An Empirical Evaluation on Attack Resilient Database Systems

Tianhao Tong
Department of Computer Science
UVA
Charlottesville, VA 22904
tt4cc@virginia.edu

Abstract

As more business has been shifted online, the database is a vital part of many critical systems. Many researches have been done on keeping database safe. While, however, admittedly, no defense mechanism could prevent every attack and some attack will unavoidably succeed. Therefore, the idea of attack resilient database system is proposed. However, how well the proposed solution works and why they still not being widely adapted is still a problem on which only limited researches have been done. This paper evaluates the current solution while points out some potential reason on why the current systems has not been widely adapted.

Keywords: Database Security, Survivability, Attack Resilient, Survivability evaluation.

1 Introduction

With the develop of wireless network connection, the pervasive computing not only push computation into everywhere, but also maximize our dependence on networked computer systems. Therefore, more and more companies have shifted their business to the Internet. Due to the large scale of Internet, the businesses can serve incredible number of customers online at the same time with very low cost. As a result, many sensitive transactions are completed online. Therefore, with this increased productivity, the loss will be correspondingly huge if the site is out of service.

Being a critical component of nearly every business information system, database is crucial to every part of the business logic and will cause whole system halt if the database went offline. However, although a lot of approaches have been developed to secure the database in multi-layer – from hardware and OS to the transaction and application layer – attacks do happen from time to time. For example, a security flaw in the OS level may let the intruder to circumvent the security policy, get control of the whole database and issue some malicious transactions to the database.

Database security concerns the confidentiality, integrity, and availability of data stored in a database. Many researches have been conducted on confidentiality as well as how to identify the attack.

However, we must admit that even the best defense cannot hold every intrusion. Even the most obvious one cannot be prevented at their outset. The success of attacks is unavoidable to some degree. However, there are very limited researches on how to survive successful database attacks. And among those proposed attack resilient database who claims can successfully preserve the availability and integrity of database under attack, little empirical evaluation has been done to quantify and measure how well a system can do.

Therefore, in this paper, I will study several state-of-art attack resilient database schemes, evaluate the level of survivability they can achieve and point out the potential
improvement we can do on those existing solutions.

The reset of the paper is organized as follows. In Section 2, I will present a discussion on related work. In Section 3, I will briefly describe the existing solutions on attack resilient database system. In Section 4, the metrics used in this evaluation will be defined. In Section 5, I will describe the evaluation environment. The results of my evaluation will be presented in Section 6 and Section 7 concludes the paper with pointing out what improvements can be done in future research.

2 Related Work

The concept of attack resilient, or intrusion tolerance, has been addressed by many early work such as information warfare[2], survivable software architectures[6], survivability of networks[4], survivability of storage[7], etc.

Some research has also been done in database survivability and evaluation. Such as [9, 10, 11, 12].

At the same time, the intrusion detector is also a critical part to the attack resilient database. Works in IDS includes [3, 5].

3 Attack Resilient Database Architectures

![Diagram of Attack Resilient Database](image)

Attack Resilient Database has been studied in [10]. The main task is to guarantee that damage spreading is (dynamically) controlled in such a way that the database will not be damaged to a degree that is unacceptable or useless.

The diagram of the system is shown in figure 1.

The two components important to this paper are the Damage Confinement Manager and Damage Repairer. Damage Confinement Manager is in charge of confine the damaged data and Damage Repairer is in charge of repair damaged data.

The difficulties of achieving the attack resilience is: 1) Execution of new transactions continues when the confinement and repair are going on and therefore the damage will spread to new transaction; and 2) even the cleaned objects can be re-damaged by new transaction.

In order to address this problem, the attack resilient database system presented in [10] employs such a basic idea that at the time of receiving intrusion alter, the database first confine all data that is altered after the intrusion happens, and then scan the log to do the unconfinement and repair.

Therefore, in this paper, I will evaluate this system using this algorithm with three configurations, namely, damage containment only, damage repair only and damage containment & repair.

4 Metrics for Evaluation

In order to quantify the evaluation we need to define the metrics for evaluation.

4.1 Integrity Metrics

In [9], the author, very intuitively, defines the integrity level of data item $o$ at time $t$ as follow:
\[ I(o, t) = \begin{cases} 0 & \text{data item is damaged} \\ 1 & \text{data item is good} \end{cases} \]

\( I(o, t) = 0 \) represents the data item \( o \) at time \( t \) has been altered or destructed by intrusions, and has not been repaired. \( I(o, t) = 1 \) represents the state of data item \( o \) at time \( t \) is good or the data item has been repaired after intrusion.

