Implementation of Agent-based Simulation for Database Security

Rashmi M Jogdand
HOD & Asst.Prof of MCA
Gogte Institute of Technology
Belgaum-590008

Swapna S Banasode
Dept of CSE
Gogte Institute of Technology
Belgaum-590008

ABSTRACT
This paper presents an implementation of secure database access using agent-based simulation. Database and Database System are an essential component of everyday life in modern society. Daily, most of users encounter several activities that involve some interaction with a Database. In multiuser Database there is no restriction on data access, which meant everybody could access anyone else's data because of which malicious destruction or alternation of data can be done. The data stored in the Database need to be protected from unauthorized access and malicious destruction or alternation. In this paper a Agent-Based Simulation program is introduced that includes permission rules for accessing the data and it also includes finding the total number of corrupted data files obtained by applying permission rule to data access so that reliability of the database can be determined using the corrupted data files. This paper aim is to finding the reliability of the database with different types of permission rules.

Keywords
Database, Agent-Based simulation, security, permission rules, **r**eliability of the database.

1. INTRODUCTION
Agent Based Simulation is successfully applied to enterprise modeling and social sciences. Many previous works have not tried simulating scenarios in more technical domains, i.e. simulations of technical systems that are distributed and involve complex interactions between human and machines. Some notable examples were [1] that described an operative framework for a generic database to simulate the security rules applied to it, and verify the various effects the rules have on efficiency, time and data corruption of the database.

In this paper, the aim is to develop a agent-based model for a generic database to simulate the security rules (permission rules) applied to it in order to maintain the reliability of the database. This paper also finds the unreliability of the database system by appropriately setting permission rule parameter for data file access. When agents of the database simultaneously access the several data files according to the permission rule parameter (Read=85% Write=50% delete=10% or Read=30% Write=45% delete=10%) then there is a possibility that data in the data files get corrupted. An index exists to measure, how many times an agent corrupted the data file. Every time agent damages a data file, index of that agent increases for a particular data file. When index crosses a certain limit then DBA (DataBase Administrator) does not provide further privileges for that agent to access data file. So by setting permission rule parameters and varying the number of data files or agents the following can be determined.

1. The possible data corruption using the parameters like permission rules, data files and agents.

2. Reliability of the system.

The rest of this paper is organized as follows:
Section 2 highlights some related works. Section 3 defines the concept of agent. Following which, section 4 introduces the database security policies adopted in this paper. Agent-Based implementation for simulation is then described in section 5, and the experimental findings and analysis of the unreliability values are presented in Section 6. Finally, section 7 discusses the conclusion of the work carried out.

2. RELATED WORK
Database security has been the focus of many database researchers for a long time. In this section some of the relevant work carried out by the researchers in the field of database [2, 3, 4, 5] has been highlighted. Yi Mu [2] proposed the SPM (Schematic Protection model) which is composed of subjects (i.e. users or processes), objects (i.e. files) and ticket. Subject has privileges on objects and can grant privileges to other subjects. A ticket that allows a given privileges to be granted to a subject. Yi Mu also introduced the concept of groups. Group is special abstract object which holds a collection of active (a user) and passive (file) objects. An object can be a member of a group or groups.

Robert [3] explained the securing of DBMS (DataBase Management System) by allowing the grouping and naming of privileges to form Named Protection Domains (NPDs).

Access Control Mechanism such as DAC (Discretionary Access Control) policies and Centralized, Ownership-based and decentralized authorization administration policies for granting and revoking user authorization on database tables and related authorization models such as System R, distributed DBMS System R* authorization models were discussed in reference [4].

Zhu Yangquing et. al. [5] discussed about twice login, audit, program control modules.

In [1] a comprehensive agent-based model for database security was described. Its implementation nevertheless, imposed no restriction on data access, which meant everybody could access anyone else's data. The damaged data were not repaired too. Based on [1], Raymond and Sandeep [6] developed a database security simulation

program that includes permission rules and immediate fixing of corrupted data when they are found.

3. AGENT DEFINED

Many people hearing the word "agent" picture a person or a business entity which is formally authorized to act on another's behalf. Agent in ABS, however, is a totally different story from the agent that we perceive in our daily life.

In 1990s Brustoloni [7] stated that "Autonomous agents are systems capable of autonomous, purposeful actions in the real world". From his brief definition, we see that agents are able to perform actions without human intervention. Brustoloni has confined his agents to live and act in the "real world".

