Operating System Design and Implementation, San Diego, CA, October 2000.

[10] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. ACM Transactions on Database Systems, 16(1):88–131, 1994.

[11] R. Sandhu and F. Chen. The multilevel relational (mlr) data model. ACM Transactions on Information and Systems Security, 1(1), 1998.

[12] M. Winslett, K. Smith, and X. Qian. Formal query languages for secure relational databases. ACM Transactions on Database Systems, 19(4):626–662, 1994.

[13] D. Barbara, R. Goel, and S. Jajodia. Using checksums to detect data corruption. In *Proc. 2000 International Conference on Extending Data Base Technology*, Mar 2000.