Then the author defines the integrity level of the whole database system \( D \) at time \( t \) as follow:

\[ I(D, t) = \frac{\sum_{i=1}^{N} w_i I(o_i, t)}{N} * 100\% \]

Here, each data object could have different weight and hence some data could be more important than others.

### 4.2 Availability Metrics

Similarly, in [9] the availability metrics of a database can be defined as follow:

\[ A(D, t) = \frac{\sum_{i=1}^{N} w_a A(o_i, t)}{N} * 100\% \]

\( A(o, t) = 0 \) represents the data item \( o \) at time \( t \) is not available while \( A(o, t) = 1 \) represents the state of data item \( o \) at time \( t \) available.

Also, each data object in the database could have different weight.

### 4.3 Survivability Metrics

After having both integrity and availability metrics, we can define survivability as a combination of both:

\[ S_w(D, t) = \sqrt{w_I I(D, t)^2} + w_A A(D, t)^2 \]

Weight is included in this equation because some system may want integrity more while other system may want availability.

### 5 Evaluation Environment and Implementation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS</td>
<td>PostgreSQL 8.4.5</td>
</tr>
<tr>
<td>db_size</td>
<td>10155</td>
</tr>
<tr>
<td>Num_disk</td>
<td>1</td>
</tr>
<tr>
<td>Num_CPU</td>
<td>1</td>
</tr>
<tr>
<td>Max_size</td>
<td>20 row read, 20 row write</td>
</tr>
<tr>
<td>Min_size</td>
<td>3 row read, 3 row write</td>
</tr>
</tbody>
</table>

The testbed is based on TPC-C[8] and PostgreSQL[13]. I choose TPC-C transactions as the background transactions generate because TPC-C Benchmark is an on-line transaction processing (OLTP) benchmark. It is popular for comparing OLTP performance on various hardware and software configurations.

The basic idea is to run TPC-C generated transactions on DBMS with the damage repair and containment proxy and obtain the full log of the transactions. After that an offline analysis will calculate the measurement we need.

Table 1 shows the platform used for the evaluation.
A simulator running on client will simulates multiple clients do business with the online market defined by TPC-C.

Other parameters are shown in table 2.

There are several problems in the implementation of the evaluation. Firstly, it is hard to measure Integrity in run time. Since the measurement of the integrity is basically the same as what the damage assessment. It is impossible to get more accurate data about the damage status than the damage assessment on-the fly, otherwise, I can use this algorithm to improve the damage assessment and improve the performance of the whole attack resilient system. Therefore, offline simulation is used. The basic idea is record the transaction executed in the database with simulated transaction marked as malicious input and then evaluates off-line by analyzing the detailed log. Here another problem rises. In order to do the offline analysis and track the damage spread, we need a log much more detail than what the DBMS will usually do. The disk write of the detailed logging (read, write, xact, etc) often even outruns the disk write of transactions and therefore tends to slow down the system, making the propagation of damage slower by blocking normal transactions in order to keep the log.

6 Evaluation Results

In this section, three configuration of the attack resilient database system will be measured, namely, Damage Confinement only, Damage Repair only and Damage Confinement and Repair. Four evaluations will be made to measure the level of survivability of these different configurations of system can achieve. Integrity Evaluation focuses on the integrity level the three configurations could achieve. Similarly, Availability evaluation focuses on the availability level. The survivability evaluation mainly focuses on evaluating the overall performance combining integrity and availability. The survivability evaluation also measures how good one configuration does under systems with different emphasis on either integrity or availability.

6.1 Integrity Evaluation

The result of integrity evaluation is shown in figure 2. The attack happens at time 0 and at time 50, the IDS reports the attack and the attack resilient system begin to react with the attack.

![Integrity Level](image)

Figure 2 Integrity Level

This result shows that while the damage confinement can confine the damage and keep the integrity level on a constant value. Both the damage repair and damage repair & confinement configuration are able to restore the database back to a health state. And noticeably, the repair & containment configuration restores the system faster.