In 1994, Smith, Cypher and Spohrer [7], stated that agent as a persistent software entity dedicated to a specific purpose. 'Persistent' distinguishes agents from subroutines; agents have their own ideas about how to accomplish tasks, their own agends, 'Special Purpose' distinguishes them from entire multifunction applications; agents are typically much smaller".

In 1995, Maes [7] defined autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed.

In 1995, Wooldridge and Jennings [7], stated that the term agent is used to represent a hardware or software–based computer system that enjoys the following features: *Autonomy:* agents operate without the direct intervention of humans or others, and some kind of control over their actions and internal state; *Social ability:* agents interact with other agents (and possibly humans) via some kind of agent-communication language; *Reactivity:* agents perceive their environment, (which may be physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it; *Pro-activeness:* agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative.

From the discussion above, it can be concur that an agent must exhibit following features that can be applied to simulation:

- Situatedness: That is ability to perform actions according to particular input received from outside.
- Reactivity: Agents perceive the context in which they operate and react to it appropriately.

4. DATABASE SECURITY POLICIES

Database security can be defined as a system or process by which the "Confidentiality, Integrity, and Availability (CIA)," of the database can be protected [8]. Unauthorized entry or access to a database server signifies a loss of confidentiality; unauthorized alteration to the available data signifies loss of integrity; and lack of access to database service signifies loss of availability. Loss of one or more of these basic facets will have a significant impact on the security of the database.

A Database Management System (DBMS) is a complex collection of software programs designed for the management of data in a database [1]. To achieve the CIA properties mentioned earlier, several security mechanisms available at DBMS level one of them is DAC mechanism [9]. DAC is based on the concept of access rights or privileges for objects (i.e. tables and views), and mechanisms for giving and revoking users privileges; in this model, the creator of a table or a view automatically gets all privileges on it. The DBMS keeps track of who subsequently gains and losses privileges, and ensures that only requests from users who have the necessary privileges, at the time the request is issued, are allowed [1].

DAC uses the concept of ownership of data. The owner of a data object automatically gets all the permissions on the data, and only the owner can entertain access requests for data he/she owns. Any user who does not have the necessary privileges to the data he/she needs to access must ask the owner for the privileges. The privileges granted by an owner may either be permanent or temporary. When a owner's privileges are revoked, all the users whom he/she has granted permissions to access his/her data will lose their privileges too.

5.AGENT-BASED IMPLEMENTATION

The simulation program is based on an agent-based model for database security described in [6]. This model is extended by adding different types of permission rules and techniques to repair the corrupted data when they are found. The implementation is written fully in C++.

The most important elements of simulation program are 1.The agents (i.e .the users) 2.The data 3.The privileges the agents have on the data. 4. The privileges that the owners of data have granted to other agents. These four crucial elements are implemented as simple objects.

The data objects are the simplest among the four. Each data object is identified by its unique dataId. It also stores the agentId of its creator.

As for the agent objects, each agent is identified uniquely by it's ID. It contains information about the data it owns and information about the agents it has granted privileges to. This is represented with three lists'priv_grant','priv_write','priv_delete'. Where each of these 3 lists contains

1. DataId of the data for which the privileges have been granted, and
2. The agents who have been granted privileges on the data one of, the read stored in 'priv_grant' list or write stored in 'priv_write' list or delete stored in 'priv_delete' list.

Similarly,information about the data, which an agent accesses is stored in 'Privileges' objects.

6. EXPERIMENTAL FINDINGS AND ANALYSIS

Before the simulation starts, the agents and data need to be created. For every data created, the creator or the owner of the data file is automatically granted all the possible privileges on the data. In this implementation, the number of data files created is 49 and agents are equal to 39.

It is necessary to mention that if the data file access involves only reading the data then there is no corruption. If it involves writing, there is possibility that the data might get corrupted. If the data gets corrupted while agent1 is writing the data file, then DBA immediately

repairs the data file for next agent to access and it will increase the reliability index of agent1. If the agent1's reliability index crosses a certain limit -10 in this implementation, then agent1 will not be provided any further privileges as a punishment.

In this paper 6 different cases have been implemented by varying the parameters like the permission rules, data files and agents to find out how many data files get corrupted. So that unreliability of the database system in each case can be determined using the corrupted data files. The unreliability of the database is measured as follows:

Unreliability = Instances of data corruption till time 't'/ Data pieces available at time 't'

6.1 Case 1

Keeping both "Read" permission rule and "Data Files" constant, and increasing both total number of "Agents" and "Write" permission rules.
In this case permission rules used are:
1. Read=100% Write=50% Delete=10%
2. Read=100% Write=75% Delete=10%

6.1.1 For Agents 1 to 39

Figure 1 shows the unreliability of the system over 41 iterations. When privileges were granted in a strict manner with the first permission rule (Read:100% Write:50% Delete:10%), the unreliability was around 0.37 at the end. Note that graph is showing the number of corrupted data found over time.