However, we should notice that the integrity level of the damage repair configuration gets even lower after the first a few se-
conds the attack is reported. This is the result that the repair only configuration tries to repair the damage but not confine them. Therefore, after the repair begin, damage may still be spread out to more data objects and therefore the integrity level continues decreasing. Although in this test case, the repair only configuration is able to restore the database to a health state, in those cases that the speed of spread is so fast and the repair cannot outrun the damage spread, and the system will never return to the health state. In fact, its integrity level will soon become so low that the whole database becomes useless.

6.2 Availability Evaluation

The result of availability evaluation is shown in figure 3. The attack happens at time 0 and at time 50, the IDS reports the attack and the attack resilient system begin to react with the attack.

This result shows that the damage containment configuration will first confine all susceptible data and then unconfine those innocent data and eventually successfully confine the damaged data and keep the availability of database on a constant level. And also we can notice that the Damage repair only configuration always maintains a very high level of availability. The reason is that this configuration will not confine data. So only the currently repairing one is unavailable while all other data can be accessed by outside. For the Repair & Containment configuration, we can see that it successfully restore the database to a fully available state.

One important fact we should notice that the Repair & Containment configuration performs better unconfinement phrase comparing to the Containment configuration and it restores the database to a state of 100% available faster than the Repair only configuration.

6.3 Survivability Evaluation

The result of Survivability Evaluation is shown in figure 4, 5, 6. Each scheme has been tested with three different weights: $wi=wa=0.5$ means the integrity is as important as availability. $wi=0.8, wa=0.2$ means the integrity is more important while $wi=0.2, wa=0.8$ means availability is more important. The attack happens at time 0 and at time 50, the IDS reports the attack and the attack resilient system begin to react with the attack.

This result shows survivability level of the Data Repair configuration. We notice that the Repair only configuration works best if for the database the availability is much more important than the integrity. This is reasonable
because the repair only configuration does not confine any data and therefore always maintains a very high availability level.

Figure 5 shows the survivability level of Data Containment configuration. We can see that the survivability of database drops sharply at time 50 because the containment phrase begins and the availability drops. And the containment only configuration will eventually achieve a consistent level of survivability. The survivability of this configuration is better when the integrity is more important than availability because the confinement itself will harm the availability however, the integrity is ensured by the fact that the damage will not be spread after confinement.

Figure 6 shows the survivability level of Data Containment & Repair configuration. It has similar result comparing to Data Containment only configuration. But they differs in two points: 1) Data Containment & Repair Configuration can restore the database to a health state with 100% survivability level while Containment only cannot do that; 2) Data Containment & Repair Configuration has better performance in the process of improving survivability.

We should notice that although seemingly, the data repair configuration has the best survivability even in the worst case, i.e. put the emphasis on integrity. The Repair configuration has the lowest value of more than 99.67% while the Containment and Repair & Containment ones are 99.48% and 99.50% respectively. We need to bear in mind that there is one fatal disadvantage of this configuration. As we discussed before, in cases that the speed of spread is so fast and the repair cannot outrun the damage spread, and the system will never return to the health state and the system will become useless.

6.4 Detection Latency Evaluation

In this evaluation, only the Containment & Repairing configuration is used. The attack starts at time 0 however the IDS alert will rise
at time 50 and time 100. The impact of detection latency on integrity and availability is shown in figure 7,8.

We can see that the performance of IDS is very vital to the process of repairing. When the delay increases, the integrity level and Availability level will drop obviously and it will take longer time to repair it.

![Figure 8 Latency - Availability](image)

7 Conclusion and Future Work

The results show that the existing solution has the desired properties of Attack-Resilient DBMS. And different configuration is suitable for different system. It shows that Damage Repair configuration works well in system that availability is a priority while Damage Containment works well in systems where integrity is important. If we want the system to repair itself after attack, Damage Containment & Repair configurations could be used. It provides reasonable good survivability and can restore the database to the health data. And if we can make sure that the repair speed will always catch up the damage spread, use the Repair only scheme will provide the best survivability.

However, the impact on performance is not neglectable. In order to make the damage assessment unit work, a full detailed log is needed which will cause more disk write and even block the normal transaction. If this problem is not properly addressed, it is unlikely any real DBMS will adopt those solutions.

Therefore, future research could focus on finding an efficient way to keep the log file and reduce the time of damage assessment by eliminating multiple repeated scans of a large interval of the log file in data confinement and repairing.

References


[13] PostgreSQL
URL http://www.postgresql.org/