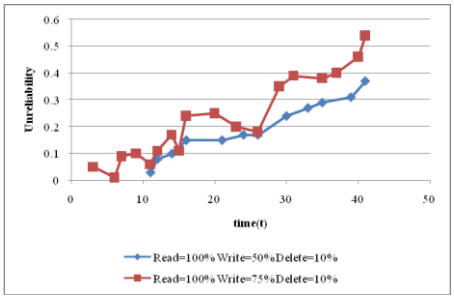

Figure 1: Results of unreliability of the system over 41 iterations

Figure 1 also shows the unreliability of the system over 41 iterations when privileges were granted slightly less strictly with the second permission rule (Read:100% Write:75% Delete:10%). When t=41, the unreliability was about 0.54, this indicates that when rule for granting privileges is not strict enough, the chance of data corruption becomes greater.

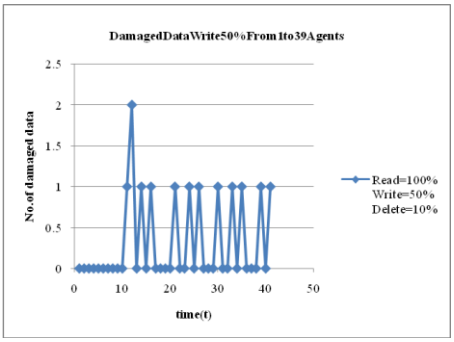

Figure 2: The number of damaged data over 41 iterations for Write=50%

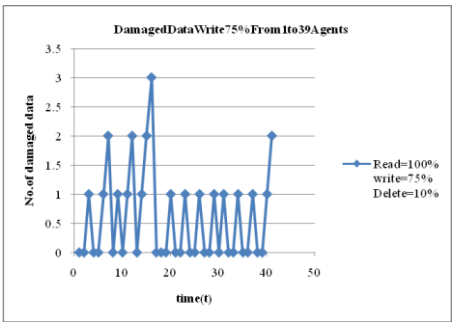

Figure 3: The number of damaged data over 41 iterations for Write=75%

Figure 2 and 3 shows the number of damaged data over 41 iterations. Figure 2 indicates that at any point, there were very few corrupted data. When the permission rule was strict (Read:100% Write:50% Delete:10%), the highest number of corrupted data was 2 when t=12. When the permission rule was less strict (Read:100% Write:75% Delete:10%), the highest number of corrupted data found was 3 when t=16.

6.1.2 For Agents 1 to 53

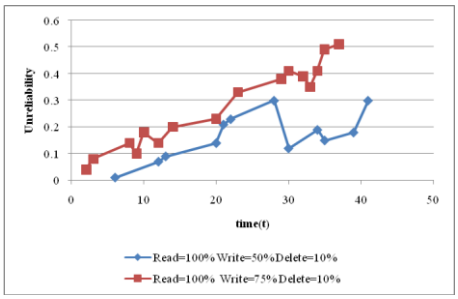

Figure 4: Results of unreliability of the system over 41 iterations

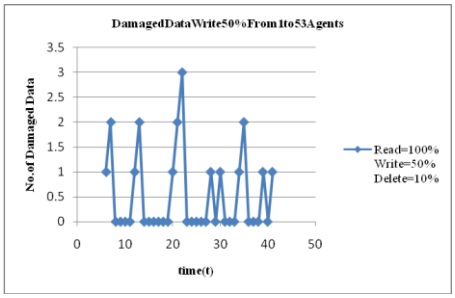

Figure 5: The number of damaged data over 41 iterations for Write=50%

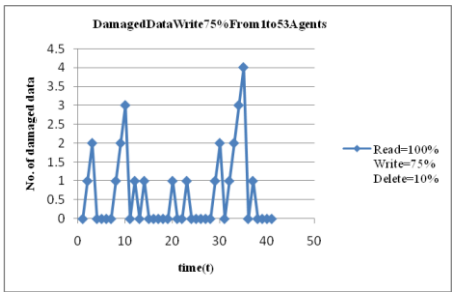

Figure 6: The number of damaged data over 41 iterations for Write=75%

Similarly simulation is carried out by adding 68 agents. And also simulation is carried out for case 2 where by keeping both Write permission rule and Data Files constant and increasing total number of agents and varying read permission rule.

Result table of unreliability values got from case 1 and case 2 with different permission rules is as shown in Table 1.

6.2 Case 3

Keeping both Write permission rule and Agents constant, and increasing total number of data files and varying read permission rule.

In this case permission rules used are:

1.Read=45% Write=50% Delete=10%
2.Read=55% Write=50% Delete=10%

6.2.1 0 to 54 Data files

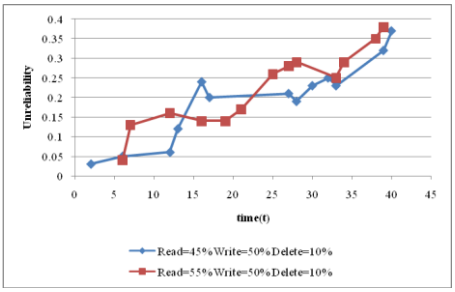

Figure 7: Results of unreliability of the system over 41 iterations

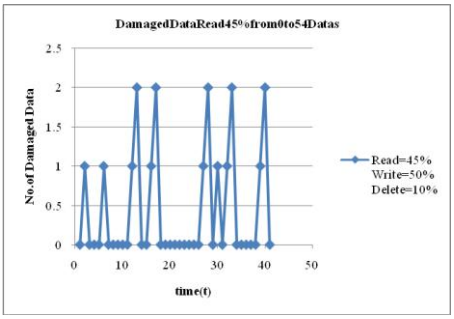

Figure 8: The number of damaged data over 41 iterations for Read=45%

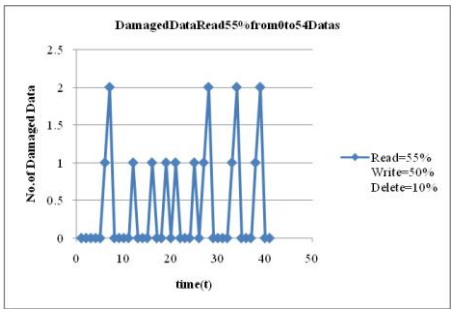

Figure 9: The number of damaged data over 41 iterations for Read=55%

Table 1. Unreliability values got from case 1 and case 2

Data Files	Agents	R=100% W=50%	R=100% W=75%	R=50% W=50%	R=30% W=50%
0 to 49	1 to 39	0.37	0.54	0.35	0.27
	1 to 53	0.30	0.51	0.34	0.32
	1 to 68	0.47	0.61	0.29	0.38

Similarly simulation is carried out for data 0 to 59. And also simulation is carried out for case 4 where by keeping both Read permission rule and Agents constant and increasing total number of data files and varying write permission rule.

Result table of unreliability values got from case 3 and case 4 with different permission rules is as shown in Table 2.

6.3 Case 5

Keeping Write permission rule constant and varying the total number of agents, data files, read permission rule. In this case the permission rules from case 3 and case 4 were used for simulation.

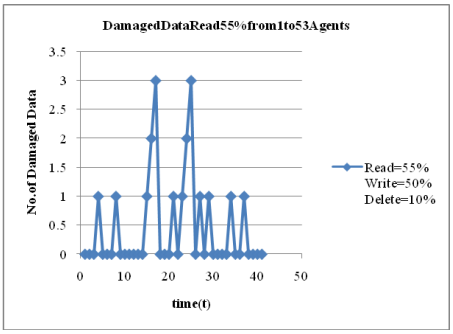

Figure 12: The number of damaged data over 41 iterations for Read=55%

Table 2. Unreliability values got from case 3 and case 4

Data Files	Agents	R=100% W=60%	R=100% W=70%	R=45% W=50%	R=55% W=50%
0 to 54	1 to 39	0.35	0.35	0.37	0.38
0 to 59		0.37	0.45	0.47	0.45

6.3.1 0 to 54 datas And 1 to 53 agents

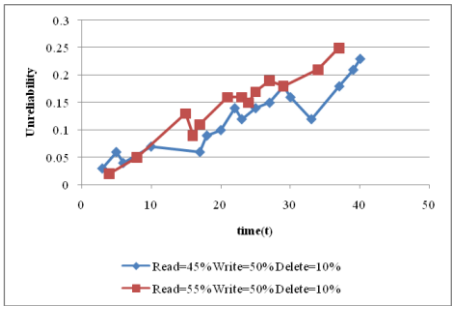

Figure 10: Results of unreliability of the system over 41 iterations

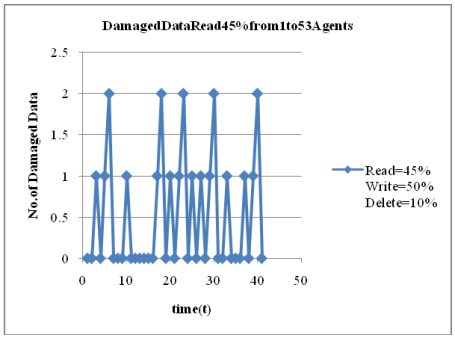

Figure 11: The number of damaged data over 41 iterations for Read=45%

Similarly simulation is carried out for data 0 to 59. And also simulation is carried out for case 6 where by keeping read permission rule constant and varying the total number of agents, data files, write permission rule.

Result table of unreliability values got from case 5 and case 6 with different permission rules is as shown in Table 3.

Analysis of Table 1 shows that with the stricter write permission rule (Read:100% Write:50% Delete:10%), the unreliability of the database system reduced to 0.30 for 53 agents compared to the unreliability of the database for 39 agents and 68 agents. And it also shows that by increasing agents upto 68, the unreliability of the database becomes greater.

Analysis of Table 2 shows that for write permission rule (Read:100% Write:60% Delete:10%), the unreliability of the database increases by increasing number of datas from 49 to 54 upto 59 for 39 agents. And also unreliability of the database increases in each permission rules (Write:70%, Read:45%, Read:55%) by increasing number of datas from 49 to 54 upto 59.

Analysis of Table 3 shows that varying both agents and datas with different permission rules the unreliability of the database is reduced for 53 agents compared to 68 agents.

Table 3. Unreliability values got from case 5 and case 6

Data Files	Agents	R=100% W=60%	R=100% W=70%	R=45% W=50%	R=55% W=50%
0 to 54	1 to 53	0.27	0.25	0.23	0.25
0 to 59		0.29	0.28	0.24	0.22
0 to 54	1 to 68	0.29	0.35	0.35	0.31
0 to 59		0.36	0.40	0.39	0.37

By comparing Table 1 with Table 2 shows that increasing the number of agents upto 53, the reliability of the database is better compared to increasing the number of datas.

By comparing Table 2 with Table 3 shows that reliability of the database is reduced by increasing the number of agents to 53 with varying data files. And also it can be seen that unreliability of the database is reduced for 68 agents compared to keeping only 39 agents with increasing number of data files upto 59.

7. CONCLUSION AND FUTURE DIRECTIONS

In this paper, Agent-Based model for database security is implemented using permission rules. The database security can be maintained better with stricter access to data through the manipulation of the permission rules. By comparison of result tables it can be concluded that by keeping the agents upto 53, the unreliability of the system can be reduced. As the number of the agents increases to 63, the unreliability of the system also increases with both strict permission rule as well as for less strict permission rule because of which the number of corrupted data becomes greater. As the number of data corrupted increases with the increase in number of agents, then there is a possibility of the database being collapsed.

The Future plan of action will be: 1)Improving the reliability of database system by increasing number of agents minimum upto 70 and maximum 120 for 49 data files.

2) Exploring the Agent-Based simulation program for large database.

3) Creating a more complex hierarchical structure for data access. In this paper, the DBA granted privileges to access data by all agents. In Future plan the owners of data files will grant privileges to access data by other agents from any host.

4) Encrypting the data files using Encryption algorithm.

8. REFERENCES

[1] Macro Remondino.2004. Multi-Agent Based Simulation For Database Security: A framework In Proceedings of 18th Europen Simualtion Multiconference.

[2] Yi Mu and Vijay Varadharajan. Towards a Protection Model for supporting Multiple Access Control Policies.

[3] Robert W. Baldwin.1990. Naming and Grouping Privileges to simplify security management in Large Databases.

[4] Elisa Bertino and Elena Ferrari. Data Security. 22nd International Computer Software and Applications Conference, IEEE Computer Society, Vienna.

[5] Zhu Yangqing.et al. 2009. Design of A New Web Database Security Model. 2nd International Symposium on Electronic Commerce and Security.

[6] Raymond Chiong and Sandeep Dhakal. 2008. Modelling Database Security through Agent-Based Simulation. 2nd Asia International Conference on Modelling and Simulation.

[7] Stan Franklin and Art Graesser.1996. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Proceedings of the 3rd international workshop on Agent Theories, Architectures, and Languages,Springer-Verlag.

[8] Macro Vieria and Henrique Madeira. 2005. Towards a Security Benchmark for Database Management Systems. In Proceedings of the International Conference on Dependable Systems and Networks.

[9] Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems.4th Edition book.